Adaptive Method of LINES

Edited by

A. Vande Wouwer
Ph. Saucez
W. E. Schiesser

CHAPMAN & HALL/CRC

Boca Raton London New York Washington, D.C.

Library of Congress Cataloging-in-Publication Data

Adaptive method of lines / edited by A. Vande Wouwer, Ph.Saucez, W.E. Schiesser.
p. cm.
Includes bibliographical references and index.
ISBN 1-58488-231-X (alk. paper)
1. Differential equations, Partial—Numerical solutions. I. Wouwer, A.Vande (Alain)
II. Saucez, Ph. (Philippe) III. Schiesser, W.E.

QA377 .A294 2001
515′.353—dc21 00-069347

International Standard Book Number 1-58488-231-X
Library of Congress Card Number 00-069347
Printed in the United States of America 1 2 3 4 5 6 7 8 9 0
Printed on acid-free paper

Preface

Partial differential equations (PDEs) arise in the mathematical description of a spectrum of chemical and physical problems. This broad utility of PDEs is illustrated in this book with, for example, applications in chemical kinetics, heat and mass transfer, hydrology, electromagnetism, and astrophysics. The PDE models are usually highly nonlinear and therefore require numerical analysis and computer-based solution techniques.

The numerical method of lines (MOL) is a comprehensive approach to the solution of time-dependent PDE problems that basically proceeds in two steps: (1) spatial derivatives are first approximated using, for example, finite difference or finite element techniques, and (2) the resulting system of semi-discrete (discrete in space — continuous in time) ordinary differential equations (ODEs) is integrated in time. The success of this method follows from the availability of high-quality numerical algorithms and associated software for the solution of stiff systems of ODEs.

Even though most of the ODE solvers automatically adjust the time-step size (and possibly the order of the integration formula) in order to meet stability and accuracy requirements, the conventional MOL proceeds only in a semi-automatic way since the spatial nodes are held fixed for the entire course of the computation. For problems developing large spatial transitions, such as steep moving fronts or shocks, this conventional approach can be inefficient since a large number of uniformly distributed nodes is required to adequately capture the regions of high spatial activity. Unfortunately, most of the nodes are "wasted" in regions of low spatial activity and it is therefore desirable to use a procedure that adapts the spatial grid — move or add/delete nodes — so as to concentrate them in the regions where they are needed, i.e., to track and accurately resolve important small-scale features. "Adaptive method of lines" refers to the concept of both *temporal* and *spatial adaptivity* in solving time-dependent PDEs.

The very active MOL community has traditionally shared their algorithms, codes, and results with others. This book resulted from the joint efforts of a group of authors who have the privilege of knowing and working with each other.

The purpose of this book is threefold:

1. To provide an introduction to the MOL and the concepts of time and space adaptation

2. To present a variety of applications from physics and engineering science

3. To describe new methods and codes and to highlight current research

Hence, this book is intended for engineers, physicists, and applied mathematicians who are not familiar with the MOL, as well as for numerical analysts interested in recent research results. The book includes several chapters that cover various aspects of time and space adaptivity in the method of lines.

Chapter 1 is introductory, and surveys the basic concepts of spatial discretization and time integration in the general MOL formulation. Then, an overview of several grid adaptation mechanisms is given, including moving grids and grid refinement, static and dynamic gridding, the equidistibution principle and the concept of a monitor function, the minimization of a functional, and the moving finite element method. The several methods are illustrated with different test examples from engineering and science, which show the great diversity of potential applications addressed by the MOL and adaptive grid techniques.

Chapter 2, titled "Application of the Adaptive Method of Lines to Nonlinear Wave Propagation Problems," continues the introduction through a series of modest one-dimensional (1D) problems solved by an adaptive grid refinement algorithm which equidistributes an arc-length or a curvature monitor function subject to constraints on the grid regularity. This algorithm is applied to several model PDEs describing nonlinear dispersive wave phenomena, including the cubic Schrödinger equation, the derivative nonlinear Schrödinger equation, the classical Korteweg-de Vries (KdV) equation, the Korteweg-de Vries-Burgers equation, and a fully nonlinear KdV equation giving rise to compactons. Numerical results for the propagation and the interaction of solitary waves are discussed in terms of computational expense and solution accuracy.

Chapter 3, titled "Numerical Solutions of the Equal Width Wave Equation Using an Adaptive Method of Lines," deals with the equal width (EW) wave equation, which is a model partial differential equation for the simulation of 1D wave propagation in media with nonlinear wave steepening and dispersion processes. The background of the EW equation is reviewed and this equation is solved by using an advanced numerical method of lines with an adaptive grid whose node movement is based on an equidistribution principle. The solution procedure is described and the performance of the solution method is assessed by means of computed solutions and error measures. Many numerical solutions are presented to illustrate important features of the propagation of solitary waves, the interaction of inelastic solitary waves, the inelastic solitary waves, the breakup of a Gaussian pulse into solitary waves, and the development of an undular bore.

Chapter 4, titled "Adaptive Method of Line for Magnetohydrodynamic PDE Models," is devoted to magnetohydrodynamic PDE models, which describe many interesting phenomena from astrophysics. These PDE systems consist of conservation laws for mass, momentum, total energy, and induction of magnetic fields. Very often, these models possess solutions having high spatial gradients that also move rapidly in time, such as steep moving temperature fronts, rotating sharp pulses, or shock waves.

In this chapter, an adaptive moving grid method is described that can be used to follow the steep gradients of the solutions in time. The method is based on an equidistribution principle enhanced with smoothing terms in the spatial and temporal direction. To follow the MOL approach, the semi-discretized PDE models that are transformed to a moving frame are coupled to the ODEs for the grid motion. A suitable stiff time-integrator is needed to obtain the fully discretized numerical solutions. Numerical results are shown for some interesting test cases: a magnetic shock-tube problem, a model for the propagation of shear Alfvén waves, and a model that describes the advection of a current-carrying cylinder.

Chapter 5, titled "Development of a 1D Error-Minimizing Moving Adaptive Grid Method," focuses on the design of a discretization technique dedicated to grid adaptation. This so-called compatible scheme allows the leading term of the local residual to be evaluated directly in terms of a local error in the numerical solution (the numerical modeling error). An error-dependent smoothing technique is used to ensure that higher-order error terms are negligible. The numerical modeling error is minimized by means of grid adaptation. Fully converged adapted grids with strong local refinements are obtained for a steady-state shallow-water application with a hydraulic jump. An unsteady application confirms the importance of taking the error in time into account when adapting the grid in space. The shortcomings of the present implementation and the remedies currently under development are discussed.

Chapter 6, titled "An Adaptive Method of Lines Approach for Modeling Flow and Transport in Rivers," considers practical problems in hydrology, which require the accurate simulation of flow and/or transport in natural rivers. Three particular applications are discussed: (1) the forecasting of water levels, (2) the simulation of the transport of soluble substances, and (3) a two space-dimensional (2D) calculation of flood planes. The basic equations for the simulation of flow and transport in rivers are presented and the method of lines is proposed for their numerical solution. Due to the hyperbolicity of the flow equations, Godunov-type upwind schemes are applied to space discretization, whereas time-integration of the semi-discretized PDEs is done by a special variant of the well-known fourth order Rosenbrock-Wanner method RODAS. Finally, the use of adaptive space meshes for the current problems is discussed and the efficiency of the proposed numerical solution methods is demonstrated through some realistic applications.

Chapter 7, titled "An Adaptive Mesh Algorithm for Free Surface Flows in General Geometries," devises a numerical method for computing incompressible free surface flows in general, three space-dimensional (3D) geometries. Adaptive mesh refinement as described by Berger and Colella [*J. Comp. Phys.,* 82, (1989), pp. 64–84] and Almgren et al. [*J. Comp. Phys.,* 142, (1998), pp. 1–46] is used. The free surface separating the gas and liquid is modeled using "embedded boundary" techniques, which allow for the arbitrary merge and break-up of fluid mass while maintaining excellent mass conservation. An embedded boundary method is also used to represent irregular geometries, e.g., ship hull. Computational results are presented for 3D jetting problems and 3D ship wave problems. In the process of describing the adaptive Cartesian grid algorithm for incompressible flow, a new (easy) way for enforcing the

contact angle boundary condition at points where the free surface meets the geometry is presented.

Chapter 8, titled "The Solution of Steady PDEs on Adjustable Meshes in Multidimensions Using Local Descent Methods," focuses on a variant of the multidimensional Moving Finite Element (MFE) method for steady PDEs called Least-Squares Moving Finite Elements (LSMFE). The MFE method is an adaptive method of lines approach in which the mesh movement is generated by an extended Galerkin method. The discovery of an optimal property of the steady MFE equations has recently led to LSMFE. In addition, the implementation of a local approach to the movement of the mesh provides new control in the adjustment of the mesh. This new approach can also be used in a similar way with other space discretization techniques in multidimensions, for example in finite-dimensional approximations in variational principles, which can include the least-squares best-fit problem, and least-squares methods for conservation laws. In the latter case, a link has been shown with the equidistribution of residuals. In this chapter, the techniques are analyzed and 2D examples, including the advection equation, a shallow water application with a hydraulic jump and the Euler equations, are given.

Chapter 9, titled "Linearly Implicit Adaptive Schemes for Singular Reaction-Diffusion Equations," is concerned with modified adaptive difference schemes for solving degenerate nonlinear reaction-diffusion equations with singular source terms. Differential equation problems play important roles in mathematical models of steady and unsteady combustion processes. Both semi-adaptive and fully adaptive schemes for solving the aforementioned problems are discussed. In the former case, an adaptive, or moving mesh, mechanism in time is considered, while in the latter, adaptation both in time and space are constructed. Modified monitor functions based on the arc length of the rate function u_t are obtained. Properties of the numerical schemes are analyzed and it is shown that under proper smoothness, consistency, and stepsize constraints, the numerical solution preserves the monotonicity of the physical solution. Numerical experiments with quenching phenomena in reaction-diffusion problems are given to further demonstrate the monotonicity and convergence properties of the methods.

Chapter 10, titled "Adaptive Linearly Implicit Methods for Heat and Mass Transfer Problems," deals with a combination of linearly implicit time integrators of Rosenbrock type and adaptive multilevel finite elements based on *a posteriori* error estimates. In the classical MOL approach, the spatial discretization is done once and for all. Here, a local spatial refinement is allowed in each time step, which results in a discretization sequence first in time then in space. The spatial discretization is considered as a perturbation, which has to be controlled within each time step. This approach has proven to work quite satisfactorily for a wide range of challenging practical problems. The performance of the adaptive method is demonstrated for two applications that arise in the study of flame balls and brine transport in porous media.

Chapter 11, titled "Unstructured Adaptive Mesh MOL Solvers for Atmospheric Reacting Flow Problems," discusses the application of the method of lines to reacting flow problems in combustion and atmospheric dispersion. The chapter describes the

finite volume spatial discretization methods used and indicates the error estimation approach employed to guide spatial mesh adaptation. The integration methods employed in time are extensions of existing MOL codes with careful treatment of the nonlinear equations which minimizes the computation cost without sacrificing accuracy. Examples from a number of large-scale problems are used to illustrate the approach employed.

Chapter 12, titled "Two-Dimensional Model of a Reaction-Bonded Aluminum Oxide Cylinder," concentrates on a particular chemical engineering application. The reaction-bonded aluminum oxide process utilizes the oxidation of intensely milled aluminum and Al_2O_3 powder compacts that are heat-treated in air to make alumina-based ceramics. A two-dimensional, simultaneous mass and energy balance model is developed in cylindrical coordinates to describe this process. The model describes the propagation of an ignition front that has been observed during reaction-bonding. The model is solved using the method of lines and spatial remeshing techniques based on the equidistribution principle and spatial regularization procedures introduced in Chapters 1 and 2.

Chapter 13, titled "Method of Lines within the Simulation Environment DIVA for Chemical Processes," introduces the simulation environment DIVA, which is an integrated numerical tool for modeling, simulation, analysis, and optimization of single chemical process units as well as integrated production plants. Attention is focused on the symbolic preprocessing tool SYPPROT, which allows an automatic method of lines discretization of chemical process models with distributed parameters. A symbolic model formulation with finite difference and finite volume discretization schemes on fixed as well as moving spatial grids is provided. These methods allow a convenient model implementation and a flexible application of the MOL. The use of the preprocessing tool and the numerical methods in DIVA are illustrated with application examples from chemical engineering.

In summary, the authors of these chapters have provided an introduction to the adaptive method of lines, and applications ranging from modest 1D PDEs, to complex 2D and 3D PDE systems. In the process of covering this spectrum of applications, the authors discuss state-of-the-art numerical algorithms for the adaptive solution of PDEs in space and time that produce solutions to difficult PDE problems requiring, in particular, high spatial resolution. This book evolved from the fruitful collaboration among the chapter authors, and could not have been achieved without their motivation and enthusiastic support. In order to continue the development of adaptive methods for PDEs, the authors welcome inquiries about their work.

Alain Vande Wouwer
Philippe Saucez
William Schiesser

Contributors

I. Ahmad SSO NSLD (CHASSNUPP), P.O. Box 113, Islamabad, Pakistan, dridreesahamad@yahoo.com

M.J. Baines Department of Mathematics, University of Reading, P.O. Box 220, Reading, RG6 6AX, U.K., m.j.baines@reading.ac.uk

M. Berzins School of Computing, The University of Leeds, Leeds LS2 9JT, U.K., martin@comp.leeds.ac.uk

M. Borsboom WF | Delft Hydraulics, Marine Coastal and Industrial Infrastructure, P.O. Box 177, 2600 MH Delft, The Netherlands, mart.borsboom@wldelft.nl

H.S. Caram Department of Chemical Engineering, Iacocca Hall, 111 Research Drive, Lehigh University, Bethlehem, Pennsylvania 18015, U.S.A., hsc0@lehigh.edu

H.M. Chan Department of Materials Science and Engineering, Whitaker Laboratory No. 5, Lehigh University, Bethlehem, Pennsylvania 18015, U.S.A., hmc0@lehigh.edu

B. Erdmann Scientific Computing, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Takustrasse 7, 14195 Berlin-Dahlem, Germany, erdmann@zib.de

S. Ghorai Department of Mathematics, Indian Institute of Technology, Kanpur, Kanpur-208016, India, sghorai@iitk.ac.in

E.D. Gilles Max-Planck-Institute for Dynamics of Complex Technical Systems, Leipziger Str. 44, D-39120 Madgeburg, Germany, gilles@mpi-madgeburg.mpg.de

J.J. Gottlieb Institute for Aerospace Studies, University of Toronto, 4925 Dufferin Street, Toronto, Ontario, M3H 5T6, Canada, gottlieb@bach.utias.utoronto.ca

S. Hamdi Institute for Aerospace Studies, University of Toronto, 4925 Dufferin Street, Toronto, Ontario, M3H 5T6, Canada, samir.hamdi@utoronto.ca

J.S. Hansen Institute for Aerospace Studies, University of Toronto, 4925 Dufferin Street, Toronto, Ontario, M3H 5T6, Canada, hansen@bach.utias.utoronto.ca

M.P. Harmer Department of Materials Science and Engineering, Whitaker Laboratory No. 5, Lehigh University, Bethlehem, Pennsylvania 18015, U.S.A., mph2@lehigh.edu

R. Keppens F.O.M. Institute for Plasma Physics 'Rijnhuizen', P.O. Box 1207, 3430 BE, Nieuwegein, The Netherlands, keppens@rijnh.nl

A.Q. Khaliq Department of Mathematics, Western Illinois University, Macomb, Illinois 61455, U.S.A., a-khaliq@wiu.edu

A. Kienle Max-Planck-Institute for Dynamics of Complex Technical Systems, Leipziger Str. 44, D-39120 Madgeburg, Germany, kienle@mpi-madgeburg.mpg.de

R. Köhler Institut für Systemdynamik und Regelungstechnik, Universität Stuttgart, Pfaffenwaldring 9, D-70550 Stuttgart, Germany, koehler@isr.uni-stuttgart.de

J. Lang Scientific Computing, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Takustrasse 7, 14195 Berlin-Dahlem, Germany, lang@zib.de

M. Mangold Max-Planck-Institute for Dynamics of Complex Technical Systems, Leipziger Str. 44, D-39120 Madgeburg, Germany, mangold@mpi-madgeburg.mpg.de

K.D. Mohl Institut für Systemdynamik und Regelungstechnik, Universität Stuttgart, Pfaffenwaldring 9, D-70550 Stuttgart, Germany, mohl@isr.uni-stuttgart.de

P. Rentrop IWRMM, Universität Karlsruhe (TH), Engesser Str. 6, Kaiserin-Augusta-Anlagen 15-17, D-76128 Karlsruhe, Germany, sekretariat@iwrmm.math.uni-karlsruhe.de

Ph. Saucez Laboratoire de Mathématique et Recherche Operationelle, Faculté Polytechnique de Mons, 7000 Mons, Belgium, saucez@mathro.fpms.ac.be

W.E. Schiesser Iacocca Hall, 111 Research Drive, Lehigh University, Bethlehem, Pennsylvania 18015, U.S.A., wes1@lehigh.edu

H. Schramm Institut für Systemdynamik und Regelungstechnik, Universität Stuttgart, Pfaffenwaldring 9, D-70550 Stuttgart, Germany, schramm@isr.uni-stuttgart.de

Q. Sheng Department of Mathematics, University of Louisiana, Lafayette, Louisiana, 70504-1010, U.S.A., qxs2336@louisiana.edu

E. Stein Max-Planck-Institute for Dynamics of Complex Technical Systems, Leipziger Str. 44, D-39120 Madgeburg, Germany, stein@mpi-madgeburg.mpg.de

G. Steinebach Bundesanstalt für Gewässerkunde, Kaiserin-Augusta-Anlagen 15-17, D-56068 Koblenz, Germany, steinebach@bafg.de

M. Sussman Department of Mathematics, Florida State University, Tallahassee, Florida 32306, U.S.A., sussman@math.fsu.edu

A.S. Tomlin Department of Fuel and Energy, , The University of Leeds, Leeds LS2 9JT, U.K., A.Tomlin@chemistry.leeds.ac.uk

J. Ware 35 Gun Place, 86 Wapping Lane, Wapping, London, U.K., justinware@onetel.net.uk

M.J. Watson Department of Chemical Engineering, Iacocca Hall, 111 Research Drive, Lehigh University, Bethlehem, Pennsylvania 18015, U.S.A., watsonmj@apci.com

A. Vande Wouwer Laboratoire d'Automatique, Faculté Polytechnique de Mons, 7000 Mons, Belgium, vdw@autom.fpms.ac.be

P.A. Zegeling Mathematical Institute, Utrecht University, Budapestlaan 6, 3584 CD Utrecht, The Netherlands, zegeling@math.uu.nl

M. Zeitz Institut für Systemdynamik und Regelungstechnik, Universität Stuttgart, Pfaffenwaldring 9, D-70550 Stuttgart, Germany, zeitz@isr.uni-stuttgart.de

Contents

Chapter 1

Introduction

Alain Vande Wouwer, Philippe Saucez, and William Schiesser

1.1 Classification of Partial Differential Equations

Partial differential equations (PDEs) are one of the most widely used forms of mathematics in science and engineering. This is due in large part to the three-dimensional form of our physical world, and its variation with time. Thus, PDEs have four independent variables, that is, three spatial dimensions and time. The variation of physical properties, e.g., density, velocity, momentum, and energy, is expressed by PDEs in terms of partial derivatives. For example, if ρ denotes density, then the dependency of density on space, $\mathbf{x}$, and time, t, can be denoted as $\rho(\mathbf{x}, t)$, where $\mathbf{x}$ is a three-vector (a vector with three components), and partial derivatives signify the variation of density with space and time. For example,

$$\frac{\partial \rho}{\partial t} \Leftrightarrow \rho_t$$

is the first order partial derivative of ρ with respect to t. Note that the partial derivative, $\frac{\partial \rho}{\partial t}$, can also be expressed as a subscripted variable, ρ_t.

A PDE that expresses the variation of ρ with $\mathbf{x}$ and t for a fluid, the *equation of continuity,*

$$\frac{\partial \rho}{\partial t} = \nabla \cdot (\mathbf{v}\rho) \tag{1.1}$$

states a basic physical principle, *conservation of mass,* where

$\mathbf{v}$	fluid velocity vector
∇	divergence operator

∇ is a vector differential operator that has three components in specific coordinate systems, for example, see Table 1.1.

When working with PDEs, we may also require the gradient of a scalar, see Table 1.2.

Table 1.1 $\nabla\cdot$ (*divergence of a vector*)

Coordinate System	Components
Cartesian	$\left[\begin{array}{l} [\nabla]_x = \dfrac{\partial}{\partial x} \\ [\nabla]_y = \dfrac{\partial}{\partial y} \\ [\nabla]_z = \dfrac{\partial}{\partial z} \end{array}\right]$
cylindrical	$\left[\begin{array}{l} [\nabla]_r = \dfrac{1}{r}\dfrac{\partial}{\partial r}(r\,) \\ [\nabla]_\theta = \dfrac{1}{r}\dfrac{\partial}{\partial \theta} \\ [\nabla]_z = \dfrac{\partial}{\partial z} \end{array}\right]$
spherical	$\left[\begin{array}{l} [\nabla]_r = \dfrac{1}{r^2}\dfrac{\partial}{\partial r}(r^2\,) \\ [\nabla]_\theta = \dfrac{1}{r\sin\theta}\dfrac{\partial}{\partial \theta}(\sin\theta\,) \\ [\nabla]_\phi = \dfrac{1}{r\sin\theta}\dfrac{\partial}{\partial \phi} \end{array}\right]$

Finally, when working with PDEs, we often require a combination of the two preceding vector differential operators, i.e., the divergence of the gradient of a scalar, see Table 1.3.

The derivation of $\nabla\cdot\nabla$ (the *Laplacian*) follows directly from the preceding components of $\nabla\cdot$ (divergence of a vector in Table 1.1) and ∇ (gradient of a scalar in Table 1.2).

Cartesian coordinates:

$$\nabla\cdot\nabla = \left(\mathbf{i}\frac{\partial}{\partial x} + \mathbf{j}\frac{\partial}{\partial y} + \mathbf{k}\frac{\partial}{\partial z}\right)\cdot\left(\mathbf{i}\frac{\partial}{\partial x} + \mathbf{j}\frac{\partial}{\partial y} + \mathbf{k}\frac{\partial}{\partial z}\right) = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$$

Cylindrical coordinates:

$$\nabla\cdot\nabla = \left(\mathbf{i}_r\frac{1}{r}\frac{\partial}{\partial r}(r) + \mathbf{j}_\theta\frac{1}{r}\frac{\partial}{\partial \theta} + \mathbf{k}_z\frac{\partial}{\partial z}\right)\cdot\left(\mathbf{i}_r\frac{\partial}{\partial r} + \mathbf{j}_\theta\frac{1}{r}\frac{\partial}{\partial \theta} + \mathbf{k}_z\frac{\partial}{\partial z}\right)$$

Table 1.2 ∇ (*gradient of a scalar*)

Coordinate System	Components
Cartesian	$\left[\begin{array}{c} [\nabla]_x = \dfrac{\partial}{\partial x} \\ [\nabla]_y = \dfrac{\partial}{\partial y} \\ [\nabla]_z = \dfrac{\partial}{\partial z} \end{array}\right]$
cylindrical	$\left[\begin{array}{c} [\nabla]_r = \dfrac{\partial}{\partial r} \\ [\nabla]_\theta = \dfrac{1}{r}\dfrac{\partial}{\partial \theta} \\ [\nabla]_z = \dfrac{\partial}{\partial z} \end{array}\right]$
spherical	$\left[\begin{array}{c} [\nabla]_r = \dfrac{\partial}{\partial r} \\ [\nabla]_\theta = \dfrac{1}{r}\dfrac{\partial}{\partial \theta} \\ [\nabla]_\phi = \dfrac{1}{r\sin\theta}\dfrac{\partial}{\partial \phi} \end{array}\right]$

$$= \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial}{\partial r}\right) + \frac{1}{r}\frac{\partial}{\partial \theta}\left(\frac{1}{r}\frac{\partial}{\partial \theta}\right) + \frac{\partial}{\partial z}\left(\frac{\partial}{\partial z}\right)$$

$$= \frac{1}{r}\left(\frac{\partial}{\partial r} + r\frac{\partial^2}{\partial r^2}\right) + \frac{1}{r^2}\frac{\partial}{\partial \theta}\frac{\partial}{\partial \theta} + \frac{\partial}{\partial z}\frac{\partial}{\partial z}$$

$$= \left(\frac{\partial^2}{\partial r^2} + \frac{1}{r}\frac{\partial}{\partial r}\right) + \frac{1}{r^2}\frac{\partial^2}{\partial \theta^2} + \frac{\partial^2}{\partial z^2}$$

Spherical coordinates:

$$\nabla \cdot \nabla = \left(\mathbf{i}_r \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\right) + \mathbf{j}_\theta \frac{1}{r\sin\theta}\frac{\partial}{\partial \theta}(\sin\theta) + \mathbf{k}_\phi \frac{1}{r\sin\theta}\frac{\partial}{\partial \phi}\right)$$

$$\cdot \left(\mathbf{i}_r \frac{\partial}{\partial r} + \mathbf{j}_\theta \frac{1}{r}\frac{\partial}{\partial \theta} + \mathbf{k}_\phi \frac{1}{r\sin\theta}\frac{\partial}{\partial \phi}\right)$$

$$= \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial}{\partial r}\right) + \frac{1}{r\sin\theta}\frac{\partial}{\partial \theta}\left(\sin\theta\,\frac{1}{r}\frac{\partial}{\partial \theta}\right) + \frac{1}{r\sin\theta}\frac{\partial}{\partial \phi}\left(\frac{1}{r\sin\theta}\frac{\partial}{\partial \phi}\right)$$

Table 1.3 $\nabla \cdot \nabla$ (*divergence of the gradient of a scalar*)

Coordinate System	Component
Cartesian	$\dfrac{\partial^2}{\partial x^2} + \dfrac{\partial^2}{\partial y^2} + \dfrac{\partial^2}{\partial z^2}$
cylindrical	$\left(\dfrac{\partial^2}{\partial r^2} + \dfrac{1}{r}\dfrac{\partial}{\partial r}\right) + \dfrac{1}{r^2}\dfrac{\partial^2}{\partial \theta^2} + \dfrac{\partial^2}{\partial z^2}$
spherical	$\dfrac{1}{r^2}\dfrac{\partial}{\partial r}\left(r^2\dfrac{\partial}{\partial r}\right) + \dfrac{1}{r^2\sin\theta}\dfrac{\partial}{\partial \theta}\left(\sin\theta\ \dfrac{\partial}{\partial \theta}\right) + \dfrac{1}{r^2\sin^2\theta}\dfrac{\partial^2}{\partial \phi^2}$

$$= \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\ \frac{\partial}{\partial r}\right) + \frac{1}{r^2\sin\theta}\frac{\partial}{\partial \theta}\left(\sin\theta\ \frac{\partial}{\partial \theta}\right) + \frac{1}{r^2\sin^2\theta}\frac{\partial^2}{\partial \phi^2}$$

Thus, the equation of continuity in Cartesian coordinates (from (1.1) and Table 1.1) is:

$$\begin{aligned}\frac{\partial \rho}{\partial t} &= \left(\mathbf{i}\frac{\partial}{\partial x} + \mathbf{j}\frac{\partial}{\partial y} + \mathbf{k}\frac{\partial}{\partial z}\right)\cdot\left(\mathbf{i}v_x\rho + \mathbf{j}v_y\rho + \mathbf{k}v_z\rho\right)\\ &= \frac{\partial}{\partial x}\left(v_x\rho\right) + \frac{\partial}{\partial y}\left(v_y\rho\right) + \frac{\partial}{\partial z}\left(v_z\rho\right) \qquad (1.2)\end{aligned}$$

The term $\dfrac{\partial}{\partial x}\left(v_x\rho\right)$ in (1.2) has a clear physical meaning. The term in parentheses, $(v_x\rho)$, is the *mass flux* in the x direction. This interpretation is suggested by the units of this term, e.g.,

$$(m/s)\left(kg/m^3\right) = kg/s - m^2\ .$$

Thus, $(v_x\rho)$ is the kg/s of fluid flowing through a unit area in the x direction (a mass flux). Consequently, $\dfrac{\partial}{\partial x}\left(v_x\rho\right)$ is the change in this flux with x. We might expect, intuitively, that this term could undergo a sharp change with respect to x, and this is indeed the case; that is, (1.2) can propagate sharp changes or fronts, and even discontinuities, which is the main reason why first-order equations such as (1.2) (note that it has only first-order derivatives in x and t) are generally difficult to integrate numerically. This conclusion suggests that we might benefit from classifying PDEs as a way of anticipating the general properties of their solutions.

The conventional *geometric classification* of PDEs as *elliptic, hyperbolic,* or *parabolic* is expressed through a single, linear PDE. However, this classification is quite restrictive and therefore not very useful for most applications; we therefore adopt a less rigorous, but more general, geometrical classification that is illustrated by Table 1.4.

Note that there are two classes of hyperbolic PDEs, i.e., first and second order. The two are related. For example, if we define two variables

$$v = u_x,\ w = u_t$$

Table 1.4 Geometric Classification of PDEs with Examples

Order in x (BV)	Order in t (IV)	Classification	Example
1	1	First-order hyperbolic	$\frac{\partial u}{\partial t} = -v\frac{\partial u}{\partial x}$ (advection equation)
2	2	Second-order hyperbolic	$\frac{\partial^2 u}{\partial t^2} = c^2\frac{\partial^2 u}{\partial x^2}$ (wave equation)
2	1	Parabolic	$\frac{\partial u}{\partial t} = \alpha\frac{\partial^2 u}{\partial x^2}$ (Fourier's or Fick's second law)
2(in x, y)	0	Elliptic	$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$ (Laplace's equation)

then by differentiation

$$v_t = u_{xt},\ w_x = u_{tx}\ .$$

If the mixed partial derivatives u_{xt} and u_{tx} are assumed equal, we have

$$v_t = w_x \tag{1.3}$$

Also, from the wave equation in Table 1.4,

$$w_t = c^2 v_x \tag{1.4}$$

Thus, the wave equation (a second-order hyperbolic PDE) is expressed in terms of two first-order hyperbolic PDEs, (1.3) and (1.4).

Combinations of these classes of PDEs are also possible. For example,

$$\frac{\partial u}{\partial t} = -v\frac{\partial u}{\partial x} + D\frac{\partial^2 u}{\partial x^2} \tag{1.5}$$

is *hyperbolic-parabolic*. The second derivative, $D\frac{\partial^2 u}{\partial x^2}$, which is the x-component of the Laplacian in Cartesian coordinates from Table 1.3, generally describes diffusion (as in Fourier's second law from Table 1.4). Thus, (1.5) is also called a *convective-diffusive* equation as reflected in the $-v\frac{\partial u}{\partial x}$ (convection with velocity v) and $D\frac{\partial^2 u}{\partial x^2}$ (diffusion with diffusivity D) terms.

In order to have a well-posed (complete) PDE problem specification, *auxiliary conditions* must also be specified. For example, in the case of (1.5), which is first-order in t, and second-order in x, one initial condition (IC) is required (for t), and

two boundary conditions (BCs) are required (for x). These might be, for example,

$$u(x,0) = f(x) \tag{1.6}$$

$$u(0,t) = u_0 \tag{1.7}$$

$$\frac{\partial u(L,t)}{\partial x} = 0\,. \tag{1.8}$$

The notation for specifying auxiliary conditions is to denote the specific value of the independent variable. For example, $t = 0$ is specified in initial condition (1.6) as $u(x,0)$ [an alternative would be to write $u(x, t = 0)$]. Boundary conditions are of three types:

BC Type	Example
Dirichlet	$u(0,t) = u_0(t)$
Neumann	$\dfrac{\partial u(L,t)}{\partial x} = u_x(L,t) = g(t)$
third-type or Robin	$D\dfrac{\partial u(0,t)}{\partial x} = Du_x(0,t) = v(u(0,t) - u_0(t))$

Note that BCs are generally specified at different values of the independent variable, e.g., $x = 0$ and $x = L$, while ICs are specified at a single value of the independent variable, e.g., $t = 0$. As the names imply, boundary conditions typically reflect what is happening at the boundaries of a physical system, and initial conditions specify how the system starts out (and then evolves according to the PDE).

Dirichlet BCs specify the value of the dependent value at a specific value of the independent variable such as (1.7), while *Neumann* BCs specify the derivative of the dependent variable with respect to the independent variable such as (1.8). A combination of Dirichlet and Neumann BCs is termed a boundary condition of the *third type* or a *Robin* BC.

Equation (1.5) is an example of a PDE with *constant coefficients* (assuming the velocity v and diffusivity D are constant). PDEs can also have coefficients that are functions of the independent variables, that is, *variable coefficients*. For example, Fourier's second law in cylindrical coordinates (using the Laplacian in cylindrical coordinates from Table 1.3)

$$\frac{\partial u}{\partial t} = -v\frac{\partial u}{\partial z} + D\left(\frac{\partial^2 u}{\partial z^2} + \frac{\partial^2 u}{\partial r^2} + \frac{1}{r}\frac{\partial u}{\partial r} + \frac{1}{r^2}\frac{\partial^2 u}{\partial \theta^2}\right)\,. \tag{1.9}$$

The term $\dfrac{1}{r}\dfrac{\partial u}{\partial r}$ has the variable coefficient $\dfrac{1}{r}$ (a function of the independent variable r), and the term $\dfrac{1}{r^2}\dfrac{\partial^2 u}{\partial \theta^2}$ has the variable coefficient $\dfrac{1}{r^2}$.

We conclude this discussion of the geometric classification of PDEs by asking whether this serves a useful purpose. The answer is a threefold "yes." When we describe a PDE system as elliptic, hyperbolic, or parabolic, we:

- Immediately convey a concise description of some of its important mathematical features. For example, an elliptic problem has no initial value variable.
- Suggest approaches to the numerical solution of the PDE. For example, the solution of an elliptic problem cannot include initial value integration unless an initial value variable is added to the elliptic problem, but in such a way that it will have no final effect on the solution as it evolves numerically.
- Become aware of the potential difficulties in computing a numerical solution. For example, we can anticipate that a hyperbolic problem might produce discontinuities or other difficult mathematical forms that must be accommodated in the numerical calculation of the solution.

Thus, in using the terminology of geometric classification, we immediately give a useful description of the PDE problem, and an indication of what must be done to solve it numerically.

All of the PDEs that have been considered thus far have been *linear* or *first degree.* That is, the dependent variable, u, and all of its derivatives have been to the first power (the degree should not be confused with the order of the derivative; for example $\dfrac{\partial^2 u}{\partial z^2}$ in (1.9) is first degree, but second order).

A second major classification of PDEs is according to their linearity, that is, *linear* as just described, or *nonlinear,* with the dependent variable and/or its derivatives not first degree (or to the first power). For example, $\left(\dfrac{\partial^2 u}{\partial z^2}\right)^3$ is second order, but third degree.

Nonlinear PDEs are an essential part of the PDE mathematical description of many physical systems. The linearity of a system is an important classification since generally, mathematical methods for solving nonlinear PDEs are unavailable; that is, generally we don't know how to solve nonlinear PDEs mathematically or analytically. The situation is analogous to that of solving nonlinear algebraic and transcendental equations; generally this cannot be done mathematically either (there are, of course, special case exceptions). However, we will observe in subsequent parts of this chapter, and throughout this book, that numerical methods can solve systems of nonlinear PDEs. In fact, there is no fundamental limit to the solution of nonlinear PDE problems numerically, although each new problem generally has to be considered on a case by case basis; that is, numerical procedures have to be developed for the specific problem system. In fact, the central topic of this book, the adaptive method of lines, is a general procedure for the numerical solution of nonlinear PDEs.

To conclude this preliminary discussion of nonlinearity, consider (1.9) with an additional term added to the RHS

$$\frac{\partial u}{\partial t} = -v\frac{\partial u}{\partial z} + D\left(\frac{\partial^2 u}{\partial z^2} + \frac{\partial^2 u}{\partial r^2} + \frac{1}{r}\frac{\partial u}{\partial r} + \frac{1}{r^2}\frac{\partial^2 u}{\partial \theta^2}\right) + k_0 e^{-E/(Ru)} \qquad (1.10)$$

where k_0, E, and R are constants. Note that the exponential $e^{-E/(Ru)}$ contains the dependent variable, u, in a nonlinear form (this can be confirmed by expanding the

exponential function in a Taylor series that will include powers of u; therefore in this series expansion, u is not to the first degree or power, and so (1.10) is a nonlinear PDE).

Boundary conditions can also be nonlinear. As an example, the boundary condition

$$k\frac{\partial u(L,t)}{\partial x} = \varepsilon(u_L^4 - u^4(L,t))$$

is nonlinear because of the term $u^4(L,t)$. In general, nonlinear boundary conditions will preclude an analytical solution to the associated PDE in the same way as if the nonlinearity appeared in the PDE; in other words, we generally don't know how to solve PDEs analytically that have nonlinear boundary conditions.

A third classification of PDEs (in addition to the geometric and linearity classifications discussed previously) that will have particular relevance in the remainder of this book is the smoothness of the PDE solutions. Specifically, PDEs can have solutions that change very abruptly in space and, because of the PDE, they will therefore also change abruptly in time. Additionally, these abrupt changes, which are also called *steep fronts,* can move in space as the solution evolves in time, that is, *steep moving fronts.* In the extreme, the steep moving fronts can be discontinuous (i.e., can be in the form of discontinuities).

The resolution of steep moving fronts so as to accurately determine where the fronts are, and what form they take, is generally a difficult computational problem in the numerical solution of a PDE system. We must look at the spatial regions where the rapid change takes place in greater detail than in the spatial regions where the solution is relatively smooth. But this implies that we know the location of the rapid changes, and what form they take, so that we can use enhanced numerical methods in those regions, even as the location of these regions changes. In other words, the numerical algorithm for the PDE solution must be *adaptive,* either through intervention by the analyst, or automatically as part of the numerical algorithm. This latter characteristic, the adaptive solution of PDEs, is the central topic of this book.

To illustrate how PDEs can propagate mathematical forms that are difficult to handle numerically, we start with (1.2). If we consider one dimension only, x, and take the velocity as constant, $v_x = v$, (1.2) becomes the *linear advection equation*

$$\frac{\partial u}{\partial t} = -v\frac{\partial u}{\partial x} \tag{1.11}$$

where u is used in place of ρ (the dependent variable of a PDE is commonly designated as u in the numerical analysis literature).

The simplicity of (1.11) is deceptive because it is also one of the most difficult PDEs to integrate numerically. To illustrate this, we consider an initial condition (one is required because of the first-order derivative $\dfrac{\partial u}{\partial t}$)

$$u(x,0) = f(x) \tag{1.12}$$

and a boundary condition (one is required because of the first-order derivative $\frac{\partial u}{\partial x}$)

$$u(0,t) = 0\,. \tag{1.13}$$

The solution to (1.11), subject to (1.12) and (1.13), is easily derived as

$$u(x,t) = f(x - vt) \tag{1.14}$$

which is known as a *traveling wave solution* since it is the same function f everywhere in the displaced spatial coordinate $x - vt$. For example, if $v > 0$ and the solution starts out as the initial function of (1.12), the solution will be this same function traveling left to right with velocity v.

To see how difficult (1.11) can be to solve, we consider as the specific initial condition function the *Heaviside unit step function,* $h(x)$

$$f(x) = h(x) \begin{array}{l} = 0, x < 0 \\ \\ = 1, x > 0\,. \end{array}$$

Thus, from (1.14), we see that the solution to (1.11) for this case is

$$u(x,t) = h(x - vt) \begin{array}{l} = 0, x - vt < 0 \\ \\ = 1, x - vt > 0\,. \end{array}$$

This solution has a finite discontinuity (unit jump) at $x = vt$. In other words, the solution is a unit step traveling left to right at velocity v. Equation (1.11) with the discontinuous initial condition $h(x)$ is an example of a *Riemann problem.*

At $x = vt$, the derivative $\frac{\partial u}{\partial x}$ in (1.11) is undefined, so in a sense, this is an impossible problem to solve numerically. Various approximations for computing the solution will be considered briefly in the next section as examples of different approaches to the Riemann problem.

Finally, if we write the one-dimensional (x only) version of (1.2) as (again with u in place of ρ)

$$\frac{\partial u}{\partial t} = -\frac{\partial (vu)}{\partial x} \tag{1.15}$$

or in subscript notation

$$u_t + F(u)_x = 0 \tag{1.16}$$

where $F(u) = vu$ is a *flux function* (note again that it has the units of a flux), and (1.16) is written in conservation form, i.e., it is a *conservation law* equation. The Riemann problem for (1.16) has a discontinuous initial condition

$$u(x,0) \begin{array}{l} = u_-, x < 0 \\ \\ = u_+, x > 0 \end{array}$$

where u_- and u_+ are unequal initial values of u. An extensive numerical analysis literature exists for the solution of conservation law equations and associated Riemann problems. In the next section we will consider only a few basic numerical methods for these problems.

1.2 The Method of Lines

Consider the PDE problem

$$\mathbf{u}_t = \mathbf{f}(\mathbf{u}), \qquad \mathbf{x}_L < \mathbf{x} < \mathbf{x}_R, \quad t > 0 \tag{1.17}$$

where

$$\begin{aligned} \mathbf{u}_t &= \frac{\partial \mathbf{u}}{\partial t} \\ \mathbf{u} &= \text{vector of dependent variables} \\ t &= \text{initial value independent variable} \\ \mathbf{x} &= \text{boundary value independent variables} \\ \mathbf{f} &= \text{spatial differential operator} \\ &= \mathbf{f}(\mathbf{x}, t, \mathbf{u}, \mathbf{u}_\mathbf{x}, \mathbf{u}_{\mathbf{xx}}, \cdots) \end{aligned}$$

Note that in order to discuss a system of PDEs with a dependent or solution vector $\mathbf{u}$, a bold-face variable denotes a vector and again, a subscript denotes a partial derivative. $\mathbf{x}$ is a three-vector that, for example, can have components (x, y, z) in Cartesian coordinates, (r, θ, z) in cylindrical coordinates, and (r, θ, φ) in spherical coordinates. Equation (1.17) is therefore quite general, and can encompass all of the PDEs considered previously in one, two, and three spatial dimensions plus time. For example, if $\mathbf{f}(\mathbf{x}, t, \mathbf{u}, \mathbf{u}_\mathbf{x}, \mathbf{u}_{xx}, \cdots) = -vu_x$, we have the scalar advection equation (1.11).

The *method of lines* (MOL) is a computational approach for solving PDE problems of the form of (1.17) that proceeds in two separate steps: first, spatial derivatives, e.g., $\mathbf{u}_\mathbf{x}, \mathbf{u}_{xx}, \cdots$, are approximated using, for instance, *finite difference* (FD) or *finite element* (FE) techniques. Second, the resulting system of semi-discrete ODEs in the initial value variable is integrated in time, t.

To illustrate the MOL, we again consider the linear advection equation (1.11) approximated on a spatial grid in x of N grid points separated uniformly by a distance Δx. If the spatial derivative $\dfrac{\partial u}{\partial x}$ is replaced with a second-order, centered FD at grid point i,

$$\frac{\partial u}{\partial x} = -v\frac{u_{i+1} - u_{i-1}}{2\Delta x} + O(\Delta x^2), i = 1, 2, \cdots, N \tag{1.18}$$

then substitution of this approximation in (1.11) with $v = 1$ gives a system of N

ODEs

$$\frac{du_i}{dt} = -\frac{u_{i+1} - u_{i-1}}{2\Delta x}, i = 1, 2, \cdots, N \tag{1.19}$$

Note spatial grid index i has the values corresponding to a system of N initial value ODEs that can be integrated by a library ODE integrator. Of course, in the process, the initial condition (1.12) must be specified at the N grid points; also, u_0 and u_{N+1} are *fictitious points* (outside the spatial domain) that must be included in the ODEs for $i = 1$ and $i = N$ (methods for using boundary conditions to handle boundary and fictitious points will be considered subsequently).

If the solution to (1.19) is computed for two unit step functions, separated by an interval in t of 50 (a square pulse), with $N = 101$ (or 100 intervals of length Δx), the MOL solution is oscillatory as indicated in Figure 1.1(a) (the solid line is the exact solution). This is an example of the first form of numerical distortion, i.e., *numerical oscillation,* and for this example, the oscillation does not diminish significantly with increasing numbers of grid points, N (even though the FD approximation is second order, i.e., $O(\Delta x^2)$, which indicates that as Δx decreases, the error of the FD approximation decreases as Δx^2, depending on some conditions that we will not discuss here). Thus, we come to the conclusion that even though the individual terms (derivatives) in a PDE are approximated by what seems to be reasonable (accurate) approximations, when these approximations are substituted in the PDE, the resulting numerical solution can be highly inaccurate. In fact, a large part of what is discussed subsequently in this book pertains to the choice and implementation of approximations that give accurate solutions to the PDEs.

In general, the accuracy of the numerical solution will depend on the smoothness of the actual (analytical) solution. For the preceding problem, the solution $u(x, t)$ has two jump discontinuities, and as a consequence, the centered FD approximation of $\dfrac{\partial u}{\partial x}$, produces unrealistic oscillations. If, however, the initial condition is not discontinuous (as it was with the square pulse in Figure 1.1(a)), but rather, is a triangular pulse that is continuous in $u(x, t)$, but discontinuous in $\dfrac{\partial u}{\partial x}$, the MOL solution is much closer to the true solution as indicated in Figure 1.1(b).

If the initial condition is a smooth cosine pulse, the agreement between the MOL and the true solution is even better as demonstrated in Figure 1.1(c). Thus, we see that the performance (accuracy) of a particular approximation of the PDE depends on the conditions of the problem, in this case, the smoothness of the initial condition. If the initial condition has a jump or discontinuity [as does $u(x, 0) = h(x)$], this jump will propagate in space and time, which is a hallmark characteristic of hyperbolic PDEs.

The preceding example demonstrates the essential features of the MOL solution of PDEs, i.e., algebraic approximation of the spatial derivatives, followed by integration of the resulting system of initial values ODEs. We also observed that the approximation of the spatial derivatives is a critical step. In the next section, we consider other approximations to the spatial derivative in (1.11), $\dfrac{\partial u}{\partial x}$, that could conceivably give better solutions than in Figures 1.1(a), (b), and (c).

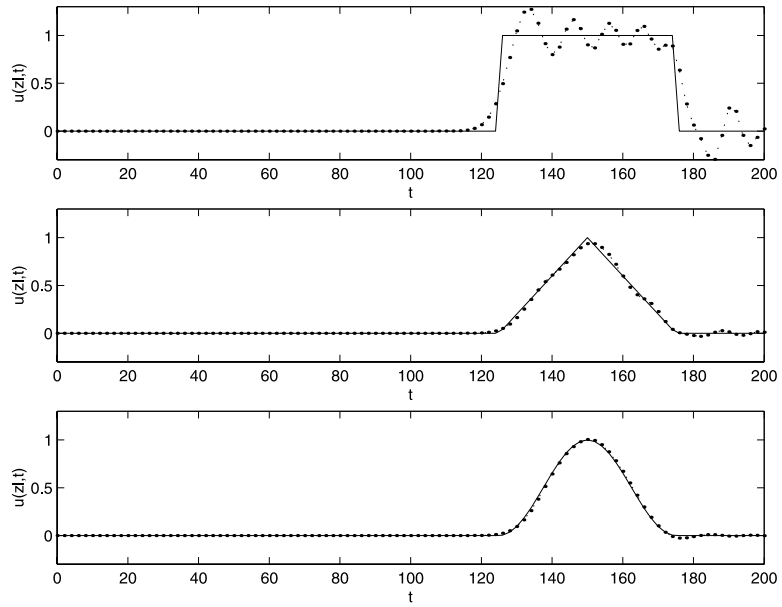


FIGURE 1.1
(1.11) approximated as (1.19).

1.2.1 Spatial Discretization

In this book dedicated to PDE problems developing steep spatial moving fronts, approximation of first-order (convective) terms (i.e., $\mathbf{u_x}$) will play a central role. To avoid undesirable oscillations in the solution profiles, it is generally necessary to resort to upwind spatial approximations. A variety of methods are available, including *upwind finite differences, upwind orthogonal collocation,* TVD (*total variation diminishing*) schemes [11] such as *flux limiters* and ENO (*essentially non-oscillatory*) schemes, etc. Recently, TVD *centered methods* have also been proposed [19], which have the advantage that no *a priori* information on the flow direction is required.

To demonstrate the idea of upwinding, we now use as the approximation of $\dfrac{\partial u}{\partial x}$ in (1.11) the first-order, two point upwind FD

$$\frac{\partial u}{\partial x} = \frac{u_i - u_{i-1}}{\Delta x} + O(\Delta x), i = 1, 2, \cdots, N \tag{1.20}$$

so that the system of ODEs becomes

$$\frac{du_i}{dt} = -v\frac{u_i - u_{i-1}}{\Delta x}, i = 1, 2, \cdots, N\ . \tag{1.21}$$

As we see in Figure 1.2(a), the numerical oscillation from the centered approximation considered previously, (1.18), has been eliminated, but now we have excessive

rounding or *numerical diffusion* in the MOL solution, which is the second major form of numerical distortion, i.e., we have now observed numerical oscillation and numerical diffusion. Further, this numerical diffusion is not substantially reduced by increasing N, and it also persists for the triangular and cosine pulses, as shown in Figures 1.2(b) and (c). The approximation for $\dfrac{\partial u}{\partial x}$, (1.20), is called an upwind FD because it uses, in addition to the point of the approximation, i, the point upwind, $i-1$ (for $v>0$), but not the point downwind, $i+1$, as in the preceding centered approximation (1.18) ($i+1$ would be the upwind point for $v<0$ so that generally for upwind approximations, we need to know the direction of flow, i.e., the sign of the velocity).

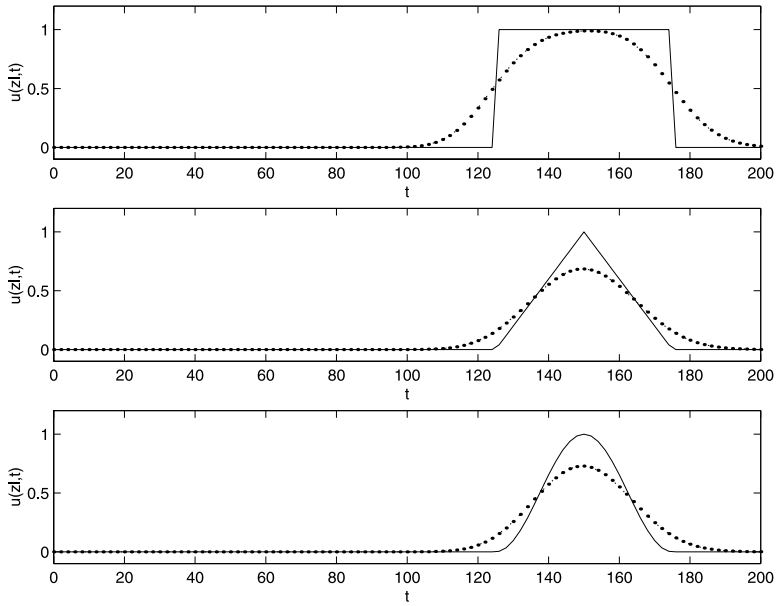


FIGURE 1.2
(1.11) approximated as (1.21).

At the boundaries corresponding to $i=1$ and $i=N$, we can take as the ODEs

$$u_1 = 0,\ \frac{du_1}{dt} = 0 \tag{1.22}$$

corresponding to BC (1.13), and

$$\frac{du_N}{dt} = \frac{u_N - u_{N-1}}{\Delta x} \tag{1.23}$$

so that (1.23) does not involve the fictitious point $i=N+1$.

What has been done with centered and upwind FD approximations, (1.18) and (1.20), cannot be improved to the point that numerical oscillation and diffusion are

essentially eliminated in the MOL solution. In other words, *linear approximations* of spatial derivatives [note that the dependent variable u appears linearly in (1.18) and (1.20)] will have these distortions to varying degrees. To essentially eliminate oscillation and substantially reduce diffusion, *nonlinear approximations* must be used. A spectrum of nonlinear approximations can be considered, e.g., ENO methods. Here we briefly mention *flux limiters.* In Figures 1.3(a), (b), and (c), the MOL solution to (1.11) for the square, triangular, and cosine pulses are given with the *van Leer flux limiter* used to approximate $\dfrac{\partial u}{\partial x}$; we observe good agreement with the analytical solutions. In Figures 1.4(a), (b), and (c), the corresponding MOL solutions are given for the *Smart flux limiter.* Thus, we observe that for the problem of (1.11), nonlinear approximations (such as flux limiters) are effective for computing accurate numerical solutions, and more generally for hyperbolic problems with steep moving fronts, these nonlinear approximations give accurate solutions. Thus, these and other approximations for these difficult problems will be considered subsequently. As the name "flux limiter" suggests, the value of the flux function $F(u)$ in (1.16) is limited to avoid numerical distortion in the numerical solutions, [e.g., of (1.11)], particularly the elimination of numerical oscillation.

1.2.2 Time Integration

Spatial discretization usually produces a system of *stiff* ODEs (ODEs with widely separated eigenvalues). As the stability restriction of an explicit time integration method is inversely proportional to some power of the grid spacing, Δx (this power is usually equal to the order of the highest spatial derivative) [14], the time step restriction in a finely gridded region (with small Δx) can be much more severe than in coarsely gridded regions. Hence, standard explicit Runge–Kutta methods may be computationally inefficient and implicit methods, e.g., BDF (backward differentiation formula) or implicit RK solvers, are better suited to solve these problems. The choice of an ODE integrator is an important aspect of the MOL solution of PDE systems. However, we will not consider various ODE integration algorithms and the associated computer codes in detail at this point in order to limit this discussion of the MOL to a reasonable length. Rather, the choice of an ODE integrator will be addressed through example applications in the remainder of this chapter and in the subsequent chapters. Fortunately, a broad choice of quality library ODE integrators is available and, in fact, one of the major advantages of the MOL approach to PDE systems is the opportunity to use the advances in ODE integrators and their associated codes.

1.3 Adaptive Grid Methods

In the classical MOL, whereas the time step size is automatically adjusted by the ODE solver, the spatial grid x_i, $i = 1, 2, \ldots, N$, is usually held constant over

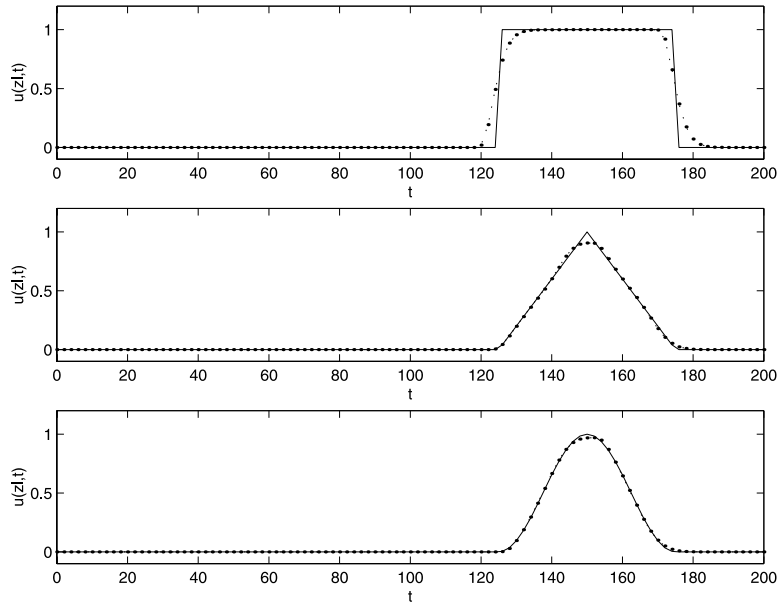


FIGURE 1.3
(1.11) approximated with van Leer flux limiter.

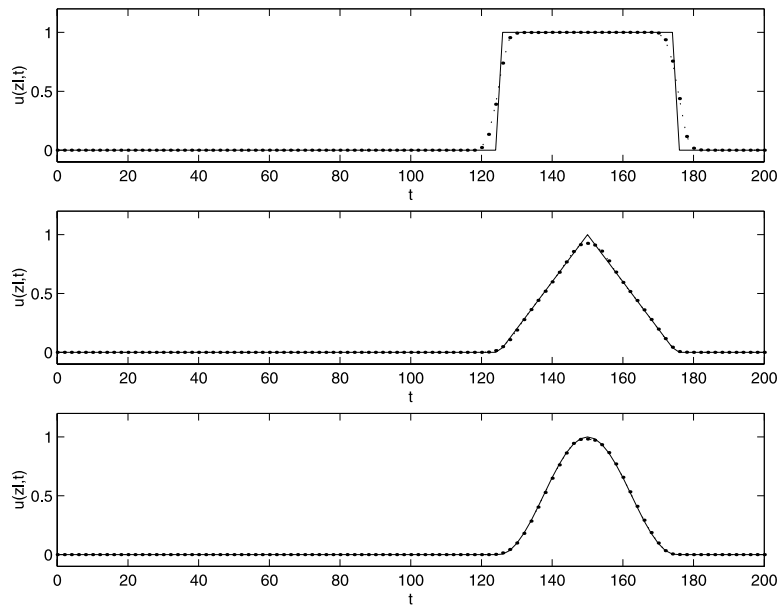


FIGURE 1.4
(1.11) approximated with Smart flux limiter.

the complete integration time interval. However, if for the problem under study, the model PDEs develop steep moving fronts (such as shock waves in compressible flows, phase boundaries during nonequilibrium thermal processes, etc.), a very fine spatial grid is required over the whole spatial domain to capture and resolve the high spatial gradients. Outside of these regions of high spatial activity, a large number of nodes are "wasted," resulting in unnecessary computational expense and lack of spatial resolution of important small-scale solution features.

Over the last 20 years, a great deal of interest has developed in procedures with *time and space adaptation* and various sophisticated techniques have been proposed. Adaptive grid methods can be classified in several ways, according to the criterion used to update the grid and to the temporal evolution of the grid point number and location. Following a set of reference papers [6, 8, 12, 14, 29], these several concepts are introduced in the next sections.

1.3.1 Grid Adaptation Criteria

One of the major approaches for defining the spatial node movement is based on the equidistribution principle, i.e., the grid points x_i, $i = 1, 2, \ldots, N$ are moved so that a specified quantity, also called the monitor function $m(u)$, is equally distributed over the spatial domain, i.e.,

$$\int_{x_{i-1}}^{x_i} m(u)dx = \int_{x_i}^{x_{i+1}} m(u)dx = c, \qquad 2 \le i \le N-1 \tag{1.24}$$

or in discrete form

$$M_{i-1}\Delta x_{i-1} = M_i \Delta x_i = c, \qquad 2 \le i \le N-1 \tag{1.25}$$

where $\Delta x_i = x_{i+1} - x_i$ is the local grid spacing, M_i is a discrete approximant of the monitor function $m(u)$ in the grid interval $[x_i, x_{i+1}]$, and c is a constant.

Equidistribution principles have been used in many different ways to numerically solve PDEs having solutions with steep moving fronts. One of the earliest attempts is due to White [36], who used the arc-length of the solution

$$m(u) = \sqrt{\left(\alpha + \|u_x\|_2^2\right)}\,. \tag{1.26}$$

Another possibility is to equidistribute a measure of the local curvature or spatial truncation error, e.g., [15]

$$m(u) = \|u_{xx}\|\ . \tag{1.27}$$

The second major approach to define the node movement is to minimize a functional depending on error measures and/or grid structure properties. A method belonging to this latter category is described by Hyman [14] and Petzold [23] who define the grid movement by minimizing a measure consisting of a combination of node velocities and time derivatives of the solution. The minimization of both the time variation in

the solution and the time variation in the grid leads to a slowly varying Jacobian of the semi-discrete ODE system and in turn, to reduced computational expense.

Probably the most important representative in this category is the moving finite element method of Miller et al. [9, 20, 21], where the error measure is the square of the residual of the PDE written in finite element form. ODEs for the solution and the nodal (grid point) positions are obtained by minimizing the integral of this error measure with respect to the time derivatives of the nodal positions and amplitudes.

1.3.2 Static vs. Dynamic Gridding

Following the MOL philosophy, static gridding algorithms proceed in four basic steps:

1. approximation of the spatial derivatives on a fixed nonuniform grid
2. integration of the resulting semi-discrete ODEs over N_{adapt} time steps
3. adaptation of the spatial grid
4. interpolation of the solution to produce initial conditions on the new grid

The main advantage of this approach is that the PDE solution and the grid adaptation procedure are uncoupled. Hence, it is easily implemented and allows the use of several artifices such as a variable number of nodes (i.e., grid refinement). The main disadvantages are: (1) the time integration is halted periodically (or at every time step if $N_{\text{adapt}} = 1$) to adapt the spatial grid (resulting in a computational overhead due to the frequent solver restarts); consequently, as the grid points are moving at discrete times only, they may be ineffectively placed (resulting in large temporal gradients when a steep moving front crosses some of the grid points, and therefore the ODE solver is required to use extremely small step sizes to retain accuracy), and (2) an interpolation technique is required to transfer the data from the old to the new grid.

Another approach is to move the grid points continuously in time, i.e., to use dynamic gridding, so that their locations follow the moving front and remain near optimal. This way, the moving front is less likely to cross a grid point and longer time steps can be taken. The development of this approach requires the introduction of the *Lagrangian formulation* [8] of the PDE problem (1.17). For this purpose, consider the continuous time trajectories of the grid points

$$x_R = x_1 < x_2(t) < \cdots < x_i(t) < \cdots < x_N = x_R \ . \tag{1.28}$$

Along $x(t) = x_i(t)$ the total temporal derivative of u is given by

$$\dot{u} = u_t + \dot{x}\, u_x = f(u) + \dot{x}\, u_x \ . \tag{1.29}$$

The ODEs defining the grid point movement, i.e., $\dot{x} = g(t)$, can be derived based on some physical *a priori* knowledge, such as a flow-related quantity. For example, in the case of the advection equation

$$u_t = -v u_x \tag{1.30}$$

a natural choice is to attach the node movement to the fluid velocity, i.e., $\dot{x} = v$, so that $\dot{u}\ (x(t), t) = 0$ along the characteristic curves. This very simple example highlights the main objective of this approach, i.e., to minimize the temporal variation of the solution (in the moving reference frame) so as to allow the largest possible time step sizes.

Another approach for defining the grid point movement is to express a spatial equidistribution principle in differential form so as to equally distribute a monitor function such as the arc length of the solution (1.26). In this case, grid point movement ensures a smoothing of the problem in space, but does not necessarily reduce the temporal variation of the solution.

As stressed in [8], it is generally not possible to fulfill both objectives, i.e., temporal and spatial smoothing, simultaneously.

1.3.3 Moving Grid and Grid Refinement Algorithms

Grid refinement algorithms are methods that change the number of nodes as time evolves. Most of these methods start with a fixed global grid (or base grid) and proceed to locally add nodes in the regions of high spatial activity and to remove nodes outside of these regions. Refinement is essential if solutions are to be calculated to a prescribed level of accuracy (this approach is close to that used in ODE solvers where as many time steps as needed are taken to bring the local truncation error to within prespecified tolerances). As a consequence of local node additions and removals, the overall grid structure may become complicated in the sense of the accompanying data structures and internal boundary treatment between fine and coarse grids. As the grid is adapted at discrete time levels only, local refinement methods belong to the static gridding category.

On the other hand, moving grid methods concentrate a constant number of nodes in the regions of high spatial activity. Node movement can be accomplished continuously in time, i.e., these methods fall into the dynamic gridding category. However, with a fixed number of nodes, local resolution is achieved at the expense of depreciating the resolution in other regions. The situation becomes critical when there are not enough nodes to describe the complete solution profile. For example, in the case of several steep moving fronts acting in different regions of the spatial domain, the numerical computation encounters problems if the grid is following one front and another one arises somewhere else. Since the number of nodes is fixed throughout the entire course of the computation, no new grid structure is created for the new front, but rather the old grid has to adjust itself abruptly. This incorrect transient restricts the size of the time steps and diminishes the overall efficiency of the method. These problems can be alleviated by combining node movement and grid refinement.

1.3.4 Grid Regularity

The accuracy of the spatial derivative approximation and the stiffness of the resulting system of semi-discrete ODEs is determined to a large extent by the regularity

of the grid point spacing and the smoothness of the ODE temporal trajectories. As adaptive grid methods tend to concentrate the grid points in regions of high spatial activity, spatial grid distortion as well as abrupt change in the node distribution must be limited through spatial and temporal regularization procedures (for example, using constraints on the grid spacing in the spatial equidistribution procedure, penalty terms in the functional to be minimized, etc.). These regularization procedures always make the method more complicated in the sense that they involve a set of additional tuning parameters.

1.4 Case Studies

This section is devoted to the numerical study of several application examples taken from physics and engineering. At this stage, we would like to illustrate the various concepts introduced thus far, to show the diversity of adaptive grid algorithms and to demonstrate their potential to address challenging applications. Of course, the selection of particular methods and applications is intended only to illustrate basic concepts and methods, and this section is by no means exhaustive. What follows is just a sampling of the spectrum of adaptive grid methods published in the literature over the last 20 years.

1.4.1 Case Study 1

As a first test-example, consider Burgers' equation

$$u_t = -u\, u_x + \epsilon u_{xx}, \quad 0 < x < 1, \quad t > 0 \tag{1.31}$$

with initial and Dirichlet boundary conditions taken from the exact solution

$$u(x,t) = (0.1r_1 + 0.5r_2 + r_3)/(r_1 + r_2 + r_3) \tag{1.32}$$

where $r_1(x,t) = e^{(-x+0.5-4.95t)/20\varepsilon}$, $r_2(x,t) = e^{(-x+0.5-0.75t)/4\varepsilon}$, and $r_3(x,t) = e^{(-x+0.375)/2\varepsilon}$. By adjusting the numerical value of the viscosity coefficient ϵ, a broad spectrum of convection-diffusion problems can be generated. Here, we consider a medium value of $\epsilon = 0.001$, which leads to moderate front steepening.

This problem is solved using ANUGB, a local refinement algorithm developed by Hu and Schiesser [15]. In this method, the grid adaptation criterion is based on the solution curvature. A uniform base grid x_i, $i = 1, \ldots, N_b$, which is the foundation upon which all subsequent grids are built, is first defined. The second-order derivative of the solution is estimated at each of the base grid points in order to locate the regions of high spatial activity; a set of threshold values for $\|u_{xx}(x_i)\|$ controls the addition or subtraction of new grid points. When $\|u_{xx}(x_i)\|$ exceeds one of the user-specified levels, the algorithm inserts a certain number of grid points in the base grid intervals $[x_{i-1}, x_i]$ and $[x_i, x_{i+1}]$ (the number of new grid points depending on the magnitude

of $\|u_{xx}(x_i)\|$). As another tuning parameter, the influence of the base grid point x_i can be extended beyond x_{i-1} and/or x_{i+1} by specifying the sizes of buffer zones on both sides of the base grid point. As the grid is updated at discrete time levels only, a buffer zone in the flow direction allows a correct description of the moving front during a certain time interval (i.e., a finely gridded buffer zone compensates for the inaccurate determination of the front location).

The PDEs are discretized using cubic spline differentiators and the resulting system of semi-discrete equations is integrated in time using the implicit RK solver RADAU5 [10] with error tolerances set to atol = rtol = 10^{-5}. The time integration is halted every $N_{\text{adapt}} = 5$ integration steps in order to insert or remove grid points. The initial values on these new grid points are interpolated using cubic splines (to provide initial conditions for the new ODEs). Numerical results are depicted in Figure 1.5 which shows the evolution of the spatial profile at $t = 0, 0.2, 0.4, \ldots, 1$. The bottom of the figure indicates the location of the grid points, i.e., the fixed $N_b = 51$ base grid points and the finer grids following the moving front.

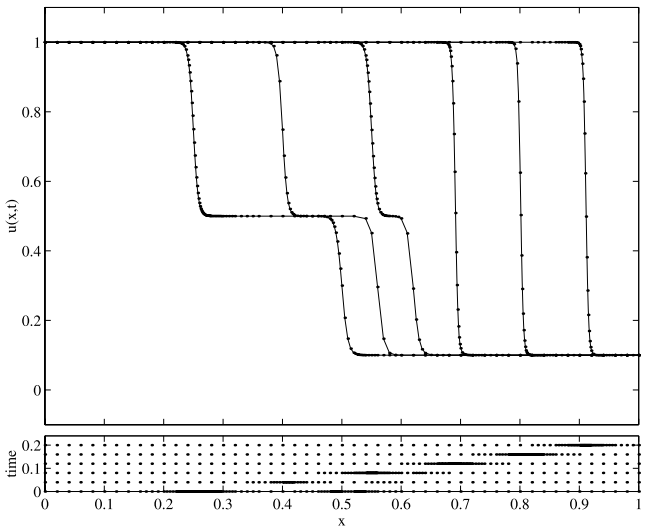


FIGURE 1.5
Burgers' equation: adaptive grid solution using ANUGB on a base grid with 51 nodes (dots) and exact solution (solid line) every 0.2 units in time.

This method is very intuitive and works quite well. However, one of the main drawbacks is that a relatively large number of base grid points is needed to sense the solution curvature. Also, in contrast with the apparent simplicity of the algorithm, the user has the requirement to adjust several parameters, e.g., threshold values, number of points in the finer grids, extension of the buffer zones, so that tuning might become quite involved. Additional applications of this method are presented by the authors in [31].

1.4.2 Case Study 2

Consider a bio-engineering application [28], e.g., a fixed-bed bioreactor in which biomass growth and death processes take place

$$\nu_1 S \xrightarrow{\varphi_1} X + \nu_2 P \tag{1.33}$$

$$X \xrightarrow{\varphi_2} X_d \tag{1.34}$$

The biomass (micro-organisms) X grows on the fixed bed due to the substrate S dissolved in the flowing medium. In addition to biomass, a product of interest P is also obtained. Simultaneously, a part X_d of the biomass dies.

It is assumed that the substrate diffusion is negligible, so that the following mass balance PDEs can be written

$$S_t = -vS_x - \nu_1 \frac{1-\epsilon}{\epsilon}\varphi_1, \quad 0 < x < L, \quad t > 0 \tag{1.35}$$

$$X_t = \varphi_1 - \varphi_2 \tag{1.36}$$

where the same notation is used for the component concentrations.

In (1.35), $v = F/(\epsilon A)$ is the superficial velocity of the fluid flowing through the bed, ϵ is the total void fraction (or bed porosity), and φ_1 is the growth rate given by a model of Contois

$$\varphi_1 = \mu_{\max} \frac{S}{k_c X + S} X \tag{1.37}$$

where $\mu_{\max}$ is the maximum specific growth rate and k_c is the saturation coefficient. In (1.34), φ_2 is the death rate given by a simple linear law

$$\varphi_2 = k_d X \, . \tag{1.38}$$

The model PDEs (1.35) and (1.36) are supplemented by a Dirichlet boundary condition in $x = 0$ and initial conditions.

The numerical values of the model parameters are: $L = 1\ m$, $A = 0.04\ m^2$, $\epsilon = 0.5$, $F = 2\ l/h$, $\nu_1 = 0.4$, $\mu_{\max} = 0.35\ h^{-1}$, $k_c = 0.4$, $k_d = 0.05\ h^{-1}$.

This application is analyzed with the local refinement code PARAB, which has been developed by the authors and implements a method originally proposed by Eigenberger and Butt [5]. The grid placement criterion is based on an error estimation procedure, i.e., the solution is represented by piecewise second-order parabolas and the interpolation error between two subsequent parabolas (defined on x_{i-2}, x_{i-1}, x_i and x_{i-1}, x_i, x_{i+1}) in the middle of each grid interval is evaluated. If this error is above a specified maximal error level $E_{\max}$, an additional node is inserted. Conversely, if two subsequent errors are below another specified minimal error level $E_{\min}$, the node in between is removed. This basic algorithm has been modified so that, if the solution displays a relatively flat profile at some time, not all the nodes are removed from the grid (here, there is no fixed base grid so that all the nodes could be deleted one after another if the interpolation error is small) and the method is able to cope with

the sudden appearance of new steep profiles, i.e., a maximal grid spacing $\Delta x_{\max}$ is specified.

Spatial approximation is accomplished using 5-point biased upwind finite differences. The tuning parameters of PARAB are selected as follows: $E_{\min} = 10^{-5}$, $E_{\max} = 10^{-4}$, $\Delta x_{\max} = 0.2$. Error tolerances atol = rtol = 10^{-4} are imposed for the time integration with RADAU5. The solver is halted every $N_{\text{adapt}} = 3$ integration steps for updating the grid. Data are transferred from the old to the new grid using cubic spline interpolation. During the course of the computation, the number of grid points varies between 223 and 59. The grid refinement algorithm is easy to tune and performs quite well. Figures 1.6 and 1.7 show the evolution of the substrate and biomass concentrations every 5000 sec following a step-change in the biomass concentration at the reactor inlet from 5 g/l to 8 g/l. A steep front of substrate concentration propagates towards the bioreactor outlet.

1.4.3 Case Study 3

We now turn to a prototype model for oil reservoir simulation taken from [19], i.e., the convection-diffusion Buckley-Leverett equation

$$u_t + f(u)_x = \varepsilon(v(u)u_x)_x, \quad 0 < x < 1, \quad t > 0\,. \tag{1.39}$$

In this expression, the diffusion coefficient $v(u)$ vanishes for $u = 0$ and 1,

$$v(u) = 4u(1-u) \tag{1.40}$$

so that (1.39) is a degenerate parabolic equation. The flux function including gravitational effects is given by

$$f(u) = \frac{u^2}{u^2 + (1-u)^2}\left(1 - 5(1-u)^2\right)\,. \tag{1.41}$$

The PDE (1.39) is supplemented by Riemann initial conditions

$$\begin{aligned} u(x,0) &= 0, \quad 0 \le x \le 1 - 1/\sqrt{2} \\ &= 1, \quad 1 - 1/\sqrt{2} \le x \le 1 \end{aligned} \tag{1.42}$$

and Dirichlet boundary conditions in $x_L = 0$ and $x_R = 1$

$$\begin{aligned} u(0,t) &= 0 \\ u(1,t) &= 1\,. \end{aligned} \tag{1.43}$$

This difficult problem is solved for $\varepsilon = 0.01$ using the moving grid code AGE [26], which is based on a method published by Sanz-Serna and Christie [25] and an extension proposed by Revilla [24].

AGE is a static gridding algorithm that equidistributes a functional $m(u)$ based on the solution curvature, i.e.,

$$\int_{x_{i-1}}^{x_i} m(u)dx = \int_{x_i}^{x_{i+1}} m(u)dx, \qquad 2 \le i \le N-1 \tag{1.44}$$

$$m(u) = \sqrt{(\alpha + \|u_{xx}\|_\infty)}\,. \tag{1.45}$$

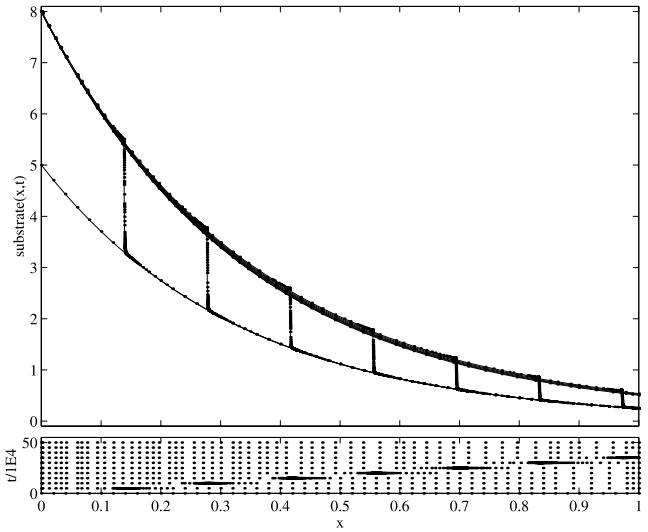


FIGURE 1.6
Substrate concentration in a fixed-bed bioreactor; adaptive grid solution using PARAB every 5000 units in time.

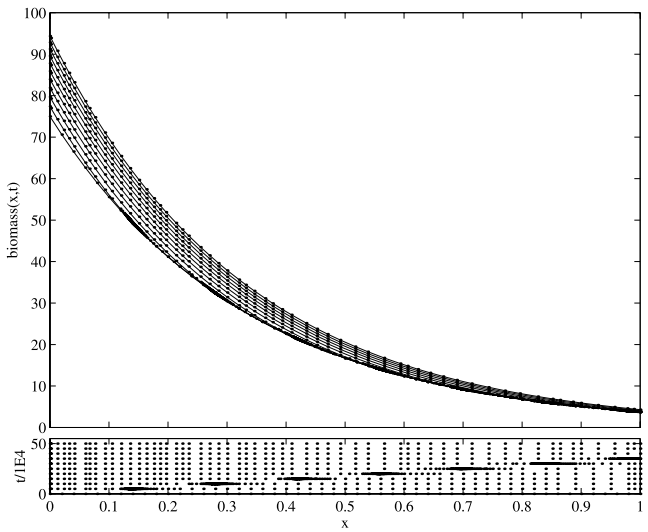


FIGURE 1.7
Biomass concentration in a fixed-bed bioreactor; adaptive grid solution using PARAB every 5000 units in time.

The scaling factor α can be used to modify the relative importance of values of x and values of u. An additional parameter β is introduced to avoid the excessive clustering of nodes in regions where $\|u_{xx}\|_\infty$ is large, i.e., the values of the second-order derivative that exceed β are reduced to the value β.

A superbee flux limiter is used for computing $f(u)_x$, whereas a 5-point centered finite difference scheme is applied to the diffusion term. The semi-discrete ODEs are solved using RADAU5 with error tolerances set to atol = rtol = 10^{-4}. Time integration and grid adaptation proceed alternately ($N_{\text{adapt}} = 1$). The tuning parameters of the adaptive grid algorithm are set as: $N = 101$, $\alpha = 10^{-3}$, and $\beta = 10^3$. Excellent numerical solutions are obtained, which are graphed in Figure 1.8 (also the figure on the book cover) every 0.4 units in time. Additional applications of this method are reported by the authors in [26, 27].

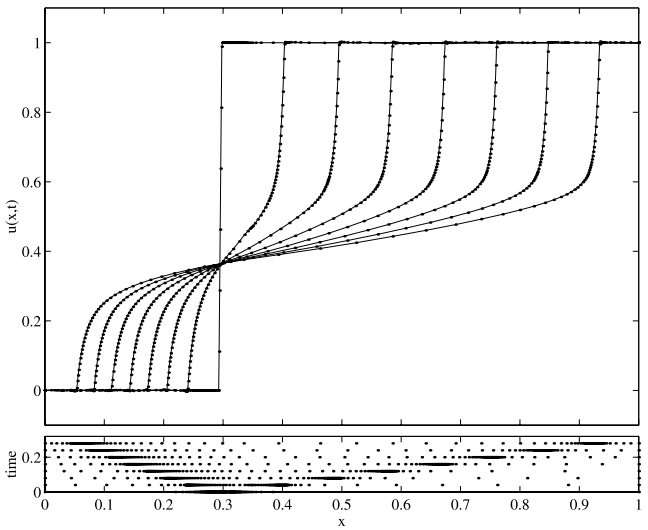


FIGURE 1.8
Buckley-Leverett equation: numerical solution on an adaptive grid with $N = 101$ nodes (AGE) at $t = 0, 0.4, \ldots, 2.8$.

1.4.4 Case Study 4

Consider a laboratory-scale catalytic fixed-bed reactor in which the hydrogenation of small amounts of CO_2 to methane is accomplished

$$CO_2 + 4H_2 \rightarrow CH_4 + 2H_2O \ . \tag{1.46}$$

Based on experimental studies, a model of the transient behavior of this plant has been derived in [30], and investigated numerically in [5]. The model consists of two

PDEs for the mass balance of CO_2 and the energy balance, respectively,

$$c_t = -vc_x + Dc_{xx} - r, \quad 0 < x < L, \quad t > 0 \tag{1.47}$$

$$T_t = -\epsilon v \frac{\rho_g c_{pg}}{\rho c_p} T_x + \frac{\lambda}{\rho c_p} T_{xx} + \frac{2k_w}{r\rho c_p}(T_w - T) + \frac{(-\Delta H)}{\rho c_p} r \tag{1.48}$$

with the reaction rate

$$r = k_r \frac{ce^{-E/RT}}{1 + k_c c} \tag{1.49}$$

and the boundary and initial conditions

$$c_x(0,t) = \frac{v}{D}(c - c_{in}), \quad T_x(0,t) = \frac{\epsilon v \rho_g c_{pg}}{\lambda}(T - T_{in}) \tag{1.50}$$

$$c_x(L,t) = 0, \quad T_x(L,t) = 0 \tag{1.51}$$

$$c(x,0) = c_0(x), \quad T(x,0) = T_0(x)\,. \tag{1.52}$$

The numerical values of the model parameters are: $L = 0.2\ m$, $r = 0.01\ m$, $\epsilon = 0.6$, $v = 1.5\ m/s$, $D = 5 \times 10^{-4}\ m^2/s$, $\rho c_p = 364\ kcal/m^3K$, $c_{pg} = 2.29\ kcal/kgK$, $\rho_g = 0.0775\ kg/m^3$, $\lambda = 3.5 \times 10^{-4}\ kcal/msK$, $k_w = 5 \times 10^{-4}\ kcal/m^2sK$, $T_w = 300\ K$, $-\Delta H = 6.006\ kcal$, $k_r = 0.971 \times 10^{13} m^{-3}s^{-1}$, $k_c = 12.7$, $E = 25.211\ kcal/mole$, $R = 1.98\ cal/mole - K$.

The concentration and temperature transients at reactor start-up, e.g., due to step changes in the feed concentration $c_{in}(t) = 0 \rightarrow 2.5\ mole\%$ and temperature $T_{in}(t) = 300 \rightarrow 500\ K$, are studied numerically using the dynamic gridding software package MOVGRD (ACM 731) [1, 33]. MOVGRD is based on a nonlinear Galerkin discretization of the Lagrangian description of the PDEs (1.29) and a smoothed equidistribution principle using regularization techniques reported by Dorfy and Drury [3].

To introduce this method, consider the spatial equidistribution equation (1.25) expressed in terms of the grid density $n_i = 1/\Delta x_i$

$$\frac{n_{i-1}}{M_{i-1}} = \frac{n_i}{M_i}, \qquad 2 \le i \le N-1 \tag{1.53}$$

where M_i is a discrete approximation of an arc-length monitor function (1.26).

In order to avoid excessive spatial distortion and temporal oscillation of the grid, two regularization procedures are used.

First, spatial smoothing is accomplished by replacing the grid density n_i in (1.53) by

$$\begin{aligned} \tilde{n}_0 &= n_0 - \kappa(\kappa+1)(n_1 - n_0) \\ \tilde{n}_i &= n_i - \kappa(\kappa+1)(n_{i+1} - 2n_i + n_{i-1}) \qquad 2 \le i \le N-1 \\ \tilde{n}_N &= n_N - \kappa(\kappa+1)(n_{N-1} - n_N) \end{aligned} \tag{1.54}$$

where κ is a positive parameter. The introduction of the "anti-diffused" density $\tilde{n}_i$ ensures that the grid is locally bounded, i.e., that adjacent grid spacings do not differ substantially from one another (the complete developments can be found in [33])

$$\frac{\kappa}{\kappa+1} \leq \frac{n_{i-1}}{n_i} \leq \frac{\kappa+1}{\kappa} \tag{1.55}$$

Second, temporal smoothing is accomplished by replacing the system of algebraic equations (1.53) by a system of differential equations

$$\frac{\tilde{n}_{i-1} + \tau\, \dot{\tilde{n}}_{i-1}}{M_{i-1}} = \frac{\tilde{n}_i + \tau\, \dot{\tilde{n}}_i}{M_i}, \qquad 2 \leq i \leq N-1 \tag{1.56}$$

where the positive parameter τ acts as a time-constant preventing abrupt changes in the grid movement. Experience shows that spatial smoothing is more important than temporal smoothing. The semi-discrete approximation of the Lagrangian form of the PDEs is combined with Equation (1.56) to yield a system of ODEs that is integrated using the BDF solver DASSL [22].

Figures 1.9 and 1.10 show the evolution of the concentration and temperature profiles every 100 s. Reaction takes place in the middle of the tubular reactor, resulting in the formation of a "hot spot." These numerical results have been obtained with the parameter values: $N = 51$, $\alpha = 10^{-4}$, $\kappa = 2$, and $\tau = 10^{-3}$. Tolerances atol = rtol = 10^{-7} are imposed for the time integration with DASSL.

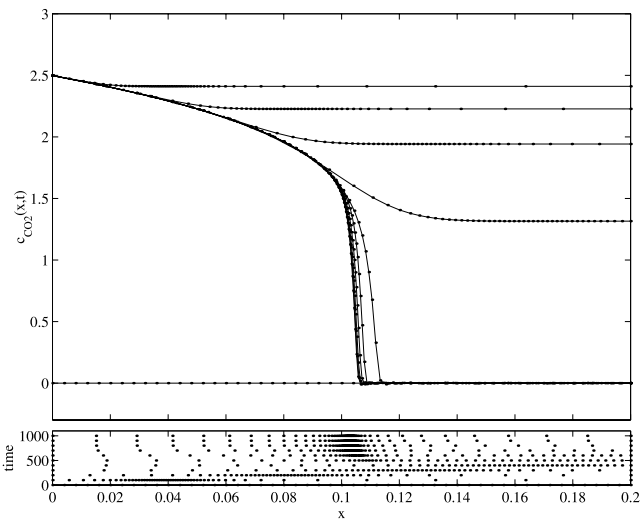


FIGURE 1.9
Fixed-bed methanator: concentration profiles on an adaptive grid with $N = 51$ nodes (MOVGRD) at $t = 0, 100, \ldots, 1000$.

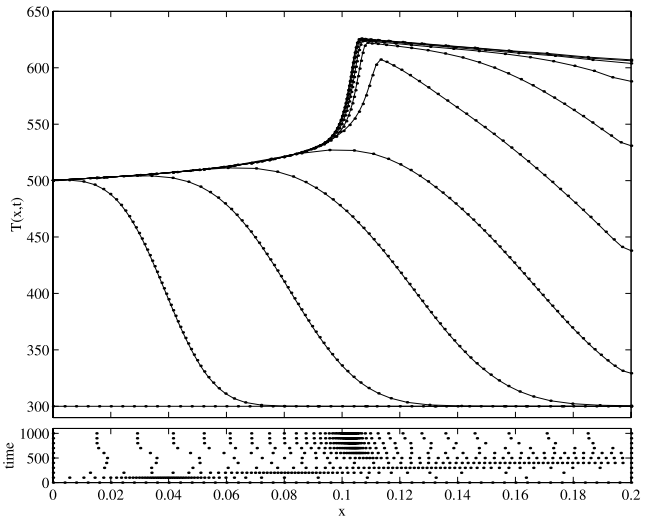


FIGURE 1.10
Fixed-bed methanator: temperature profiles on an adaptive grid with $N = 51$ nodes (MOVGRD) at $t = 0, 100, \ldots, 1000$.

1.4.5 Case Study 5

Consider a model of flame propagation [4] consisting of two coupled equations for mass density and temperature

$$\begin{aligned} \rho_t &= \rho_{xx} - N_{DA}\rho \\ T_t &= T_{xx} + N_{DA}\rho, \qquad 0 < x < 1, t > 0 \end{aligned} \tag{1.57}$$

where $N_{DA} = 3.52 \times 10^6\, e^{-4/T}$.

The initial conditions are given by

$$\rho(x, 0) = 1, \qquad T(x, 0) = 0.2, \qquad 0 \le x \le 1 \tag{1.58}$$

and the boundary conditions are

$$\begin{aligned} \rho_x(0, t) &= 0, \qquad T_x(0, t) = 0\,, \\ \rho_x(1, t) &= 0, \qquad T(1, t) = f(t), \qquad t \ge 0\,, \end{aligned} \tag{1.59}$$

with

$$\begin{aligned} f(t) &= 0.2 + t/2 \times 10^{-4}, \qquad t \le 2 \times 10^{-4} \\ &= 1.2 \qquad\qquad\qquad\quad t \ge 2 \times 10^{-4}\,. \end{aligned} \tag{1.60}$$

The heat source located at $x = 1$ generates a flame front that propagates from right to left at an almost constant speed.

This problem is solved on the time interval $(0, 0.006)$ using the moving grid software MOVCOL [17]. In MOVCOL, the node movement is governed by a continuous moving mesh equation (or moving mesh PDE - MMPDE) instead of a discrete moving mesh equation as in MOVGRD [i.e., the system of ODEs (1.56)]. MMPDEs can be derived in several ways, as reviewed in [16]. Here, we briefly sketch the derivation steps of a basic MMPDE based on the equidistribution principle.

A one-to-one transformation between physical (x) and computational (ξ) coordinates is introduced

$$\begin{aligned} x &= x(\xi, t), \quad \xi \in [0, 1] \\ x(0, t) &= x_L, \quad x(1, t) = x_R \end{aligned} \tag{1.61}$$

by equidistributing a monitor function $m(x, t)$, i.e.,

$$\int_{x_L}^{x(\xi,t)} m(z, t)\, dz = \xi \int_{x_L}^{x_R} m(z, t)\, dz\,. \tag{1.62}$$

Differentiating (1.62) with respect to ξ once and twice yields two differential forms of the equidistribution principle

$$m(x(\xi, t), t) \frac{\partial x(\xi, t)}{\partial \xi} = \int_{x_L}^{x_R} m(z, t)\, dz\,. \tag{1.63}$$

$$\frac{\partial}{\partial \xi} \left\{ m(x(\xi, t), t) \frac{\partial x(\xi, t)}{\partial \xi} \right\} = 0 \tag{1.64}$$

which are called quasi-static equidistribution principles since they do not contain the node speed $\dot{x}\ (\xi, t)$.

To derive an MMPDE, we require that the mesh satisfies the latter equation at the time $t + \tau$ instead of at t, so that

$$\frac{\partial}{\partial \xi} \left\{ m(x(\xi, t + \tau), t + \tau) \frac{\partial x(\xi, t + \tau)}{\partial \xi} \right\} = 0 \tag{1.65}$$

gives a relaxation time τ for the mesh to satisfy the equidistribution principle.

Using the Taylor series expansions

$$\begin{aligned} m(x(\xi, t + \tau), t + \tau) &= m(x(\xi, t), t) + \tau\ \dot{x}\ (\xi, t) \frac{\partial m(x(\xi, t), t)}{\partial x} \\ &\quad + \tau \frac{\partial m(x(\xi, t), t)}{\partial t} + O(\tau^2) \\ \frac{\partial x(\xi, t + \tau)}{\partial \xi} &= \frac{\partial x(\xi, t)}{\partial \xi} + \tau \frac{\partial\ \dot{x}\ (\xi, t)}{\partial \xi} + O(\tau^2) \end{aligned} \tag{1.66}$$

an MMPDE is obtained as

$$\frac{\partial}{\partial \xi} \left\{ m \frac{\partial\ \dot{x}}{\partial \xi} \right\} + \frac{\partial}{\partial \xi} \left\{ \frac{\partial m}{\partial \xi}\ \dot{x} \right\} = -\frac{\partial}{\partial \xi} \left\{ \frac{\partial m}{\partial t} \frac{\partial x}{\partial \xi} \right\} - \frac{1}{\tau} \frac{\partial}{\partial \xi} \left\{ m \frac{\partial x}{\partial \xi} \right\} \tag{1.67}$$

that can be further simplified by dropping $\frac{\partial}{\partial \xi}\left\{\frac{\partial m}{\partial t}\frac{\partial x}{\partial \xi}\right\}$ as well as $\frac{\partial}{\partial \xi}\left\{\frac{\partial m}{\partial \xi}\dot{x}\right\}$, i.e.,

$$\frac{\partial}{\partial \xi}\left\{m\frac{\partial \dot{x}}{\partial \xi}\right\} = -\frac{1}{\tau}\frac{\partial}{\partial \xi}\left\{m\frac{\partial x}{\partial \xi}\right\} . \tag{1.68}$$

This simplification is justified by the fact that the term involving $\frac{\partial m}{\partial t}$ is often difficult to compute and is not absolutely necessary since the term $-\frac{1}{\tau}\frac{\partial}{\partial \xi}\left\{m\frac{\partial x}{\partial \xi}\right\}$ is a source of node movement which measures how closely the mesh satisfies the equidistribution principle, even when $m(x,t)$ is independent of t.

Of course, temporal mesh smoothing is automatically built into the MMPDE. However, for PDE problems involving large solution variations, the monitor function $m(x,t)$ is generally fairly nonsmooth in space, and spatial smoothing must be introduced [18]. To this end, a potential approach is to replace the monitor function m by a "smoothed" M, which satisfies a PDE in ξ and t involving an artificial diffusion term

$$M - \frac{1}{\lambda^2}\frac{\partial^2 M}{\partial \xi^2} = m \tag{1.69}$$

with boundary conditions

$$\frac{\partial M}{\partial \xi}(0,t) = \frac{\partial M}{\partial \xi}(1,t) = 0 . \tag{1.70}$$

However, this approach is not suitable from a computational point of view and alternatives are developed in [18]. Here, we just mention that MOVCOL is based on the following smoothed MMPDE:

$$\frac{\partial}{\partial \xi}\left\{\frac{1}{m}\left(1 - \frac{1}{\lambda^2}\frac{\partial^2}{\partial \xi^2}\right)\left(\tau\,\dot{n} + n\right)\right\} = 0 \tag{1.71}$$

for the mesh concentration function $n = 1/(\partial x/\partial \xi)$. The diffusion parameter λ is of the form $(N-1)/\sqrt{\gamma(\gamma+1)}$, where γ is the user defined, spatial smoothing parameter.

MOVCOL uses cubic Hermite collocation for discretization of the physical PDEs in divergence form, and a three-point finite difference discretization of the MMPDE. The resulting semi-discrete ODE system is solved using the time integrator DASSL [22].

The evolution of the temperature and density profiles are graphed every 0.0006 in Figures 1.11 and 1.12. These numerical results have been obtained with $N = 21$, $\gamma = 1$, $\tau = 10^{-4}$, and atol = rtol = 10^{-4}.

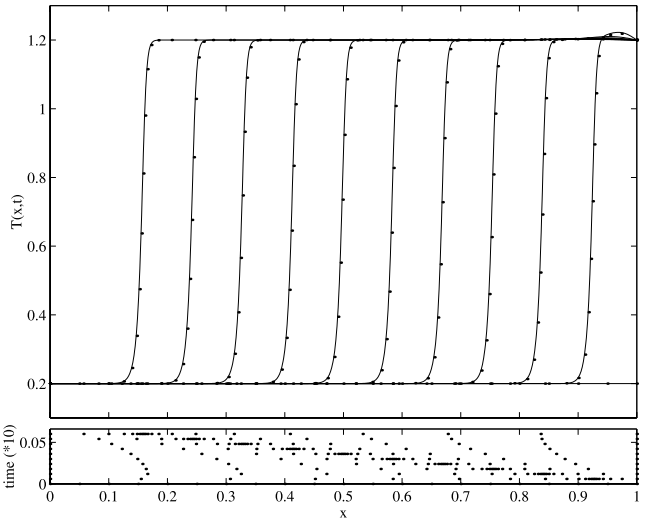


FIGURE 1.11
Flame propagation: temperature profiles on an adaptive grid with $N = 21$ nodes (MOVCOL) at $t = 0, 0.0006, \ldots, 0.006$.

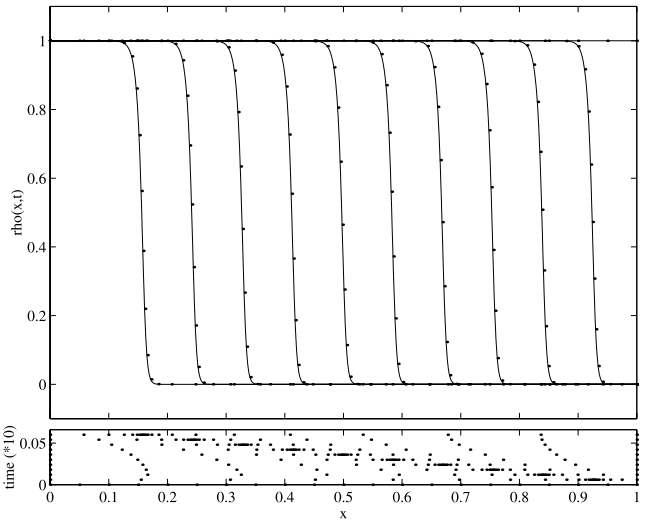


FIGURE 1.12
Flame propagation: density profiles on an adaptive grid with $N = 21$ nodes (MOVCOL) at $t = 0, 0.0006, \ldots, 0.006$.

1.4.6 Case Study 6

Consider a problem taken from [9] describing two countercurrent reactive square waves

$$\begin{aligned} v_t &= -0.5v_x - vw \\ w_t &= 0.5w_x - vw \ . \end{aligned} \tag{1.72}$$

The initial conditions are

$$\begin{aligned} v(x,0) &= 1, \quad 0 < x < 20 \\ &= 0, \quad \text{otherwise} \\ w(x,0) &= 1, \quad 80 < x < 100 \\ &= 0, \quad \text{otherwise} \ . \end{aligned} \tag{1.73}$$

The boundary conditions are

$$\begin{aligned} v(0,t) &= v(100,t) = 0 \\ w(0,t) &= w(100,t) = 0 \ . \end{aligned} \tag{1.74}$$

This problem is solved on the time interval (0, 140) using the moving finite element (MFE) method proposed by Miller and co-workers [9, 20, 21]. In this method, the solution to (1.17) is approximated using a finite element formulation with piecewise linear basis functions α_j

$$u(x,t) \approx U(x,t) = \sum_{j=1}^{N} U_j(t)\, \alpha_j(x, X(t)) \ , \tag{1.75}$$

in which both the nodal amplitudes $U_j(t)$, $j = 1, \ldots, N$, and the nodal positions $x_L = X_1(t) < X_2(t) < \cdots < X_N(t) = x_R$ are unknown functions of time.

Partial differentiation of (1.75) with respect to time yields

$$\partial U(x,y)/\partial t = \sum_{j=1}^{N} \dot{U}_j(t)\, \alpha_j(x, X(t)) + \dot{X}_j(t)\, \beta_j(x, X(t)) \ , \tag{1.76}$$

in which $\beta_j = -(\partial U/\partial x)\, \alpha_j$ can be considered as a second type of basis function.

The $2N$ unknown functions $U_j(t)$ and $X_j(t)$ are determined by minimizing the L_2 norm of the PDE residual $\|R(U)\|_2^2 = \|\partial U/\partial t - L(U)\|_2^2$ with respect to $\dot{U}_j(t)$ and $\dot{X}_j(t)$, which results in a system of $2N$ ODEs

$$\sum_{j=1}^{N} \left\langle \alpha_i, \alpha_j \right\rangle \dot{U}_j + \left\langle \alpha_i, \beta_j \right\rangle \dot{X}_j = \langle \alpha_i, L(U) \rangle \ , \tag{1.77}$$

$$\sum_{j=1}^{N} \left\langle \beta_i, \alpha_j \right\rangle \dot{U}_j + \left\langle \beta_i, \beta_j \right\rangle \dot{X}_j = \langle \beta_i, L(U) \rangle \ , \qquad i = 1, \ldots, N \ . \tag{1.78}$$

By reordering the unknown variables in a column vector $Y = (U_1, X_1, U_2, X_2, \ldots, U_N, X_N)^T$, it is possible to write (1.77) and (1.78) in a compact form

$$A(Y)\dot{Y} = g(Y) \,, \tag{1.79}$$

where $A(Y)$ is an $N \times N$ block-tridiagonal matrix (each block is a 2×2 matrix consisting of inner products of the basis functions α_j and β_j).

Integrating (1.79) in time can become problematic for two reasons. First, the mass matrix $A(Y)$ becomes ill-conditioned when some nodes drift very close together and extremely nonuniform grids are generated. Second, the mass matrix $A(Y)$ becomes singular when parallelism occurs, i.e., when at a particular node X_j the solution curvature vanishes. In this case, the MFE method intrinsically fails to determine the direction in which the node X_j should be moved.

To avoid these problem degeneracies, Miller [20, 21] introduced regularization terms in the residual minimization, which penalize the relative motions between nodes. The new minimization problem can be written as follows:

$$\dot{U}_j, \dot{X}_j \min \|\partial U/\partial t - L(U)\|_2^2 + \sum_{j=2}^{N} \left(\varepsilon_j \Delta \dot{X}_j - S_j\right) \,. \tag{1.80}$$

While the ODEs (1.77) remain unchanged, the ODEs (1.78), which govern the node motion, become

$$\begin{aligned} \sum_{j=1}^{N} \left\langle \beta_i, \alpha_j \right\rangle \dot{U}_j + \left\langle \beta_i, \beta_j \right\rangle \dot{X}_j + \varepsilon_i^2 \Delta \dot{X}_i - \varepsilon_{i+1}^2 \Delta \dot{X}_{i+1} \\ = \langle \beta_i, L(U) \rangle + \varepsilon_i S_i - \varepsilon_{i+1} S_{i+1} \,. \end{aligned} \tag{1.81}$$

The left-hand side terms, which involve the internodal viscosities ε_j, regularize the dynamic internodal node movements and keep the resulting mass-matrix positive definite, while the right-hand side terms, which contain the internodal spring functions S_j, allow a regularization of the long-term or equilibrium system.

According to a simplified algorithm formulation given in [9], the regularizing functions are given by

$$S_j = \frac{c_1}{\Delta X_j - \delta} \,, \tag{1.82}$$

$$\varepsilon_j = \frac{c_2}{\Delta X_j - \delta} \,, \tag{1.83}$$

in which c_1, c_2, and δ are tuning parameters. In particular, δ can be interpreted as a minimum permissible internodal separation. Both the viscosities and the internodal spring functions become infinite as the internodal separation approaches δ.

The evolution of the two square waves at $t = 0, 80, 140$ is graphed in Figures 1.13, 1.14, and 1.15, respectively. These results have been obtained with 32 elements ($N = 33$ nodes) and the tuning parameters $c_1 = 10^{-4}$, $c_2 = 10^{-4}$, and $\delta = 10^{-4}$.

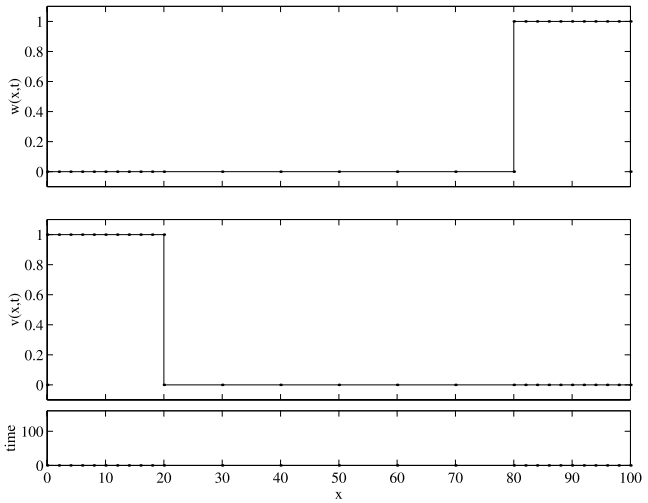


FIGURE 1.13
Two countercurrent reactive square waves: initial condition (MFE with $N = 29$).

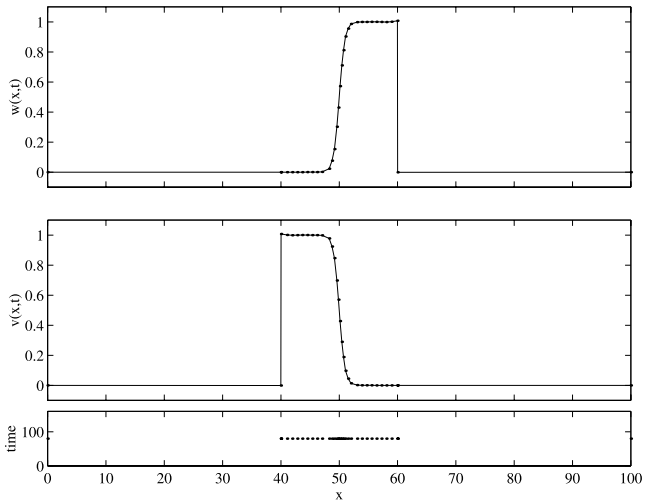


FIGURE 1.14
Two countercurrent reactive square waves: interaction at $t = 80$ (MFE with $N = 29$).

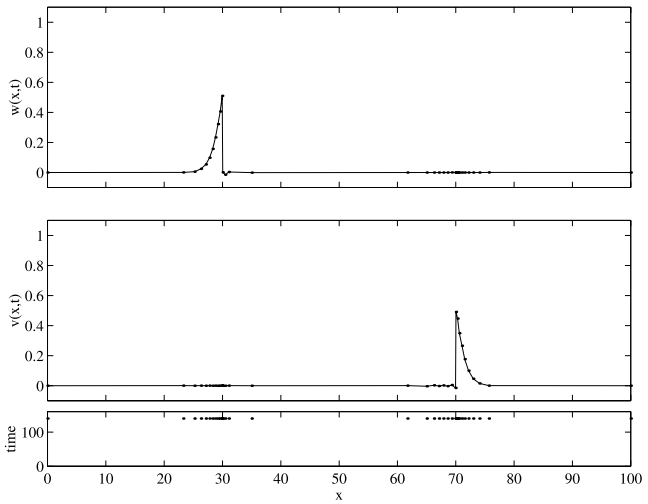


FIGURE 1.15
Two countercurrent reactive square waves: propagation at $t = 140$ (MFE with $N = 29$).

Time integration is performed with the BDF solver LSODI [13] with error tolerances atol $= 10^{-2}$ and rtol $= 10^{-4}$.

MFE has attracted considerable attention and, over the years, several improvements have been proposed, including matrix preconditioning [34, 35] and gradient weighting [2]. A review of some of these results and additional applications of the method are presented in [32].

1.5 Summary

We have briefly reviewed the basic computational methods for adaptive MOL and illustrated their use through a series of example applications. This survey illustrates the effectiveness of adaptive methods in resolving the sharp spatial and temporal features of PDE solutions that would be difficult to resolve with fixed grid methods. This discussion is also intended to highlight the computer codes that are readily available for adaptive methods.

However, the adaptive approach generally involves additional complexity compared to a fixed-grid formulation, including the tuning of method parameters to achieve the desired solution resolution and accuracy. Thus, some trial and error is inevitable in using adaptive MOL, and the expectation is that this additional effort is worthwhile; experience has demonstrated that generally this is the case.

Now we proceed in the remainder of this book to discussions by developers of adaptive methods who elucidate the features of a spectrum of methods. The intention is to facilitate the use of adaptive methods by highlighting the features of the methods and associated codes, and by demonstrating the characteristics and effectiveness of the adaptive approach to PDE solutions through example applications.

References

[1] J.G. Blom and P.A. Zegeling, Algorithm 731: a moving-grid interface for systems of one-dimensional time-dependent partial differential equations, *ACM Trans. Math. Software,* **20**, (1994), 194–214.

[2] N. Carlson and K. Miller, Design and application of a gradient-weighted moving finite element code, Part I, in 1-D, *SIAM J. Sci. Comput.,* **19**, (1998), 728–765.

[3] E.A. Dorfi and L.O'C. Drury, Simple adaptive grids for 1-D initial value problems, *J. Comp. Phys.,* **69**, (1987), 175–195.

[4] H.A. Dwyer and B.R. Sanders, Numerical modeling of unsteady flame propagation, *Sandia National Lab. Livermore Report SAND77-8275,* 1978.

[5] G. Eigenberger and J.B. Butt, A modified Crank-Nicolson technique with non-equidistant space steps, *Chem. Eng. Sci.,* **31**, (1976), 681–691.

[6] P.R. Eiseman, Adaptive grid generation, *Comput. Meth. Appl. Mech. Eng.,* **64**, (1987), 321–376.

[7] B. Fornberg, Generation of finite difference formulas on arbitrarily spaced grid, *Math. Comp.,* **51**, (1988), 699–706.

[8] R.M. Furzeland, J.G. Verwer, and P.A. Zegeling, A numerical study of three-moving grid methods for one-dimensional partial differential equations which are based on the method of lines, *J. Comp. Phys.,* **89**, (1990), 349–388.

[9] R. Gelinas, S. Doss, and K. Miller, The moving finite element method: applications to general equations with multiple large gradients, *J. Comp. Phys.,* **40**, (1981), 202–249.

[10] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II - Stiff and Differential-Algebraic Problems,* Springer-Verlag, Berlin, 1991.

[11] A. Harten, High resolution schemes for hyperbolic conservation laws, *J. Comp. Phys.,* **49**, (1983), 357–393.

[12] D.F. Hawken, J.J. Gottlieb, and J.S. Hansen, Review of some adaptive node-movement techniques in finite element and finite difference solutions of partial differential equations, *J. Comp. Phys.,* **95**, (1991), 254–302.

[13] A.C. Hindmarsh, ODEPACK: A systematized collection of ODE solvers, in R.S. Stepleman, ed., *Scientific Computing,* IMACS, North Holland, 1983, 55–64.

[14] J.M. Hyman, Adaptive moving mesh methods for partial differential equations, in *Advances in Reactor Computations,* American Nuclear Society Press, La Grange Park, IL, 1983, 24–43.

[15] S.S. Hu and W.E. Schiesser, An adaptive grid method in the numerical method of lines, in R. Vichnevetsky and R.S. Stepleman, eds., *Advances in Computer Methods for Partial Differential Equations,* IMACS, North Holland (1981), 305–311.

[16] W. Huang, Y. Ren, and R.D. Russell, Moving mesh partial differential equations (MMPDEs) based on the equidistribution principle, *SIAM J. Numer. Anal.,* **31**, (1994), 709–730.

[17] W. Huang and R.D. Russell, A moving collocation method for solving time dependent partial differential equations, *Appl. Num. Math.,* **20**, (1996), 101.

[18] W. Huang and R.D. Russell, Analysis of moving mesh PDEs with spatial smoothing, *SIAM J. Numer. Anal.,* **34**, (1997), 1106.

[19] A. Kurganov and E. Tadmor, New high-resolution central schemes for nonlinear conservation laws and convection-diffusion equations, *UCLA Computational and Applied Mathematics Report,* April 1999.

[20] K. Miller and R. Miller, Moving finite elements, Part I, *SIAM J. Numer.,* **18**, (1981), 1019–1032.

[21] K. Miller, Moving finite elements, Part II, *SIAM J. Numer.,* **18**, (1981), 1033–1057.

[22] L.R. Petzold, A description of DASSL: a differential/algebraic system solver, in R.S. Stepleman, ed., *Scientific Computing,* IMACS, North-Holland (1983), 65–68.

[23] L.R. Petzold, Observations on an adaptive moving grid method for one-dimensional systems of partial differential equations, *Appl. Numer. Math.,* **3**, (1987), 347–360.

[24] M.A. Revilla, Simple time and space adaptation in one-dimensional evolutionary partial differential equation, *Int. J. Numer. Methods Eng.,* **23**, (1986), 2263–2275.

[25] J.M. Sanz-Serna and I. Christie, A simple adaptive technique for nonlinear wave problems, *J. Comp. Phys.,* **67**, (1986), 348–360.

[26] P. Saucez, A. Vande Wouwer, and W.E. Schiesser, Some observations on a static spatial remeshing method based on equidistribution principles, *J. Comp. Phys.,* **128**, (1996), 274–288.

[27] P. Saucez, A. Vande Wouwer, and W.E. Schiesser, An adaptive method of lines solution of the Korteweg-de Vries equation, *Comp. Math. Applic.,* **35**, (1998), 13–25.

[28] N. Tali-Maamar, T. Damak, J.P. Babary, and M.T. Nihtilä, Application of a collocation method for simulation of distributed parameter bioreactors, *Math. Comp. Sim.,* **35**, (1993), 303–319.

[29] J.F. Thompson, A survey of dynamically-adaptive grids in the numerical solution of partial differential equations, *Appl. Numer. Math.,* **1**, (1985), 3–27.

[30] H. Van Doesburg and W.A. De Jong, Dynamic behavior of an adiabatic fixed-bed methanator, *Int. Symp. Chem. React. Eng., Evanston, Advances in Chem. Series,* **133**, (1974), 489–503.

[31] A. Vande Wouwer, P. Saucez, and W.E. Schiesser, Some user-oriented comparisons of adaptive grid methods for partial differential equations in one space dimension, *Appl. Numer. Math.,* **26**, (1998), 49–62.

[32] A. Vande Wouwer, P. Saucez, and W.E. Schiesser, Numerical experiments with the (gradient-weighted) finite element method, *submitted.*

[33] J.G. Verwer, J.G. Blom, R.M. Furzeland, and P.A. Zegeling, A moving-grid method for one-dimensional PDEs based on the method of lines, in J.E. Flaherty, P.J. Paslow, M.S. Shephard, and J.D. Vasilakis, eds., *Adaptive Methods for Partial Differential Equations,* SIAM, Philadelphia (1989), 160–175.

[34] A.J. Wathen and M.J. Baines, On the structure of the moving finite-element equations, *IMA J. Numer. Anal.,* **5**, (1985), 161–182.

[35] A.J. Wathen, Mesh-independent spectra in the moving finite element equations, *SIAM J. Numer.,* **23**, (1986), 797–814.

[36] A.B. White, On the numerical solution of initial/boundary-value problems in one-space dimension, *SIAM J. Numer. Anal.,* **19**, (1982), 683–697.

Chapter 2

Application of the Adaptive Method of Lines to Nonlinear Wave Propagation Problems

Alain Vande Wouwer, Philippe Saucez, and William Schiesser

2.1 Introduction

In recent years, much interest has developed in the numerical treatment of PDEs giving rise to nonlinear wave phenomena, and particularly, solitary waves. In this chapter, attention is focused on a few particular cases, i.e., the cubic Schrödinger equation (CSE) and the derivative nonlinear Schrödinger equation (DNLS), as well as several Korteweg-de Vries (KdV)-like equations in one space dimension. These equations have been used extensively to model nonlinear dispersive waves in a wide range of application areas, such as water wave models, laser optics, and plasma physics.

In order to efficiently compute numerical solutions of these equations, it is appealing to resort to an adaptive grid technique that automatically concentrates the spatial nodes in the regions of rapid solution variations (i.e., the wave moving fronts). In this connection, an adaptive grid refinement algorithm based on the equidistribution principle and spatial regularization procedures is implemented and applied to several illustrative problems, including the propagation of a single soliton, the interaction between two solitons, and the bound state of several solitons, and the propagation of a compacton (a soliton with compact support).

Some implementation details are given and the performance of the method is discussed in terms of accuracy and computational efficiency.

2.2 Adaptive Grid Refinement

In this section, an adaptive grid method which equidistributes a given monitor function subject to constraints on the grid regularity is presented. The time-stepping procedure as well as some implementation issues are discussed.

2.2.1 Grid Equidistribution with Constraints

As we have seen in Chapter 1, spatial equidistribution is an important principle on which many adaptive grid algorithms are built. If $m(u)$ is a specified monitor function, the spatial equidistribution equation for the grid points x_i, $i = 1, 2, \ldots, N$, can be expressed in continuous form

$$\int_{x_{i-1}}^{x_i} m(u)\,dx = \int_{x_i}^{x_{i+1}} m(u)\,dx = c, \qquad 2 \le i \le N-1 \tag{2.1}$$

or in discrete form

$$M_{i-1}\Delta x_{i-1} = M_i \Delta x_i = c, \qquad 2 \le i \le N-1 \tag{2.2}$$

where $\Delta x_i = x_{i+1} - x_i$ is the local grid spacing, M_i is a discrete approximant of the monitor function $m(u)$ in the grid interval $[x_i, x_{i+1}]$, and c is a constant.

A popular monitor function is based on the arc length of the solution [21], i.e.,

$$m(u) = \sqrt{(\alpha + \|u_x\|_2^2)}\,. \tag{2.3}$$

Other monitor functions can be used as well and our experience [22] suggests the use of the following curvature-related function:

$$m(u) = \sqrt{(\alpha + \|u_{xx}\|_\infty)}\,. \tag{2.4}$$

In this expression, $\alpha > 0$ ensures that the monitor function is strictly positive and acts as a regularization parameter which forces the existence of at least a few nodes in flat (2.3) or linear (2.4) parts of the solution. Another regularization parameter $\beta > 0$ can be introduced [21] to avoid excessive clustering of nodes in regions where the solution exhibits very steep slope (2.3) or very high curvature (2.4), i.e., β is used instead of $\|u_x\|_2^2$ or $\|u_{xx}\|_\infty$ in the evaluation of the corresponding monitor function. In our experience, this latter regularization is particularly useful in the case of a curvature-based monitor function (2.4).

The accuracy of the spatial derivative approximations (e.g., using finite difference techniques) and the stiffness of the semi-discrete system of differential equations are largely influenced by the regularity and spacing of the grid points. This stresses the importance of limiting grid distortion using spatial regularization procedures. In practice, the use of parameters α and β is not sufficient to ensure, in a systematic way, the grid regularity, and in the following, a more advanced procedure due to Kautsky and Nichols [12] based on the concept of *locally bounded grid* is explored. This procedure, as we shall see, involves a *variable number of nodes.*

A grid is said to be locally bounded with respect to a constraint $K \ge 1$ if

$$\frac{1}{K} \le \frac{\Delta x_i}{\Delta x_{i-1}} \le K, \qquad 2 \le i \le N-1\,. \tag{2.5}$$

Then the equidistribution problem becomes the following.

Given a monitor function $m \in C^+$ (the set of continuous piecewise functions on $[x_L, x_R]$) and constants $c > 0$ and $K \ge 1$, find the grid that is

1. sub-equidistributing with respect to m and c on $[x_L, x_R]$, i.e., for the smallest number of nodes N such that $Nc \geq \int_{x_L}^{x_R} m dx$, we have $\int_{x_i}^{x_{i+1}} m dx \leq c$
2. locally bounded with respect to K

The idea of the solution to this problem, which is developed in [12], is to increase the given monitor function m — in a procedure that is called "padding" — in such a way that, when the padded monitor function is equidistributed, the ratio of consecutive grid steps are bounded as required. The padding is chosen so that the equidistributing grid has adjacent steps with constant ratios equal to the maximum allowed. Such a function exists and is given by the following formal results [12]:

Let λ be a given number. For any $m \in C^+$, we define a padding $P(m)$ of m by

$$P(m)(z) = \max_{x \in [x_L, x_R]} \frac{m(x)}{1 + \lambda |z - x| m(x)} \tag{2.6}$$

$P(m)$ has the properties:

1. $P(m)$ is strictly positive on $[x_L, x_R]$, except in the case $m \equiv 0$
2. $P(m) \geq m$ on $[x_L, x_R]$
3. $P(P(m)) = P(m)$ on $[x_L, x_R]$

Let $\lambda > 0$, $m \in C^+$ and a grid π be equidistributing on $[x_L, x_R]$ with respect to $P(m)$ and some $c > 0$. Then

1. the grid π is subequidistributing with respect to m and c
2. for $K = e^{\lambda c}$ we have

$$\frac{1}{K} \leq \frac{\Delta x_i}{\Delta x_{i-1}} \leq K, \qquad i = 2, \ldots, N-1 .$$

Based on these results, it is now possible to build a grid which is subequidistributing with respect to m and c and which is locally bounded with respect to K. In practice, the algorithm proceeds as follows:

1. pad the monitor function using $\lambda = (\log K)/c$
2. determine the smallest number of nodes N such that $Nc \geq \int_{x_L}^{x_R} P(m)\, dx$
3. equidistribute $P(m)$ with respect to $d = (\int_{x_L}^{x_R} P(m)\, dx)/N$

Clearly, we cannot know the constant d with respect to which the padded function $P(m)$ should be equidistributed before actually performing the padding. The procedure could therefore be iterated, padding the monitor function using $\lambda = (\log K)/d$ and so on.

As $d \leq c$, the grid is locally bounded with respect to a constant $L \leq K$, so that the number of points in the grid may be greater than required to strictly satisfy the problem constraints.

2.2.2 Time-Stepping Procedure and Implementation Details

The adaptive grid refinement is a static procedure and as such, proceeds in four separate steps:

1. approximation of the spatial derivatives on a fixed nonuniform grid
2. time integration of the resulting semi-discrete ODEs
3. adaptation/refinement of the spatial grid
4. interpolation of the solution to produce new initial conditions

In Step 1, the spatial derivatives are approximated using finite difference approximations up to any level of accuracy on a nonuniform grid as implemented in the standard Fortran subroutine WEIGHTS by Fornberg [3]. This algorithm is used for generating "direct" as well as "stagewise" schemes. In the latter case, higher-order derivatives are obtained by successive numerical differentiations of lower-order derivatives. An example of the use of these particular schemes will be given in the section on the Korteweg-de Vries equation.

In Step 2, time integration of the semi-discrete system of stiff ODEs or DAEs is accomplished using the variable step, fifth-order, implicit Runge–Kutta solver RADAU5 [7]. Time integration is halted periodically, i.e., every N_{adapt} integration steps, to adapt/refine the spatial grid ($N_{\text{adapt}} = 1$ corresponds to alternate time integration and grid adaptation).

In Step 3, the grid is updated using the algorithm described in the previous section. For this purpose, a new subroutine called AGEREG (from AGE, a routine previously developed by the authors [22], in which spatial grid regularization is now incorporated) has been implemented. The coding of this routine has been inspired largely by the code NEWMESH that Steinebach developed in his Ph.D. Thesis [24] and by the PDE software package SPRINT [1]. This algorithm is extended to problems in two space dimensions in Chapter 6.

Implementation issues involve computation of the monitor function (2.3) or (2.4) using cubic spline differentiators, padding of the monitor function in two sweeps of the grid (in the forward and backward direction), and grid equidistribution by inverse linear interpolation based on a trapezoidal rule.

Finally, in Step 4, the solution is interpolated using cubic splines in order to generate initial conditions on the new grid.

2.3 Application Examples

In the past several years, the solution of partial differential equations governing nonlinear waves in dispersive media has aroused considerable interest. As a result numerous research papers dealing with mathematical or numerical aspects of these

equations as well as several research monographs (see, e.g., [28, 13, 25]) have been published.

Particularly, the existence of *solitary waves* or *solitons* has been a subject of fascination. Solitons are solutions that possess two permanence properties: (a) they evolve without change of shape over large distances, and (b) they exhibit elastic collisions with other solitons. Solitons exist in some particular equations such as the nonlinear Schrödinger (NLS) equation and the Korteweg-de Vries (KdV) equation, which will be studied numerically in the continuation of this chapter.

The importance of solitons in today's literature is explained by the fact that very general nonlinear wave equations have particular regimes in which their long time behavior is modeled by an equation that has solitons. This modeling procedure, called the reductive perturbation method, may be compared to a linearization in which solitons play the role of exponential solutions [13].

2.3.1 The Nonlinear Schrödinger Equation

The nonlinear Schrödinger equation arises in a number of physical situations in the description of nonlinear waves (see, e.g., [28]) such as the propagation of a laser beam in a medium whose index of refraction is sensitive to the wave amplitude, the modulational instability of water waves, the propagation of heat pulses in anharmonic crystals, and the nonlinear modulation of plasma waves.

It provides a canonical description of the modulation of nearly monochromatic wavetrains propagating in a weakly nonlinear dispersive medium. When written in a reference frame moving at the group velocity of the carrying wave, it takes the simple form

$$iu_t + u_{xx} + qu\left|u^2\right| = 0, \quad -\infty \le x \le \infty, \quad t \ge 0 \tag{2.7}$$

$$u(x,0) = u_0(x) \tag{2.8}$$

where $u = u(x,t)$ is a complex-valued function defined over the whole real line and q is a real constant. In the following, only the focusing case is considered, which corresponds to $q > 0$.

In this equation, the cubic term opposes dispersion (i.e., the term in u_{xx}) and allows the existence of solutions, such as the solitons, where the competing forces of dispersion and nonlinearity balance each other exactly.

Zakharov and Shabat [29] derived analytical solutions to the initial-value problem (2.7) and (2.8) using an inverse scattering procedure [4]. This IVP possesses an infinite set of conservation laws, among which the conservation of the energy

$$E(u) = \int |u|^2 \, dx \tag{2.9}$$

and the Hamiltonian

$$H(u) = \int \left(|u_x|^2 - \frac{1}{2} q \, |u|^4 \right) dx \tag{2.10}$$

plays an important role in the analysis of the NLS equation.

In addition, a number of finite-difference and finite-element schemes have been suggested for the numerical study of (2.7) and (2.8); see, e.g., [2, 6, 8, 19, 20, 21, 26]. Of particular significance in the development of these schemes is the numerical conservation of the invariants (2.9) and (2.10). Particularly, conservation of energy implies the L^2-boundedness of the solution, thus preventing blow-up of the computed solution.

For the numerical treatment of the NLS equation, we assume that for the time interval under consideration, the solution vanishes outside some interval (x_L, x_R) and we introduce (artificial) homogeneous Dirichlet boundary conditions

$$u\left(x_L, t\right) = u\left(x_R, t\right) = 0 \tag{2.11}$$

thereby converting the pure initial-value problem (2.7) and (2.8) into an initial-boundary value problem. Note that homogeneous Neumann boundary conditions $u_x(x_L, t) = u_x(x_R, t) = 0$ could have been used as well.

Moreover, the complex-valued function $u(x, t)$ is decomposed into its real and imaginary parts $u(x, t) = v(x, t) + iw(x, t)$ so that (2.7), (2.8), and (2.11) can be re-written as

$$\begin{aligned} v_t + w_{xx} + qw\left(v^2 + w^2\right) &= 0 \\ w_t - v_{xx} - qv\left(v^2 + w^2\right) &= 0 \end{aligned} \tag{2.12}$$

$$\begin{aligned} v(x, 0) &= v_0(x) \\ w(x, 0) &= w_0(x) \end{aligned} \tag{2.13}$$

$$\begin{aligned} v\left(x_L, t\right) &= v\left(x_R, t\right) = 0 \\ w\left(x_L, t\right) &= w\left(x_R, t\right) = 0\,. \end{aligned} \tag{2.14}$$

In the following sections, several particular cases, including the propagation of a single soliton, the interaction between two solitons, and the bound state of three solitons, are investigated.

2.3.1.1 Propagation of a Single Soliton

The initial condition is given by

$$u_0(x) = \sqrt{2a/q}\exp\left[i0.5s\left(x - x_0\right)\right]\operatorname{sech}\left[\sqrt{a}\left(x - x_0\right)\right] \tag{2.15}$$

and the corresponding soliton solution is

$$u(x, t) = \sqrt{2a/q}\exp\left[i0.5s\left(x - x_0\right) - \left(0.25s^2 - a\right)t\right]\operatorname{sech}\left[\sqrt{a}\left(x - x_0\right) - st\right]\,. \tag{2.16}$$

The modulus $|u(x, t)|$ represents a wave initially located in $x = x_0$ traveling with velocity s in the positive direction of x. The amplitude $\sqrt{2a/q}$ is determined by the real parameter a.

As in [20, 21], the problem is solved for $q = 1, a = 1, s = 1$, and $x_0 = 0$. The time interval of interest is (0, 30) and, accordingly, the artificial boundaries are located at $x_L = -30$ and $x_R = 70$. This simple example is used as a first test for our adaptive mesh refinement algorithm, allowing the effect of the several tuning parameters to be highlighted.

Numerical Results The second-order spatial derivatives in (2.12) are approximated using a 5-point centered finite-difference scheme. The resulting system of semi-discrete ODEs is integrated using the implicit RK solver RADAU5, with absolute and relative error tolerances set to atol = rtol = 10^{-5}.

First, tuning of the adaptive mesh refinement algorithm is accomplished in order to obtain good numerical accuracy and computational efficiency. A curvature monitor function (2.4) is used with the tuning parameters $\alpha = 10^{-5}$, $\beta = \infty$ (no limitation of the second-order derivative), $c = 0.1$, $K = 1.2$, and $N_{\text{adapt}} = 1$ (alternate time integration and grid adaptation). Accuracy is evaluated by computing the L_2-norm of the error in the numerical solution as compared to the exact solution (2.16)

$$\|e\|_2 = \sqrt{\frac{1}{(x_R - x_L)} \sum_{i=1}^{N-1} \frac{(x_{i+1} - x_i)}{2} \left(\text{error}\,(x_i)^2 + \text{error}\,(x_{i+1})^2\right)}\,. \qquad (2.17)$$

Figure 2.1 shows the computed solution at time $t = 0, 5, 10, 15, 20, 25, 30$ (dots) and the exact solution (solid line). The location of the adaptive grid points is displayed at the bottom of the figure.

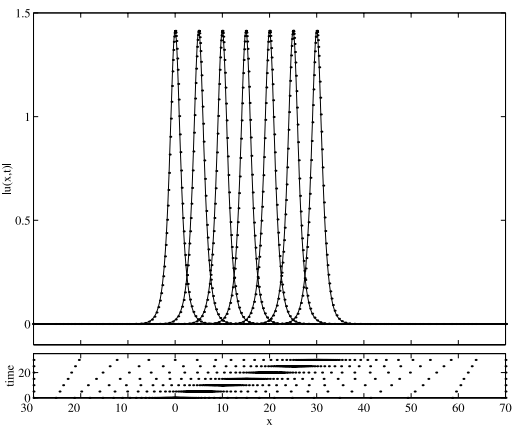


FIGURE 2.1
CSE: propagation of a single soliton every 5 units in t — numerical solution on an adaptive grid (dots) and exact solution (solid line).

With a fixed, uniform grid, $N = 651$ nodes are required to achieve the same level of accuracy (but not the same level of graphical resolution since a smaller number of nodes are concentrated in the peak, yielding a coarser picture of the soliton). Table 2.1 compares the computational statistics (i.e., the number of function evaluations FNS, the number of Jacobian evaluations JACS, the number of computed steps STEPS, and the computational costs CPU; for simplicity, the computational costs have been normalized with respect to the CPU of the adaptive grid solution) when using an adaptive or a fixed, uniform grid. In the latter case, time integration can be performed without interruption. However, it is interesting to consider the situation where it is halted after every time integration step, in order to evaluate the computational costs associated with the solver restarts (apparently, about 50% more computation time is required when restarting the solver periodically, resulting however in a slightly better overall accuracy). Note that the values of $\|e\|_2$ reported in Table 2.1 are average values over the time span of interest.

Clearly, the adaptive grid algorithm performs very satisfactorily, both in terms of accuracy and computational costs. The number of nodes is almost constant, a logical observation since the soliton propagates without change of shape.

The effects of the several tuning parameters are now investigated. In this example, α and β do not play a significant role, i.e., α and β can be set to 0 and ∞, respectively (a small value of α, as the one selected in our reference run, has a positive effect on the grid regularity). If c is reduced, the number of nodes increases, resulting in improved accuracy and in an almost proportional increase of the CPU time (which shows that the grid regularity — which is determined by K — is unaffected). If K is reduced, grid regularity is enforced, resulting in an increase of the number of nodes and, in turn, of the CPU time. The influence of c and K on the grid spacing Δx_i is illustrated in Figures 2.2 and 2.3. The grid adaptation period N_{adapt} can be increased up to 10, without significant effect on the accuracy, but with a positive effect on the computation time which is reduced to 0.52. It is possible to further increase N_{adapt} (up to 40) and obtain a satisfactory numerical solution. However, too infrequent grid adaptation yields an increase in the number of nodes (to compensate for their inappropriate placement) and a more difficult time-stepping procedure (characterized by increasing computation time).

Table 2.1 Propagation of a Single Soliton (CSE): Computational Statistics and Average Values of the L_2-Norm of the Error

Grid	N	N_{adapt}	FNS	JACS	STEPS	CPU	$\|e\|_2$
Adaptive	$85 - 86$	1	2328	420	426	1.0	$\approx 5 \times 10^{-5}$
Uniform	651	1	1737	312	318	5.0	$\approx 6 \times 10^{-5}$
Uniform	651	∞	1047	144	150	2.7	$\approx 8 \times 10^{-5}$

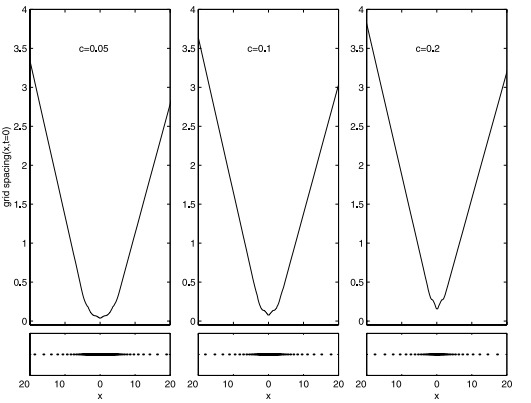


FIGURE 2.2
Influence of c on the grid spacing Δx_i.

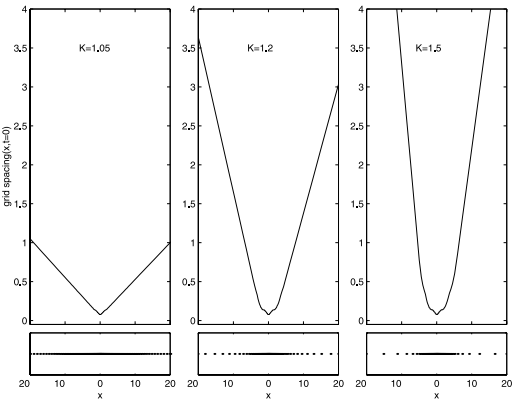


FIGURE 2.3
Influence of K on the grid spacing Δx_i.

2.3.1.2 Interaction of Two Solitons

Consider now an initial condition given by

$$\begin{aligned} u_0(x) &= \sqrt{2a_1/q}\exp\left[i0.5s_1(x-x_{01})\right]\operatorname{sech}\left[\sqrt{a_1}\,(x-x_{01})\right] \\ &\quad +\sqrt{2a_2/q}\exp\left[i0.5s_2\,(x-x_{02})\right]\operatorname{sech}\left[\sqrt{a_2}\,(x-x_{02})\right] \end{aligned} \quad (2.18)$$

which is the superposition of two solitons with amplitudes a_1 and a_2, respectively, located in x_{01} and x_{02}, and traveling with speed s_1 and s_2.

Specifically, we consider two solitons with different amplitudes and initial locations, propagating in opposite directions, e.g., $a_1 = 0.2, a_2 = 0.5, x_{01} = 0, x_{02} = 25$, $s_1 = 1.0$, $s_2 = -0.2$. The two solitons interact as if they were particle-like entities, i.e., they exhibit elastic collisions from which they emerge with the same shape. The

time interval of interest is $(0, 45)$ and, accordingly, the artificial boundaries are located at $x_L = -20$ and $x_R = 80$.

Numerical Results This example motivates the use of an adaptive grid technique with a *variable number of nodes*. Indeed, more nodes are required for reproducing the two separate solitons traveling in opposite direction than for capturing the interaction of these two entities. A curvature monitor function (2.4) is used with the tuning parameters $\alpha = 10^{-5}$, $\beta = \infty$ (no limitation of the second-order derivative), $c = 0.1$, $K = 1.5$, and $N_{\text{adapt}} = 5$. Tolerances atol = rtol $= 10^{-6}$ are imposed for the time integration with RADAU5. The number of nodes varies between $N = 89$ and 109 over the time interval $(0, 45)$. Figures 2.4, 2.5, and 2.6 show snapshots at $t = 0, 20, 45$, respectively, and compare the adaptive grid solution (dots) with a reference solution computed with 2001 fixed nodes (solid line) as well as with a fixed grid solution with $N = 140$ nodes (dashed line), which requires the same computational expense as the adaptive grid solution. Clearly, the adaptive grid algorithm performs very satisfactorily, whereas the fixed grid solution with $N = 140$ nodes is unacceptable.

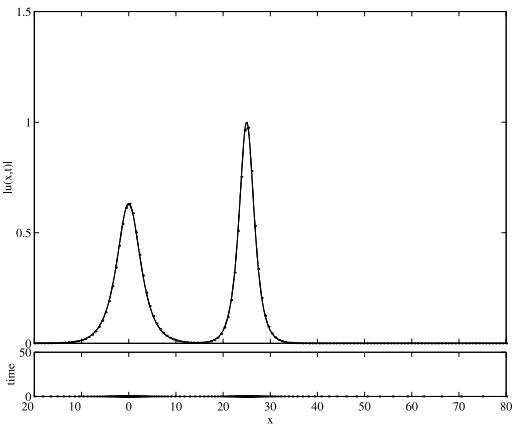


FIGURE 2.4
Two solitons traveling in opposite directions $(t = 0)$**.**

Figure 2.7 shows the original curvature-monitor function (2.4) computed on the initial condition (2.18) and the padded monitor function (2.6).

2.3.1.3 Bound State of Several Solitons

The parameters a and s are independent so that solitons with different amplitudes can move at the same speed, all the time interacting with one another. In the early 1980s, Miles [15] showed that for $q = 2N^2$ (where N is a positive integer) and an initial condition given by

$$u_0(x) = \operatorname{sech}(x) \tag{2.19}$$

$u(x, t)$ is a bound state of N solitons.

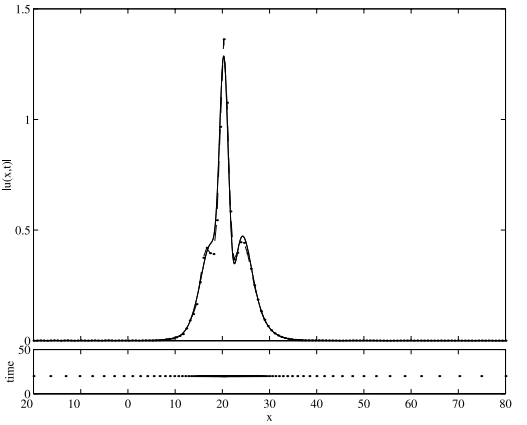


FIGURE 2.5
Interaction of two solitons $(t = 20)$**.**

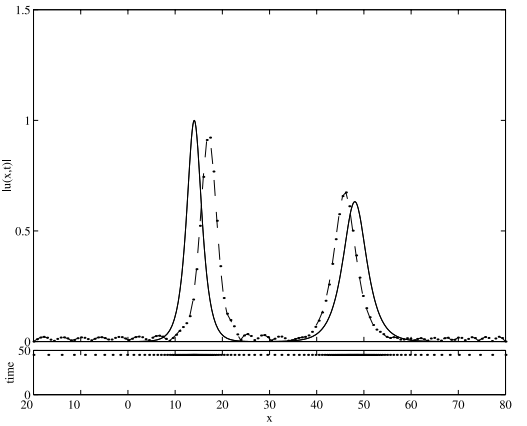


FIGURE 2.6
Two solitons after an elastic collision $(t = 45)$**.**

The bound state of several solitons results in very steep gradients in space and time and provides a more severe test of our numerical scheme than the two previous situations. As in [8, 20, 21], we consider the bound state of three solitons, i.e., $q = 18$, and use artificial boundaries located at $x_L = -20$ and $x_R = 20$. In this problem, the solution is periodic in time, a period being approximately $T = 0.8$. In the following numerical investigations, we are particularly interested in the integration of (2.7) and (2.19) over large time intervals. Indeed, results reported in previous studies (see, e.g., [8, 22]) show that accuracy deteriorates as time evolves [phase errors, non-conservation of the invariants (2.9) and (2.10), spurious oscillations, non-symmetric profiles, and eventually blow-up of the numerical solution].

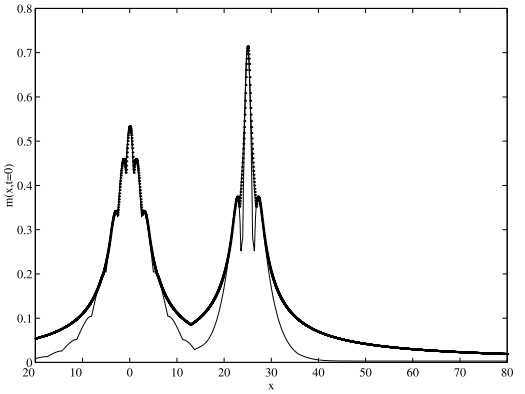


FIGURE 2.7
Monitor function m (solid line) and padded monitor function $P(m)$ (dotted line) $(t = 0)$.

Numerical Results A curvature monitor function (2.4) is used with the tuning parameters $\alpha = 2.5 \times 10^{-5}$, $\beta = \infty$ (no limitation of the second-order derivative), $c = 0.07$, $K = 1.5$, and $N_{\text{adapt}} = 5$. Tolerances atol = rtol = 10^{-5} are imposed for the time integration with RADAU5.

Over a period $(0, 0.8)$, the number of nodes varies between $N = 66$ and 157, depending on the profile complexity and sharpness. To see what happens in longer time integrations, the problem is solved over the interval $(0, 4)$, i.e., over 5 periods. The quality of the numerical solution can be checked by graphical inspection (see Figure 2.8 where the numerical results are graphed every 0.2 units of time) and by monitoring the conservation of the invariants (2.9) and (2.10), which should be close to their exact values $E = 2$ and $H = 2/3(1 - q) = -34/3 = -11.333$.

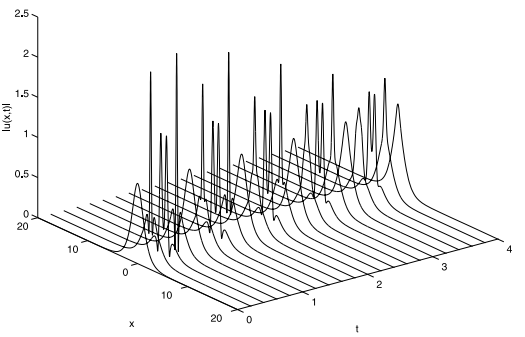


FIGURE 2.8
Bound state of three solitons from $t = 0$ to $t = 4$ at time intervals of 0.2.

Indeed, Figure 2.9 shows that the approximations to these quantities (computed by numerical quadrature) are very well conserved (the average values are $\bar{E} = 2.003$ and $\bar{H} = -11.403$).

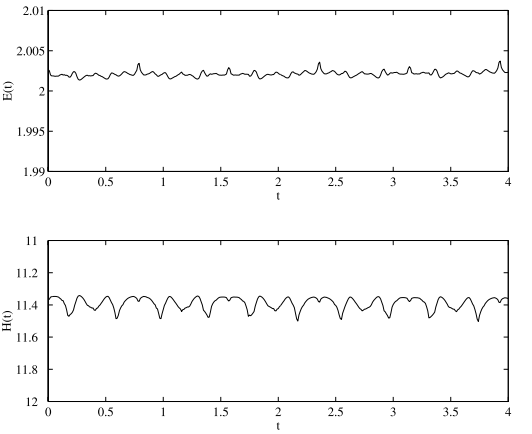


FIGURE 2.9
Evolution of the conserved quantities E (top) and H (bottom).

From Figure 2.8, the periodic behavior is observed throughout the time interval $(0, 4)$, but the numerical solution suffers from phase errors. For instance, the solution profile computed at $t = 0.6$ should be reproduced at $t = 1.4$, 2.2, and 3.8. However, Figure 2.10, which compares the profile at $t = 0.6$ (solid line) with the profiles computed at $t = 3.73, 3.74, 3.75, 3.76$, respectively (dotted lines), shows that this profile is reproduced at $t = 3.75$ (i.e., with a phase error of 0.05 unit of time or 22.5°). It is important to note that these phase errors are twice as big (e.g., 45°) when a solution is computed on a fixed, uniform grid with $N = 3001$ nodes, which demonstrates the superiority of the adaptive grid refinement method.

2.3.2 The Derivative Nonlinear Schrödinger Equation

The derivative nonlinear Schrödinger equation

$$iu_t + u_{xx} + i\left(u\left|u^2\right|\right)_x = 0, \quad -\infty \le x \le \infty, \quad t \ge 0 \tag{2.20}$$

$$u(x, 0) = u_0(x) \tag{2.21}$$

was originally derived by Mjolhus [16] to describe the long wavelength propagation of circular polarized waves parallel to the magnetic field in a cold plasma. In (2.20), $u(x, t) = v(x, t) + iw(x, t)$ represents the transverse components of the magnetic field to lowest order. The time and space coordinates t, x are in a reference frame traveling with the Alvén speed. Using the inverse scattering technique, Kaup and Newell [11] obtained the one-soliton solution and demonstrated the existence of an infinity of conservation laws.

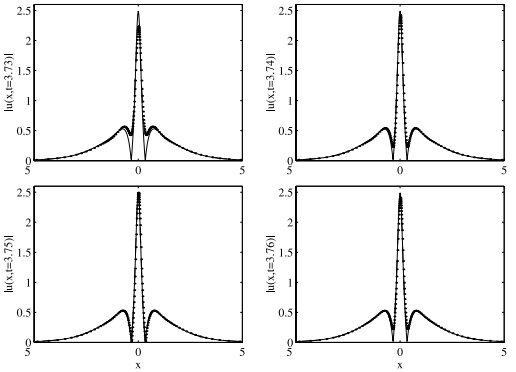


FIGURE 2.10
Phase error evaluation: comparison between solution profile at $t = 0.6$ and computed profiles at $t = 3.73, 3.74, 3.75$, and 3.76.

Here, we investigate numerically a simple example corresponding to an initial condition in the form

$$u_0(x) = \text{sech}(x) \; . \tag{2.22}$$

2.3.2.1 Numerical Results

As the sech-initial conditions (2.22) disperse away, the adaptive grid algorithm has to gradually add nodes in spatial regions further away from the initial location. The time span of interest is (0, 50) and, accordingly, artificial homogeneous Dirichlet boundary conditions are imposed at $x_L = -300$ and $x_R = 500$.

Figures 2.11 and 2.12 show the transverse component profiles $v(x, t)$ and $w(x, t)$ at $t = 0, 5, 10, 15$ computed on an adaptive grid based on a curvature monitor function with the tuning parameters $\alpha = 10^{-4}$, $\beta = \infty$, $c = 0.05$, $K = 1.1$, and $N_{\text{adapt}} = 5$. Tolerances atol = rtol = 10^{-5} are imposed for the time integration with RADAU5. The evolution of the solution modulus $|u(x, t)|$ is graphed every 5 units in t, along with the location of the adaptive grid points, in Figure 2.13. On the time interval (0, 50), the number of nodes gradually increases from $N = 276$ to 2831. In this case, the main advantage of a refinement procedure over a fixed uniform grid solution is to allow, at any time, a fine description of the solution details [for instance, the initial condition is a very narrow peak, which would require a very large number of nodes on the complete space interval $(-300, 500)$ to be represented accurately].

2.3.3 The Korteweg-de Vries Equation

This equation was originally introduced by Korteweg and de Vries in 1895 [14] to describe the behavior of small amplitude shallow-water waves in one space dimension. Over the years, the KdV equation has found application in several areas, including plasma physics, liquid-gas bubble mixtures, and anharmonic crystals.

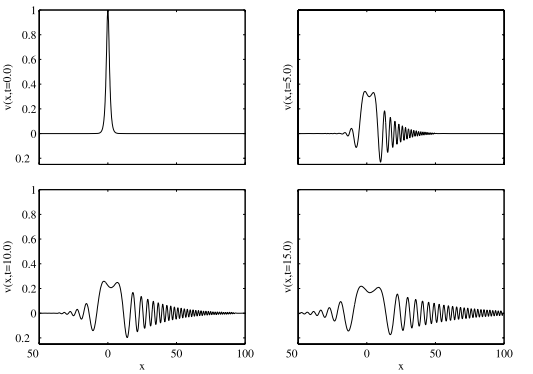


FIGURE 2.11
DNLS equation: graph of the transverse component $v(x, t)$ at $t = 0, 5, 10, 15$.

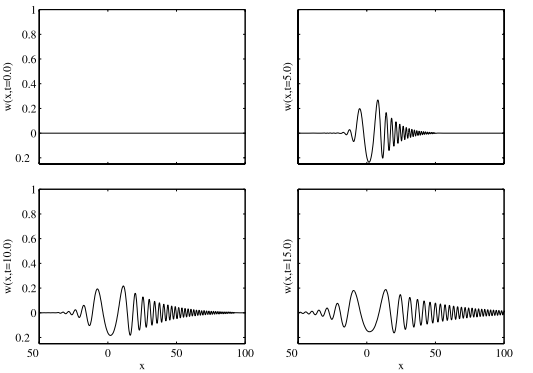


FIGURE 2.12
DNLS equation: graph of the transverse component $w(x, t)$ at $t = 0, 5, 10, 15$.

In this section, attention is focused on the classical KdV equation given by

$$u_t + 6uu_x + u_{xxx} = 0 \quad -\infty \le x \le \infty, \quad t \ge 0 \tag{2.23}$$

$$u(x, 0) = u_0(x) \tag{2.24}$$

which combines the effect of nonlinearity and dispersion. The spectral approach (or inverse scattering transform [4]) has had a major impact on the analysis of the KdV equation. This approach can be used to produce analytical solutions as well as to develop numerical algorithms; see for instance the surveys of Taha and Ablowitz [27] and Nouri and Sloan [17].

The IVP (2.23) and (2.24) possesses an infinity of invariants, e.g., the conservation of mass

$$I_1(u) = \int u \, dx \tag{2.25}$$

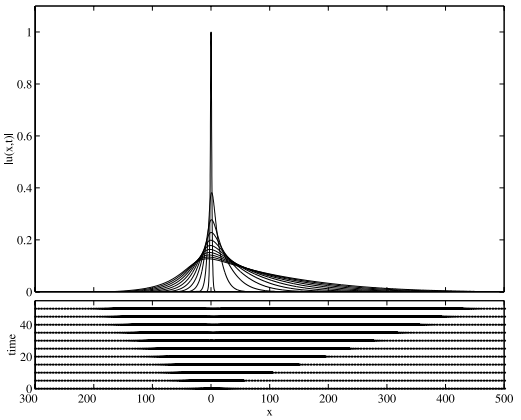


FIGURE 2.13
DNLS equation: evolution of the modulus of the solution every 5 units in t.

the conservation of energy

$$I_2(u) = \int u^2 \, dx \tag{2.26}$$

and a conservation law proposed by Whitham [28]

$$I_3(u) = \int \left(2u^3 - u_x^3\right) \, dx \; . \tag{2.27}$$

In the following, the propagation of a single soliton

$$u(x,t) = 0.5 \; s \; \text{sech}^2 \left[0.5\sqrt{s}(x - st)\right] \tag{2.28}$$

is investigated numerically. Particularly, the importance of appropriate finite difference approximations for the dispersive term u_{xxx} in (2.23) is highlighted.

2.3.3.1 Numerical Results

The KdV equation is solved for an initial condition given by (2.28) with $s = 0.5$, i.e.,

$$u_0(x) = 0.25 \; \text{sech}^2 \left[0.5^{3/2} x\right]$$

For the time span under consideration, e.g., $0 \leq t \leq 70$, it is assumed that the solution vanishes outside a finite interval $[-30, 70]$. At the endpoints $x_L = -30$ and $x_R = 70$, homogeneous Dirichlet boundary conditions are imposed, so that the pure IVP (2.23) and (2.24) is converted into an IBVP.

One of the main difficulties encountered in the MOL solution of the KdV equation is the approximation of the dispersive term u_{xxx}, which appears as a primary determinant of the solution accuracy. In a previous work [23], the authors observed

that higher-order finite difference schemes for the third-order spatial derivative (e.g., 5-, 7-, 9-, or 11-point centered schemes), which perform satisfactorily on a uniform spatial grid, give poor results on a nonuniform, adaptive grid. Instead, lower-order stagewise difference schemes, e.g., successive numerical computation of a first-order derivative $u_{xxx} = ((u_x)_x)_x$, produce very satisfactory solutions. Further numerical investigations confirm these observations and show that, for long integration times [the results presented in [23] were restricted to integrating over a relatively short time interval $(0, 30)$], simulation runs using higher-order finite difference schemes eventually fail, even on a fixed uniform grid. Hence, stagewise differentiation is used throughout the present study (the reader interested in the stability and convergence analysis of specific schemes for third-order derivative terms is referred to [5]). The resulting system of semi-discrete ODEs is integrated using the implicit RK solver RADAU5, with absolute and relative error tolerances set to atol = rtol = 10^{-5}.

Very satisfactory numerical results can be obtained using either an arc-length monitor function (2.3) with $\alpha = 0$, $\beta = \infty$, $c = 0.005$, $K = 1.1$, and $N_{\text{adapt}} = 1$, or curvature monitor function (2.4) with $\alpha = 10^{-5}$, $\beta = \infty$, $c = 0.02$, $K = 1.1$, and $N_{\text{adapt}} = 1$. In both cases, grid regularity is enforced using a relatively small value for $K = 1.1$ (remember that $K = 1$ corresponds to a uniform grid). This constraint on the grid regularity is related to the delicate approximation of the third-order derivative term. In contrast with the discussion on the effect of K when studying the CSE equation, numerical experiments show that if K is increased, the computation time does not decrease with the number of nodes, indicating that grid distortion is detrimental to the time-stepping procedure.

Figures 2.14 and 2.15 show how differently the nodes are distributed according to these two monitor functions. To achieve similar accuracy, a uniform grid with $N = 701$ nodes is required. Tables 2.2 and 2.3 give the computational statistics and several indicators of the solution quality, i.e., the L_2-norm of the error (2.17) (average values over the time span of interest) and the values of the invariants at the final time $t = 70$ (for the example under consideration, the exact [up to 5 decimal places] values of the invariants are $I_1 = 1.41421$, $I_2 = 0.11785$, $I_3 = 0.7071$). In this application example, the adaptive grid solution does not provide much benefit in terms of computational expense, but allows a finer graphical resolution of the solitary wave.

Table 2.2 Propagation of a Single Soliton (KdVE): Computational Statistics

Grid	N	N_{adapt}	FNS	JACS	STEPS	CPU
Adaptive (2.3)	$153 - 154$	1	965	187	197	1.0
Adaptive (2.4)	$145 - 147$	1	1051	249	259	1.1
Uniform	701	1	720	128	135	2.5
Uniform	701	∞	452	59	65	1.5

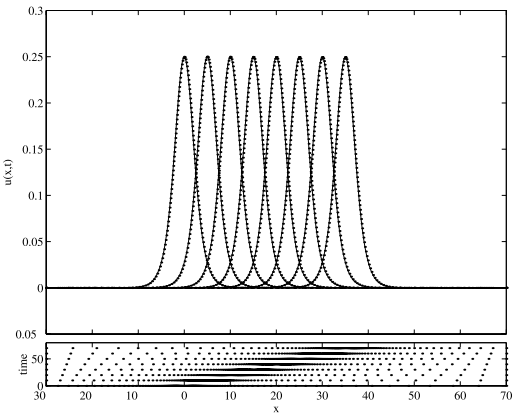


FIGURE 2.14
KdV equation: propagation of a single soliton every 10 units in t: numerical solution on an adaptive grid based on an arc-length monitor (dots) and exact solution (solid line).

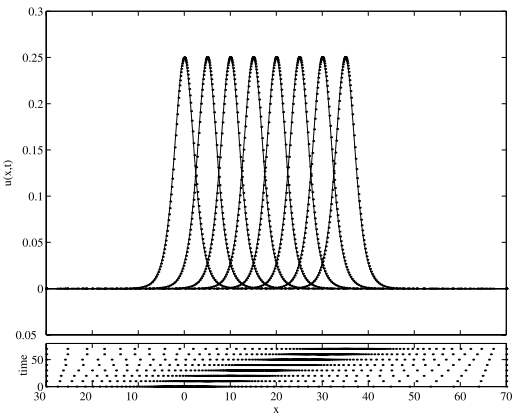


FIGURE 2.15
KdV equation: propagation of a single soliton every 10 units in t — numerical solution on an adaptive grid based on a curvature monitor (dots) and exact solution (solid line).

2.3.4 The Korteweg-de Vries-Burgers Equation

Johnson [10] derived the Korteweg-de Vries-Burgers (KdVB) equation in the study of the weak effects of dispersion, dissipation, and nonlinearity in waves propagating in a liquid-filled elastic tube. The KdVB equation for $u(x,t)$ is given by

$$u_t + 2auu_x + 5bu_{xx} + cu_{xxx} = 0 \quad -\infty \le x \le \infty, \quad t \ge 0 \tag{2.29}$$

$$u(x,0) = u_0(x)\,. \tag{2.30}$$

Table 2.3 Propagation of a Single Soliton (KdVE): Conserved Quantities and Average Values of L_2-Norm of the Error

Grid	N	I_1	I_2	I_3	$\|e\|_2$
Adaptive (2.3)	$153 - 154$	1.41217	0.11746	0.07033	$\approx 1.2 \times 10^{-4}$
Adaptive (2.4)	$145 - 147$	1.41563	0.11808	0.07099	$\approx 1.4 \times 10^{-4}$
Uniform	701	1.41292	0.11784	0.07076	$\approx 1.2 \times 10^{-5}$

In the limits $b \to 0$ or $c \to 0$, i.e., when the effects of dissipation or dispersion are negligible, the KdVB equation reduces either to the KdV equation

$$u_t + 2auu_x + cu_{xxx} = 0 \tag{2.31}$$

(which has been investigated numerically for $a = 3$ and $c = 1$ in the previous section) or the well-known Burgers equation

$$u_t + 2auu_x + 5bu_{xx} = 0 \tag{2.32}$$

In [9], Jeffrey and Xu introduced a transformation that reduces the KdVB equation to a quadratic form which can be solved in terms of a series of exponentials. In contrast with the KdV equation, which possesses an infinite sequence of exact solutions (the n-soliton solutions), the quadratic form of the KdVB equation yields only two traveling waves given by

$$u_1(x,t) = \frac{3b^2}{2ac}\left[\operatorname{sech}^2\left(\frac{\vartheta}{2}\right) + 2\tanh\left(\frac{\vartheta}{2}\right) + 2\right] \tag{2.33}$$

with $\vartheta = \left(\frac{b}{c}\right)x - \left(\frac{6b^3}{c^2}\right)t + \beta$ and

$$u_2(x,t) = \frac{3b^2}{2ac}\left[\operatorname{sech}^2\left(\frac{\vartheta}{2}\right) - 2\tanh\left(\frac{\vartheta}{2}\right) - 2\right] \tag{2.34}$$

with $\vartheta = -\left(\frac{b}{c}\right)x - \left(\frac{6b^3}{c^2}\right)t + \beta$.

These solutions cannot be reduced to the sech^2 solution to the KdV equation in the limit $b \to 0$ or to the tanh solution to Burgers equation in the limit $c \to 0$.

2.3.4.1 Numerical Results

The problem is solved for $a = 1$, $b = -1$, $c = 3$, and $\beta = 0$ and an initial condition corresponding to (2.33). The time interval of interest is (0, 15) and, accordingly, the artificial boundaries are located in $x_L = -15$ and $x_R = 100$. At the endpoints $x_L = -15$ and $x_R = 100$, homogeneous Dirichlet boundary conditions are imposed, so that the pure IVP (2.29) and (2.30) is converted into an IBVP. As for the classical KdV equation, stagewise differentiation for the approximation of the

third-order derivative term yields better performance than direct differentiation using, e.g., a 7-point centered finite difference scheme. Tolerances atol = rtol = 10^{-4} are imposed for the time integration with RADAU5.

The best results, in terms of accuracy and computational expense, are obtained with a curvature monitor function and the following parameters: $\alpha = 0$, $\beta = \infty$, $c = 0.03$, $K = 1.1$, and $N_{\text{adapt}} = 1$. The number of nodes is almost constant, i.e., $N \approx 141$. The solution is graphed at time $t = 0, 3, 6, 9, 12, 15$ in Figure 2.16.

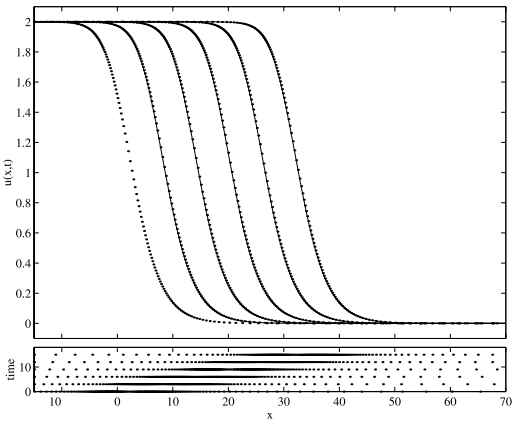


FIGURE 2.16
KdVB equation: numerical solution on an adaptive grid based on a curvature monitor (dots) and exact solution (solid line) graphed every 3 units in t.

2.3.5 KdV-Like Equations: The Compactons

Seeking to understand the role of nonlinear dispersion in the formation of patterns in liquid drops, Rosenau and Hyman [18] introduced a family of fully nonlinear KdV-like equations in the form

$$u_t + \left(u^m\right)_{xx} + \left(u^n\right)_{xxx} = 0, \qquad m > 0, \quad 1 < n \leq 3\,. \tag{2.35}$$

These equations, which are denoted $K(m, n)$, have the property that for certain m and n, their solitary wave solutions have compact support. This remarkable property has suggested the name "compacton" to these authors.

In fact, nonlinear dispersion in (2.35) (which is accounted for by the term $(u^n)_{xxx}$) is weaker for small u than linear dispersion in the classical KdV equation, which allows the formation of a compact-support solution. On the other hand, dispersion is much more important at high amplitudes and counterbalances the steepening effect of nonlinear convection.

Particularly, the $K(2, 2)$ equation possesses a solitary wave solution with a compact support given by

$$\begin{aligned} u_c(x,t) &= \frac{4s}{3}\cos^2\left(\frac{x-st}{4}\right), \qquad |x-st| \le 2\pi \\ &= 0, \qquad\qquad\qquad\quad \text{otherwise .} \end{aligned} \tag{2.36}$$

Although the second derivative of the compacton is discontinuous at its edges, (2.36) is a strong solution of Equation (2.35) since the third derivative is applied to u^2, which has three smooth derivatives everywhere including the edge.

The compacton's amplitude depends on its speed but, unlike the KdV-soliton which narrows as the speed increases, its width is independent of the speed.

Similarly to the soliton interactions associated with the cubic Schrödinger equation or the classical Korteweg-de Vries equation, compactons exhibit elastic collisions, in which, after colliding with other compactons, they emerge with the same coherent shape. However, the point where two compactons interact is marked by the birth of a low amplitude compacton-anticompacton pair.

2.3.5.1 Numerical Results

First, attention is focused on the propagation of a single compacton (2.36) with speed $s = 0.5$ over the time interval $(0, 80)$. Accordingly, homogeneous Dirichlet boundary conditions are imposed at $x_L = -30$ and $x_R = 70$.

As stressed in [18], there are several numerical difficulties in solving the $K(2, 2)$ equation, which are due to nonlinear dispersion and the lack of smoothness at the edge of the compacton, possibly leading to instability.

Using the numerical methods described in the previous sections, it was not possible to solve satisfactorily the $K(2, 2)$ equation on a fixed uniform grid, with the exception of the particular setting:

- stagewise differentiation of the nonlinear dispersive term
- $N = 501$ spatial nodes
- time integration with atol = rtol = 10^{-5}

Even in this fortuitous situation, the graph of the solution (chat 2.17) displays unacceptable downstream oscillations. However, any attempt to improve on this situation by increasing the number of nodes or reducing the error tolerances leads to failure of the simulation run.

When using the adaptive grid procedure, only the arc-length monitor function allows the problem to be solved (i.e., every attempt to solve this problem with a curvature monitor function failed). The tuning parameters take the following values: $\alpha = 10^{-6}$, $\beta = \infty$, $c = 0.01$, $K = 1.1$, and $N_{\text{adapt}} = 1$. The corresponding solution, which is now very satisfactory, is graphed every 10 units in t in Figure 2.18. In fact, with the same tuning parameters, it appears even possible to approximate the nonlinear dispersive term with a classical 7-point centered finite-difference scheme (instead of

using stagewise differentiation, which seems to be a "universal" solution for all the KdV-like problems considered thus far). In this latter case, accuracy improves at the price of larger computational costs. The computational statistics as well as the average value of the L_2-norm of the error are summarized in Table 2.4.

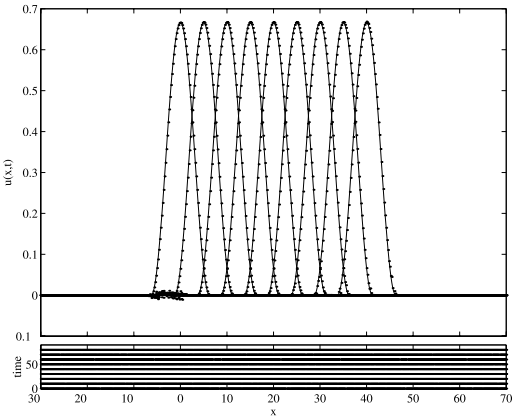


FIGURE 2.17
Propagation of a single compacton every 10 units in t: numerical solution on a fixed uniform grid with $N = 501$ nodes (dots) and exact solution (solid line).

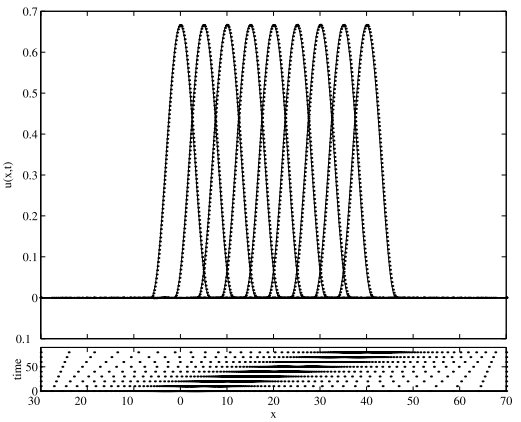


FIGURE 2.18
Propagation of a single compacton every 10 units in t: numerical solution on an adaptive grid based on an arc-length monitor (dots) and exact solution (solid line).

Second, the interaction of two compactons initially centered in $x_{01} = 0$ and $x_{02} = 15$, respectively, and traveling in the same direction but with different speeds $s_1 = 0.5$ and $s_2 = 0.25$, is considered. The time span of interest is now $(0, 120)$. As time

Table 2.4 Propagation of a Single Compacton: Computational Statistics and Average Values of the L_2-Norm of the Error

Grid	N	Approx.	N_{adapt}	FNS	JACS	STEPS	CPU	$\|e\|_2$
Adaptive	$203-204$	$((u_x)_x)_x$	1	1813	228	448	1.0	$\approx 1.5\times 10^{-3}$
Adaptive	$203-204$	u_{xxx}	1	3258	513	798	2.3	$\approx 3.5\times 10^{-4}$
Uniform	501	$((u_x)_x)_x$	1	1514	367	374	2.5	$\approx 1.2\times 10^{-3}$

evolves, the faster compacton catches the slower one and passes through it. The point where the two compactons collide is marked by the birth of a low amplitude compacton-anticompacton pair.

All our efforts to solve this challenging problem on a fixed uniform grid were unsuccessful, and only an adaptive grid solution based on an arc-length monitor function could be obtained, after quite a lot of tuning, for the parameter setting:

- stagewise differentiation of the nonlinear dispersive term
- $\alpha = 10^{-3}$, $\beta = \infty$, $c = 0.015$, $K = 1.5$, and $N_{\text{adapt}} = 1$
- time integration with atol $=$ rtol $= 10^{-4}$

In contrast with our previous experiments with the KdV equation, a large value of α is used to force a relatively high and almost uniform density of nodes outside the compactons, whereas a large value of K allows grid deformations and higher node concentrations in the compactons. The solution is graphed every 10 units in t in Figure 2.19.

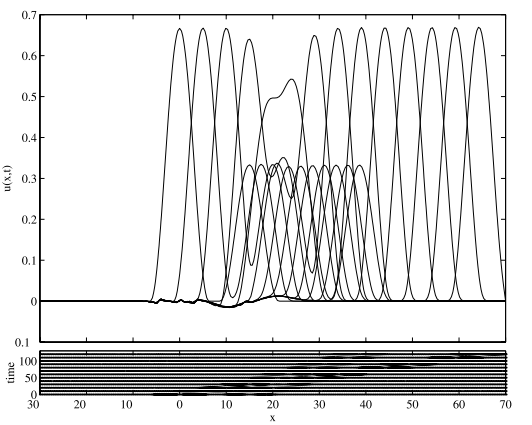


FIGURE 2.19
Interaction between two compactons every 10 units in t.

2.4 Conclusions

In this chapter, a simple static grid refinement algorithm is implemented and applied to a set of nonlinear dispersive wave problems. The number and placement of the nodes is determined by equidistributing a monitor function related to the arc-length or the curvature of the computed solution. As grid distortion is detrimental to spatial accuracy and stiffness of the semi-discrete system of ordinary differential equations, spatial grid regularization is accomplished by padding the monitor function. This procedure originally devised by Kautsky and Nichols [12] enforces that the ratio between adjacent grid steps lies between two bounds specified by the user.

First, some solutions of the nonlinear Schrödinger equation, including the propagation of a single soliton, the interaction between two solitons traveling in opposite direction, and the bound state of three solitons, are studied. In all these test-examples, very satisfactory numerical solutions, both in terms of accuracy and computational demand, can be obtained. Particularly, long time integration of the bound state of three solitons shows excellent conservation of invariants (energy and Hamiltonian) as well as relatively small phase errors. In addition, some numerical results for the derivative nonlinear Schrödinger equation are presented.

Second, several KdV-like equations, including the classical Korteweg-de Vries equation, the Korteweg-de Vries-Burgers equation, and a fully nonlinear KdV equation giving rise to compactons, are considered. There are several numerical difficulties in solving these equations, which are related to the approximation of the third-order spatial derivative term (dispersive term). Somewhat surprisingly, high-order "direct" finite difference schemes provide relatively poor results, whereas low-order "stage-wise" schemes (which proceed by successive numerical differentiation of lower order derivatives) appear as simple and efficient approximations in most cases.

The propagation and interaction of compactons are very challenging problems, for which numerical solutions could only be obtained at the price of a careful selection of the algorithm parameters. At this stage, further investigations are required.

References

[1] M. Berzins and R.M. Furzeland, *A User's Manual for SPRINT - A Versatile Software Package for Solving Systems of Algebraic Ordinary and Partial Differential Equations,* Thornton Research Centre, Shell Maatschappij (1985, 86 and 89).

[2] M. Delfour, M. Fortin, and G. Payne, Finite difference solution of a nonlinear Schrödinger equation, *J. Comp. Phys.,* **44**, (1981), 277–288.

[3] B. Fornberg, Generation of finite difference formulas on arbitrarily spaced grids, *Math. Comp.,* **51**, (1988), 699–706.

[4] C.S. Gardner, J. Green, M. Kruskal, and R. Muira, Method for solving the Korteweg-de Vries equation, *Phys. Rev. Lett.,* **19**, (1967), 1095–1097.

[5] B. Garcia-Archilla and J.M. Sanz-Serna, A finite difference formula for the discretization of d^3/dx^3 on nonuniform grids, *Math. Comp.,* **57**, (1991), 239–257.

[6] D.F. Griffiths, A.R. Mitchell, and J.Ll. Morris, A numerical study of the nonlinear Schrödinger equation, *Comput. Meth. Appl. Mech. Eng.,* **45**, (1984), 177–215.

[7] E. Hairer and G. Wanner, *Solving ordinary differential equations II. Stiff and differential-algebraic problems,* Springer-Verlag, Berlin, 1991.

[8] B.M. Herbst, J.Ll. Morris, and A.R. Mitchell, Numerical experience with the nonlinear Schrödinger equation, *J. Comp. Phys.,* **60**, (1985), 282–305.

[9] A. Jeffrey and S. Xu, Exact solutions to the Korteweg-de Vries-Burgers Equation, *Wave Motion,* **11**, (1989), 559–564.

[10] R.S. Johnson, A nonlinear equation incorporating damping and dispersion, *J. Fluid Mech.,* **42**, (1970), 49–60.

[11] D.J. Kaup and A.C. Newell, An exact solution for the derivative nonlinear Schrödinger equation, *J. Math. Phys.,* **19**, (1978), 798–801.

[12] J. Kautsky and N. K. Nichols, Equidistributing meshes with constraints, *SIAM J. Sci. Stat. Comput.,* **1**, (1980), 499–511.

[13] S. Kichenassamy, *Nonlinear Wave Equations,* Marcel Dekker, 1996.

[14] D.J. Korteweg and G. de Vries, On the change of form of long waves advancing in a rectangular canal and on a new type of long stationary waves, *Philos. Mag.,* **39**, (1895), 422–443.

[15] J.W. Miles, An envelope soliton problem, *SIAM J. Appl. Math.,* **41**, (1981), 227–230.

[16] E. Mjolhus, On the modulational instability of hydromagnetic waves parallel to the magnetic field, *J. Plasma Phys.,* **16**, (1976), 321–334.

[17] F.Z. Nouri and D.M. Sloan, A comparison of Fourier pseudospectral methods for the solution of the Korteweg-de Vries equation, *J. Comp. Phys.,* **83**, (1989), 324–344.

[18] P. Rosenau and J.M. Hyman, The compacton: a soliton with compact support, *Phys. Rev. Lett.,* **70**, (1993), 564.

[19] J.M. Sanz-Serna, Methods for the numerical solution of the nonlinear Schrödinger equation, *Math. Comp.,* **43**, (1984), 21–27.

[20] J.M. Sanz-Serna and J.G. Verwer, Conservative and nonconservative schemes for the solution of the nonlinear Schrödinger equation, *IMA J. Numer. Anal.,* **6**, (1986), 25–42.

[21] J.M. Sanz-Serna and I. Christie, A simple adaptive technique for nonlinear wave problems, *J. Comp. Phys.,* **67**, (1986), 348–360.

[22] P. Saucez, A. Vande Wouwer and W.E. Schiesser, Some observations on a static spatial remeshing method based on equidistribution principles, *J. Comp. Phys.,* **128**, (1996), 274–288.

[23] P. Saucez, A. Vande Wouwer, and W.E. Schiesser, An adaptive method of lines solution of the Korteweg-de Vries equation, *Comp. Math. Applic.,* **35**, (1998), 13–25.

[24] G. Steinebach, *Die Linienmethode und ROW-Verfahren zur Abfluss- und Prozess-simulation in Fliessgewässern am Beispiel von Rhein und Mosel,* Ph.D. Thesis, Darmstadt Technical University, Germany, 1995.

[25] C. Sulem and P.-L. Sulem, *The Nonlinear Schrödinger Equation — Self-Focusing and Wave Collapse,* Springer-Verlag, Berlin, 1999.

[26] T.R. Taha and M.J. Ablowitz, Analytical and numerical aspects of certain nonlinear evolution equation. II. Numerical, nonlinear Schrödinger equation, *J. Comp. Phys.,* **55**, (1984), 203–230.

[27] T.R. Taha and M.J. Ablowitz, Analytical and numerical aspects of certain nonlinear evolution equation. III. Numerical, nonlinear Korteweg-de Vries equation, *J. Comp. Phys.,* **55**, (1984), 231–253.

[28] G.B. Whitham, *Linear and Nonlinear Waves,* Wiley-Interscience, New York, 1974.

[29] V.E. Zakharov and A.B. Shabat, Exact theory of two-dimensional self-focusing and one-dimensional self-modulation of waves in nonlinear media, *Soviet Phys. JEPT,* **34**, (1972), 62–69.

Chapter 3

Numerical Solutions of the Equal Width Wave Equation Using an Adaptive Method of Lines

S. Hamdi, J.J. Gottlieb, and J.S. Hansen

Abstract The equal-width wave (EW) equation is a model partial differential equation for the simulation of one-dimensional wave propagation in media with nonlinear wave steepening and dispersion processes. The background of the EW equation is reviewed and this equation is solved by using an advanced numerical method of lines with an adaptive grid whose node movement is based on an equidistribution principle. The solution procedure is described and the performance of the solution method is assessed by means of computed solutions and error measures. Many numerical solutions are presented to illustrate important features of the propagation of a solitary wave, the inelastic interaction between two solitary waves, the breakup of a Gaussian pulse into solitary waves, and the development of an undular bore.

3.1 Introduction

Wave propagation has intrigued scientists for many centuries owing to their fascinating nonlinear behavior, originating with mankind's observation of the spectacular *breaking* of water waves. Some waves can propagate with constant shape and speed, others when perturbed can undergo decay and shed a trailing disturbance, some can partially disintegrate and shed a train of weak trailing waves, whereas others can accelerate as they become spatially narrower and blow-up in amplitude. Such wave behavior is illustrated in Figure 3.1, where different waves are combined in one time-distance diagram.

Nonlinear wave phenomena has been studied extensively in recent years by many researchers, and some of them have directed their efforts at formulating mathematical models for the description of wave propagation in media with nonlinear wave steepen-

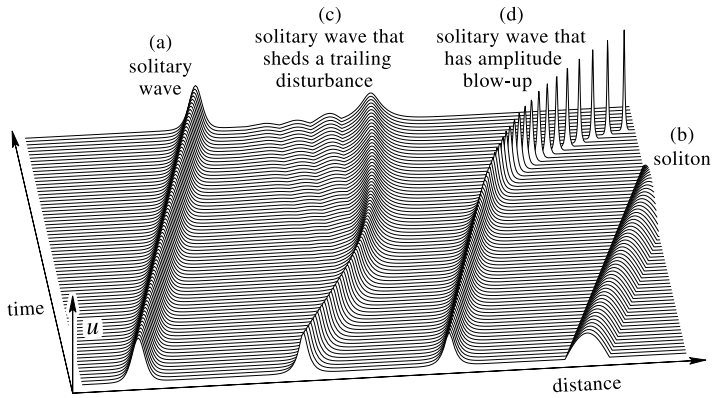


FIGURE 3.1
Illustrations of the behavior of different solitary waves.

ing and dispersion effects. The well-known Korteweg and de Vries (KdV) equation, $u_t + uu_x + u_{xxx} = 0$, is the first classical nonlinear partial differential equation (PDE) that has been very successful in this regard. This model equation, formulated by Korteweg and de Vries [19] in the year 1895, simulates the time-dependent motion of shallow water waves in one space dimension. The pioneering study by Korteweg and de Vries showed that when nonlinear wave steepening, from the nonlinear term uu_x, is balanced by wave dispersion, owing to the term linear u_{xxx}, their equation predicts a unidirectional solitary wave, that is a pulse which moves in one direction with a permanent shape and constant speed. For example, see the waves labeled (a) and (b) in Figure 3.1. A remarkable property of these solitary waves is that they can be exceptionally stable while traveling relatively long distances without undergoing any noticeable alterations in shape, amplitude, and speed.

Nonlinear wave steepening and dispersion processes are important not only in hydrodynamics but also in many other disciplines of engineering and science, in which the KdV equation has also become a powerful tool for the modeling of wave phenomena. The study of Berezin and Karpman [4] contains several examples of the propagation of nonlinear waves with moderately large wavelengths and small but finite amplitudes in liquids, compressible gases, cold plasmas, and other media with dispersive effects.

Benjamin et al. [3] advocated that the PDE $u_t + uu_x + u_x - \mu u_{xxt} = 0$ modeled the same physical phenomena equally well as the KdV equation, given the same assumptions and approximations that originally led Korteweg and de Vries [19] to their equation. This PDE of Benjamin et al. [3] is now often called the *regularized* long wave (RLW) equation, although it is also known as the BBM equation. The word regularized in RLW stems from past developments of more expedient mathematical tools and properties for the RLW equation, as compared to the KdV equation, which have facilitated rigorous proofs of the existence and uniqueness of periodic and nonperiodic solutions on an unbounded domain, and which have helped prove that these

solutions are stable and continuous with regard to various initial conditions, including the perturbation of an initially specified solitary wave.

Peregrine [23] was the first to solve the RLW equation to describe the development of undular bores, which are smooth solitary waves that were observed propagating in shallow water channels. The RLW equation was solved successfully in additional applications involving the time-dependent motion of one-dimensional drift waves in plasmas, and Morrison et al. [21] mention that the RLW equation can also be used to simulate Rossby waves for geophysical applications.

Morrison et al. [21] proposed the one-dimensional PDE, $u_t + uu_x - \mu u_{xxt} = 0$, as an equally valid and accurate model for the same wave phenomena simulated by the KdV and RLW equations. This PDE is now called the equal-width (EW) equation because the solutions for solitary waves with a permanent form and speed, for a given value of the parameter μ, are waves with an equal width or wavelength for all wave amplitudes. The EW equation is a simpler and lesser known alternative to the RLW equation, and the solitary wave solutions are less general because of the equal-width constraint.

The properties of solutions from the KdV, RLW, and EW equations can differ remarkably, even though they are model equations for similar types of wave motion. The solution of the KdV equation for a solitary wave that is initially perturbed illustrates that this wave can propagate without significant change in shape and speed and remain stable, as shown by the waves labeled (a) and (b) in Figure 3.1, whereas the solutions of the RLW and EW equations show that a perturbed solitary wave can evolve instead into a decaying wave with an oscillating tail or evolve into a contracting wave with amplitude blow-up, as depicted by the waves labeled (c) and (d) in Figure 3.1. The solution of the KdV equation for the overtaking of one solitary wave by another features two transmitted waves that retain their original shapes and speeds, but they are displaced from their original straight trajectories, as depicted by the wave system labeled (a) in Figure 3.2. Such interactions are called *elastic* or *clean* interactions. In contrast, the solutions of the RLW and EW equations for solitary wave interactions exhibit transmitted waves that can shed trailing disturbances, can split into a set of weak waves, or can increase unboundedly in speed and amplitude, as shown by the wave system labeled (b) in Figure 3.2. These latter interactions are labeled *inelastic, anelastic,* or *unclean* interactions.

The KdV equation can be solved by analytical means for some specific problems and in general by the inverse scattering transform (IST) technique and spectral methods (SMs). The RLW and EW equations cannot be solved by the IST, but these equations, as well as the KdV equation, can be solved by using various numerical techniques (e.g., the method of lines). The differences in the solution procedures (nonexistence of an IST solution) and related numerical difficulties in solving the RLW and EW equations in contrast to the KdV equation, all stem directly from the dispersive term u_{xxt} in the RLW and EW equations. Mathematically, the KdV equation is said to be *integrable* or a completely integrable Hamiltonian system (with reversible energy exchange between the degrees of freedom), whereas the RLW and EW equations are *nonintegrable*, which corresponds directly to their inelastic or un-

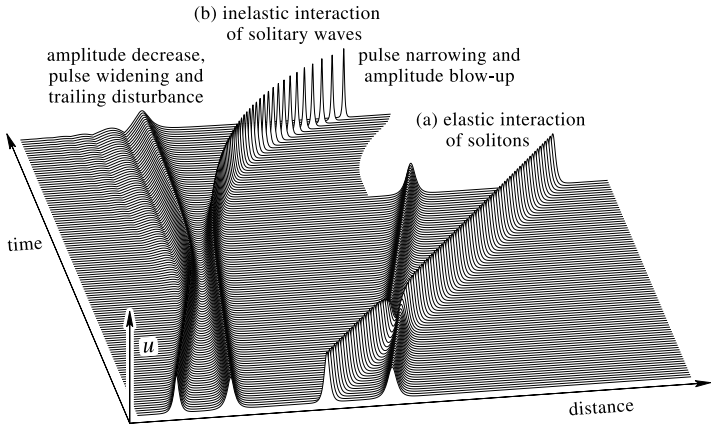


FIGURE 3.2
Illustrations of elastic and inelastic interactions of solitary waves.

clean behavior involving wave interactions. An integrable equation admits an infinite number of conservation laws and invariants of motion, whereas a nonintegrable equation has only a limited number of invariants of motion. An integrable equation can be rewritten as a compatibility condition in terms of two linear equations called the Lax pair; see the book by Whitham [32] for details. This last property is the essence of the IST method.

Distinguishing between solitary waves and solitons is sometimes important. Solitons are very special types of solitary waves that have the property of elastic wave interactions. Solitary waves associated with integrable equations, such as the KdV equation, are solitons, whereas those associated with nonintegrable equations, such as the RLW and EW equations, are not. This restricted definition is adopted herein. Note that no distinction is made between solitons and solitary waves in plasma physics and quantum mechanics.

From an historical perspective, the RLW and EW equations were originally believed to be integrable and yield elastic wave interactions, as mentioned by Santarelli [26] and Abdulloev et al. [1]. This belief stemmed from computations using inaccurate or low-resolution numerical methods, from which the small effects of inelastic wave interactions were undetected in the numerical solutions. For example, the slow decay of a solitary wave and its shedding of weak trailing waves were simply unresolved, or they may have been misinterpreted or confused as numerically generated oscillations. This numerical difficulty was first identified and overcome by Santarelli [26] in his studies of the one-dimensional collision of two solitary waves that produced large and readily observable inelastic effects in the form of a train of solitary waves between the transmitted waves. Santarelli's discoveries were confirmed later by Lewis and Tjon [20] for similar types of problems.

Accurate numerical techniques for solving the KdV, RLW, and EW equations are required to realistically capture important large and small qualitative and quantitative

features of the solution, especially when overtaking and colliding solitary waves exhibit very rapid solution variations (e.g., at shocks). Several numerical studies yielding more accurate solutions of the RLW and EW equations, and some other closely related PDEs, have been reported by Jain et al. [18] and Bona et al. [5]. The methods they reviewed are based on classical finite-difference and finite-element techniques with an accurate space discretization that is normally coupled to a low second-order time integration scheme; see Gardner et al. [10, 11] for some examples. The primary weakness of these earlier numerical methods is the use of a low-order time integration scheme for the solution of long-duration evolutionary waves. A further shortcoming stems from the use of a simple uniform grid that limits the spatial resolution of rapidly varying solutions in space.

More accurate numerical techniques have been developed recently for solving the KdV equation, and these have been implemented in the numerical method of lines (MOL) by, for example, Schiesser [30]. High-order spatial discretizations with finite-difference schemes are readily usable today with either uniform or nonuniform grids by incorporating, for example, the algorithm called `WEIGHTS` from Fornberg [8]. High-order time integration techniques for non-stiff differential equations are also readily available; for example, the time integrator or solver called `RKF45` uses an explicit variable fourth- and fifth-order Runge-Kutta method, as described by Forsythe et al. [9]. For numerically stiff and explicit systems of ordinary differential equations, several efficient time integrators are also available; for example, Hairer and Wanner [14] developed a variable time-step, fifth-order, implicit Runge-Kutta solver called `RADAU5`. For numerically stiff and implicit systems of differential-algebraic equations, Petzold [24] and Hindmarch [17] developed the time integrators called `DASSL` and `LSODI`, respectively, which are both based on variable time-step, variable-order, backward-differentiation formulae. Adaptive grid techniques, with node movement based on an equidistribution principle, have become instrumental in helping to accurately resolve very steep solution gradients and curvatures in space, and various techniques and programs are now available from a number of authors such as White [31], Sanz-Serna and Christie [27], Revilla [25] and, more recently, Saucez et al. [28, 29], who use the MOL approach.

This study focuses primarily on solving the EW equation, $u_t + uu_x - \mu u_{xxt} = 0$, by using an advanced MOL with adaptive gridding. The main numerical difficulty in solving this equation stems from the dispersive term u_{xxt} which results in the coupling of the space and time derivatives. The spatial discretization produces a fully implicit set of differential-algebraic equations, and these are integrated numerically in time by using a recently developed robust integration solver such as `DASSL` from Petzold [24]. Our adaptive MOL approach stems originally from the studies of Schiesser [30] and more recently from our related studies with the KdV and EW equations as reported by Hamdi et al. [15, 16]. In this study, our numerical techniques are described, the performance of our methods are assessed by means of numerical results and error measures, and many interesting problems involving the EW equation are solved and their solutions presented to illustrate salient features of the propagation of a solitary wave, the inelastic interaction of two solitary waves, the breakup of a Gaussian pulse into solitary waves, and the development of an undular bore.

3.2 Equal-Width Equation

Basic information and analytical tools for the equal-width (EW) and related regularized long wave (RLW) equations are provided in this section. These are prerequisites for describing the solution procedure, assessing the solution method, and understanding important features of the numerical results.

The EW equation is a partial differential equation (PDE) given by

$$u_t + uu_x - \mu u_{xxt} = 0 \,, \tag{3.1}$$

in which $u = u(x, t)$ is a function of the two independent variables x and t that normally denote space and time, respectively. As subscripts on u, x and t denote partial derivatives of the dependent variable u. The parameter μ is a positive real constant. In most fluids related problems, $u(x, t)$ represents the wave amplitude or some similar physical quantity, whereas in plasma applications it is the negative electrostatic potential. In most applications the terms uu_x and u_{xxt} produce nonlinear wave steepening and dispersion, respectively.

The EW equation is a simpler form of the RLW equation, $u_t+uu_x+u_x-\mu u_{xxt} = 0$, as mentioned in the introduction. The RLW equation is a PDE that has been used to simulate wave motion in media with nonlinear wave steepening and dispersion, such as shallow water waves and ion acoustic plasma waves. However, the simpler EW equation is an equally valid and accurate model for the same wave phenomena; see the study of Morrison et al. [21] for more details. Although the EW equation can be transformed into the RLW equation by means of $u_{EW} \rightarrow u_{RLW} + 1$, a solution of the EW equation cannot provide a solution to the RLW equation because the boundary conditions are incompatible.

The EW equation requires boundary conditions for solitary and other wave motion of the form $u(x, t) \rightarrow u_L$ and u_U as $x \rightarrow -\infty$ and $+\infty$, at which the constants u_L and u_U are normally zero. The boundary conditions for our solutions are approximated on the finite computational domain $x_L \leq x \leq x_U$ by $u(x_L, t) = u_L$ and $u(x_U, t) = u_U$, which have been used in previous studies. These are good approximations because our numerical solutions are computed when all of the initial conditions and wave motion are well within the interior of the domain, such that the amplitudes of the data and waves die out asymptotically to a constant or zero at each domain boundary.

Some important analytical solutions for the motion of solitary waves for both the EW and RLW equations are available in the paper of Morrison et al. [21]. These are given by

$$u(x, t) = 3\, c\, \mathrm{sech}^2[k\,(x - x_0 - vt)] \,, \tag{3.2}$$

for which the wave number k and wave speed ν are defined by

$$k = \begin{cases} \sqrt{\dfrac{1}{4\mu}} & \text{for EW,} \\ \sqrt{\dfrac{1}{4\mu}\,\dfrac{c}{c+1}} & \text{for RLW,} \end{cases} \qquad \nu = \begin{cases} c & \text{for EW,} \\ c+1 & \text{for RLW,} \end{cases} \tag{3.3}$$

for the two equations. These solutions correspond to solitary waves moving in the positive or negative x directions, depending on the sign of ν. These waves have a positive or negative constant peak amplitude $3c$, unchanging wave shape or profile, and steady wave speed ν. The wave is centered initially at the location x_0. The motion of such a solitary wave is depicted by the wave labeled (a) in Figure 3.1. Note that Equation (3.2) can be derived by assuming that a solitary wave of constant shape and speed exists, having the generic form $u(x,t) = f(x - x_0 - \nu t)$. This form is then substituted into the RLW and EW equations (PDEs), and the solutions of the resulting ODEs yield Equation (3.2) for the solitary wave.

Solitary wave solutions of the EW equation exist for all wave speeds $-\infty < \nu = c < \infty$. This is unlike the case of the RLW equation for which solitary wave speeds exist only when $\nu = c + 1 < 0$ or $\nu = c + 1 > 1$, conditions that make k a real nonzero number. Alternately, the forbidden wave speeds are $0 \le \nu \le 1$ for the RLW equation.

The solitary wave solution given by Equation (3.2) features a symmetric wave profile about the path $x = x_0 + \nu t$, along which the wave has the peak amplitude $u_{\text{peak}} = 3c$. To obtain the width λ of this wave at a given fraction of the peak amplitude, consider another parallel path $x = x_0 + \nu t + \lambda/2$, along which the amplitude is also constant and given by $u_\lambda = 3c\ \text{sech}^2[k\lambda/2]$. A normalized amplitude can be defined by $\tilde{u} = u_\lambda/u_{\text{peak}} = \text{sech}^2[k\lambda/2]$, where $\tilde{u}$ is a specified constant (e.g., 1/2). The solution for the previously defined width is then

$$\lambda = \frac{2}{k}\,\text{sech}^{-1}\left[\sqrt{\tilde{u}}\right] = \begin{cases} 4\sqrt{\mu}\,\text{sech}^{-1}\left(\sqrt{\tilde{u}}\right) & \text{for EW,} \\ 4\sqrt{\mu\,\dfrac{c+1}{c}}\,\text{sech}^{-1}\left(\sqrt{\tilde{u}}\right) & \text{for RLW,} \end{cases} \tag{3.4}$$

and the corresponding time duration is $\tau = \lambda/\nu$. For any specified amplitude ratio $\tilde{u}$, solitary waves of the RLW equation exhibit a different width for different wave amplitudes, because the wave number k and width λ depend on both the constants μ and c (one-third wave amplitude). In contrast, solitary waves of the EW equation have a constant width for arbitrary wave amplitudes and speeds, because k and λ depend only on μ. This is the special feature after which the EW equation was named. For the specific case when $\tilde{u} = 1/2$ for the EW equation, the width at the one-half amplitude level is $\lambda = 4\sqrt{\mu}\ \text{sech}^{-1}\left(1/\sqrt{2}\right) = 4\sqrt{\mu}\ \ln\left(1+\sqrt{2}\right) = 3.5255\sqrt{\mu}$. If the width of the solitary wave had been defined alternatively as $\lambda = k^{-1} = 2\sqrt{\mu}$, then λ would correspond to another specific width of the solitary wave measured at the amplitude ratio $\tilde{u} = \text{sech}^2(1/2) = 0.78645$.

Olver [22] has shown that solutions of the EW equation, like those of the RLW equation, have only three conservation laws that can be written in the general form $T_t + X_x = 0$. These laws are the equivalents of the conservation of mass, momentum, and energy in fluid mechanics. Olver showed that the three laws lead directly to three so-called invariants of motion given by

$$C_1 = \int_{-\infty}^{+\infty} u \, \mathrm{d}x \ , \ C_2 = \int_{-\infty}^{+\infty} \left(u^2 + \mu u_x u_x \right) \mathrm{d}x \ , \ C_3 = \int_{-\infty}^{+\infty} u^3 \, \mathrm{d}x \ , \quad (3.5)$$

provided that the integrals converge. These invariants of motion for the EW equation need to be extended for this study. This is done by multiplying the EW equation, $u_t + uu_x - \mu u_{xxt} = 0$, by 1, u, and $u^2 - 2\mu u_{xt}$, and then the resulting three equations can each be expressed in the form $T_t + X_x = 0$, which are summarized as $(u)_t + \left(\frac{1}{2}u^2 - \mu u_{xt} \right)_x = 0$, $\left(u^2 + \mu u_x u_x \right)_t + \left(\frac{2}{3}u^3 - 2\mu u u_{xt} \right)_x = 0$, and $\left(u^3 \right)_t + \left(\frac{3}{4}u^4 - 3\mu u_t u_t - 3\mu u^2 u_{xt} + 3\mu^2 u_{xt} u_{xt} \right)_x = 0$. These three conservation laws can now be integrated easily with respect to x over a finite spatial domain $[x_L, x_U]$ instead of $[-\infty, +\infty]$ to obtain the intermediate results

$$\begin{aligned} \frac{\partial}{\partial t} \int_{x_L}^{x_U} u \, \mathrm{d}x + \frac{1}{2} \left(u_U^2 - u_L^2 \right) &= 0 \ , \\ \frac{\partial}{\partial t} \int_{x_L}^{x_U} \left[u^2 + \mu u_x u_x \right] \mathrm{d}x + \frac{2}{3} \left(u_U^3 - u_L^3 \right) &= 0 \ , \\ \frac{\partial}{\partial t} \int_{x_L}^{x_U} u^3 \, \mathrm{d}x + \frac{3}{4} \left(u_U^4 - u_L^4 \right) &= 0 \ , \end{aligned} \quad (3.6)$$

after simplifications. In these equations, $u_L = u(x_L, t)$ and $u_U = u(x_U, t)$ are time-invariant constants at the domain boundaries. In the simplifications, the terms $[u_{xt}]_{x_L}^{x_U}$, $[uu_{xt}]_{x_L}^{x_U}$, $[u_t u_t]_{x_L}^{x_U}$, $[u^2 u_{xt}]_{x_L}^{x_U}$, and $[u_{xt} u_{xt}]_{x_L}^{x_U}$ are zero at the boundaries because u_L, and u_U are constants thereat. Equation (3.6) can now be integrated with respect to t to yield

$$\begin{aligned} C_1 &= \int_{x_L}^{x_U} u \, \mathrm{d}x + \frac{1}{2} \left(u_U^2 - u_L^2 \right) t \ , \\ C_2 &= \int_{x_L}^{x_U} \left[u^2 + \mu u_x u_x \right] \mathrm{d}x + \frac{2}{3} \left(u_U^3 - u_L^3 \right) t \ , \\ C_3 &= \int_{x_L}^{x_U} u^3 \, \mathrm{d}x + \frac{3}{4} \left(u_U^4 - u_L^4 \right) t \ . \end{aligned} \quad (3.7)$$

These invariants of motion are generalizations of those given by Equation (3.5), extended for the case of a finite length spatial domain when $u(x, t)$ is constant but not necessarily zero at the domain boundaries. The extra terms stem directly from the convection of mass, momentum, and energy into and out of the lower and upper boundaries of the spatial domain. These invariants of motion are equal to the initial ($t = 0$) mass, momentum, and energy inside the domain $[x_L, x_U]$. Note that during

numerical computations that provide solutions to the EW equation, C_1, C_2, and C_3 can be calculated after each successive time step over the entire spatial domain $x_L \leq x \leq x_U$ that contains the wave motion, such that the conservation properties of the numerical algorithm can be monitored and thereby assessed.

3.3 Numerical Solution Procedure

A fairly complete description of the numerical solution procedure for the equal width (EW) equation, $u_t + uu_x - \mu u_{xxt} = 0$, is given in this section. The method consists in essence of numerically integrating this partial differential equation (PDE) forward in time to advance the solution $u(x, t)$ at every node of a spatial grid, with $u(x, t)$ specified at each grid node at some initial time (e.g., $t = 0$) and boundary conditions applied at each time step to specify $u(x, t)$ at the two edge nodes of the grid. The solution of the EW equation on a uniform grid or nonuniform adaptive grid requires discretizations of the spatial derivative terms u_x and u_{xxt}, and these discretizations can lead to a large set of implicit ordinary differential equations (ODEs), one for each node. The resulting large set of stiff ODEs are integrated forward in time by using an advanced ODE solver. This entire procedure is often called the method of lines (MOL) for the sake of brevity. However, the numerical subprocedures in the MOL can vary substantially from one researcher to another; for example, the type and order of the discretization, the type of the ODE solver, the use of a uniform or an adaptive grid, and the method of interpolating $u(x, t)$ and other data from a previous to a new adaptive grid. Our numerical subprocedures and techniques are described herein.

The EW equation, $u_t + uu_x - \mu u_{xxt} = 0$, is a time-dependent PDE in one space dimension. To help describe the solution procedure more concisely, the EW equation is written in functional notation as

$$f\,(u_t, uu_x, u_{xxt}) = 0, \qquad x_{\mathrm{L}} \leq x \leq x_{\mathrm{U}}\,, \tag{3.8}$$

in which u is the dependent variable, x and t are the independent variables, and x_{L} and x_{U} correspond to the lower and upper limits or boundaries on x. As subscripts, x and t denote partial derivatives of the variable.

Initial conditions are specified before the solution procedure can commence. In symbolic form they are stated as $u_0(x) \equiv u(x, t = 0)$ for the finite domain $x_{\mathrm{L}} \leq x \leq x_{\mathrm{U}}$. The boundary conditions at x_{L} and x_{U} are required to determine a solution either analytically or numerically. However, for most problems in which the solitary or other wave propagation occurs well inside the boundaries the solution is negligible at or outside the boundaries or interval $[x_{\mathrm{L}}, x_{\mathrm{U}}]$ during the time span $0 \leq t \leq t_U$ of consideration. Consequently, the boundary conditions at the lower and upper ends of the interval $[x_L, x_U]$, given by $u(x_L, t) = u_L$ and $u(x_U, t) = u_U$, are used, as mentioned in the last section regarding the EW equation.

A nonuniform spatial grid for the numerical solution procedure can be defined by the vector $\underline{x} = [x_1, x_2, \ldots, x_i, \ldots, x_n]^T$, in which x_i is the location of the ith node, n is the total number of grid nodes, the superscript T denotes the transpose of the vector, and the lower and upper nodes x_1 and x_n correspond directly to the computational domain boundaries x_L and x_U. The distance or spacing between adjacent nodes can be defined by $\Delta x_i = x_{i+1} - x_i$, for which $i = 1, 2, \ldots, n-1$. These Δx_i are all constant for a uniform grid, they vary in space for a nonuniform grid, and they also vary in time for an adaptive grid.

The numerical solution is defined by the vector $\underline{u} = [u_1, u_2, \ldots, u_i, \ldots, u_n]^T$, in which u_i is the numerical solution corresponding to grid node x_i. This corresponds to a discrete approximation of the PDEs in terms of space. At the initial time defined as $t = 0$, the solution vector $\underline{u}$ is initialized by using the initial data $u_0(x)$, which is also discretized in space for the node locations $\underline{x}$. The solution vector $\underline{u}$ is advanced in time as the numerical computations proceed, as will be described later. Note also that $\partial u_i/\partial x$ and $\partial u_i/\partial t$ correspond to first-order derivatives at the ith node, and $\partial \underline{u}_i/\partial x$ and $\partial^2 \underline{u}_i/\partial x^2$ are vectors of the first and second derivatives with respect to distance.

The spatial discretizations of the terms u_x and u_{xxt} in the EW equation are obtained by using finite-difference approximations on a nonuniform grid. These finite differences can be expressed at node x_i by

$$\left.\frac{\partial^k u}{\partial x^k}\right|_{x_i} \approx \sum_{j=i-m}^{i+n} c_{i,j,k} u_j, \quad k = 1, 2, \ldots, \tag{3.9}$$

in which normally $m \geq 0$ and $n \geq 0$, the number of grid points used for the derivative is obviously $m+n+1$, which is called the stencil width, and $c_{i,i-m,k}$, $c_{i,i-m+1,k}$, $\ldots$, $c_{i,i+n,k}$ are weights for the ith node for the kth derivative. These weights are specific to the type of derivative, being different for the general cases of forward finite differences when $m < n$, centered finite differences when $m = n$, and backward finite differences when $m > n$. The order of the first and second derivatives is given by $m + n$ when $k = 1$ and 2, respectively, and higher order finite differences correspond directly to a larger stencil width.

Acronyms are used to denote various finite-difference schemes; for example, cfd3p2o denotes a centered finite-difference scheme with $m = n = 1$, using a three-grid-point stencil width of $m + n + 1 = 3$, and the first derivative is of order $m + n = 2$. When centered finite differences cannot be used at and near the lower and upper boundaries, appropriate forward and backward finite differences with the same stencil width and order are used instead.

Two spatial discretization schemes are used in this study, and their influence on the solution accuracy will be highlighted later. For the first scheme, labeled cfd5p4o, the derivative u_x is approximated by a five-point, fourth-order, centered finite difference in space and stored as the vector $\underline{u}_x$, and the derivative u_{xxt} is approximated also by a five-point, fourth-order, centered finite difference in space and stored as the vector $\underline{u}_{xxt}$. For the second scheme, called cfd7p6o, the derivatives u_x and u_{xxt} are both approximated by a seven-point, sixth-order, centered finite difference in space.

All spatial discretizations in this study were generated systematically by using the versatile algorithm called WEIGHTS from Fornberg [8].

The discretization process for the EW equation, or Equation (3.8), on a spatial grid produces a set of equations that can also be expressed in functional form as

$$\begin{aligned} f_1\left(u_1, u_2, \ldots, u_n, \frac{\mathrm{d}u_1}{\mathrm{d}t}, \frac{\mathrm{d}u_2}{\mathrm{d}t}, \ldots, \frac{\mathrm{d}u_n}{\mathrm{d}t}, t\right) &= 0\,, \\ f_2\left(u_1, u_2, \ldots, u_n, \frac{\mathrm{d}u_1}{\mathrm{d}t}, \frac{\mathrm{d}u_2}{\mathrm{d}t}, \ldots, \frac{\mathrm{d}u_n}{\mathrm{d}t}, t\right) &= 0\,, \\ &\vdots \\ f_n\left(u_1, u_2, \ldots, u_n, \frac{\mathrm{d}u_1}{\mathrm{d}t}, \frac{\mathrm{d}u_2}{\mathrm{d}t}, \ldots, \frac{\mathrm{d}u_n}{\mathrm{d}t}, t\right) &= 0\,, \end{aligned} \qquad (3.10)$$

one equation for each grid node. This set of equations can be written concisely in vector notation as

$$\underline{f}\left(\underline{u}, \frac{\mathrm{d}\underline{u}}{\mathrm{d}t}, t\right) = \underline{0}\,, \qquad (3.11)$$

and the initial conditions can be expressed likewise as

$$\underline{f}\left(\underline{u}\big|_{t_0}, \left.\frac{\mathrm{d}\underline{u}}{\mathrm{d}t}\right|_{t_0}, t_0\right) = \underline{0}\,, \qquad (3.12)$$

in which $\underline{u} = [u_1, u_2, \ldots, u_n]$, $\mathrm{d}\underline{u}/\mathrm{d}t = [\mathrm{d}u_1/\mathrm{d}t, \mathrm{d}u_2/\mathrm{d}t, \ldots, \mathrm{d}u_n/\mathrm{d}t]$ and $\underline{f} = [f_1, f_2, \ldots, f_n]$ for the grid $\underline{x} = [x_1, x_2, \ldots, x_n]$. In the spatial discretization process that resulted in Equations (3.10) and (3.11), the partial derivatives u_x and u_{xxt} of the EW equation have been expressed in terms of u and u_t at various grid nodes (according to the type of discretization scheme). Because the only derivatives that remain after the discretization are the partial derivatives $u_t|_i = \partial u_i/\partial t$ at various nodes, these are then considered as, and replaced by, the total derivatives $\mathrm{d}u_i/\mathrm{d}t$, which explains the sudden appearance of the total derivatives in Equation (3.10). This discretization is called a semidiscretization because it occurs in space only and not in time (i.e., the time derivatives remain).

The semidiscretization of the EW equation on a spatial grid, which involves the space and the mixed space and time derivatives u_x and u_{xxt}, results in a fully implicit set of ODEs in terms of u and its derivative $\mathrm{d}u/\mathrm{d}t$, given by Equations (3.10) and (3.11). This system of ODEs is fully implicit because the vector of derivatives $\mathrm{d}\underline{u}/\mathrm{d}t$ is defined implicitly through the arguments of the vector function $\underline{f}$. In other words, the system of ODEs cannot be written in the explicit form $\mathrm{d}\underline{u}/\mathrm{d}t = \underline{f}(\underline{u}, t)$.

For some problems with the EW equation, the functions $f_1, f_2, \ldots, f_n$ in Equations (3.10) and (3.11) each contain one or more derivatives of $u(x, t)$ with respect to time, and all of the equations of the system are therefore differential equations (DEs). For other problems, some of the functions $f_1, f_2, \ldots, f_n$ do not contain any derivative terms and those that do not are algebraic equations (AEs). When the system

given by Equations (3.10) and (3.11) contains a mixture of DEs and AEs, they are called differential-algebraic equations (DAEs), or a system of differential-algebraic equations. DAEs normally stem from the application of boundary conditions that are algebraic in form, making f_1 and f_n algebraic equations.

The first general method for solving a system of DAEs like Equations (3.10) and (3.11) was proposed originally by Gear [13] in 1971. His basic idea was to replace each derivative $\mathrm{d}u_i/\mathrm{d}t$ in Equation (3.11) by a backward differentiation formula (BDF) as an approximation, so that the resulting system of nonlinear algebraic equations can be solved for the vector $\underline{u}_j$ at time level t_j, by using a suitable iterative method such as a Newton iterative procedure. For example, consider the simple implicit Euler formula $\underline{u}_j = \underline{u}_{j-1} + \dfrac{\mathrm{d}\underline{u}_j}{\mathrm{d}t}\left(t_j - t_{j-1}\right)$, which can be rearranged to give the first-order BDF as $\dfrac{\mathrm{d}\underline{u}_j}{\mathrm{d}t} = \dfrac{\underline{u}_j - \underline{u}_{j-1}}{t_j - t_{j-1}}$. Equation (3.11) can then be expressed as $\underline{f}\left(\underline{u}_j, \dfrac{\underline{u}_j - \underline{u}_{j-1}}{t_j - t_{j-1}}, t_j\right) = \underline{0}$, which is a set of nonlinear algebraic equations that can be solved for $\underline{u}_j$ at time level t_j, by using a modified Newton iterative procedure such as that given in the book by Ascher and Petzold [2]. However, a first-order BDF is not sufficiently accurate for the time integration of the EW equation in this study.

Gear's [13] more time-accurate integration procedure involves replacing the first-order implicit Euler formula by the more accurate higher order implicit formula

$$\underline{u}_j = \sum_{\ell=1}^{q} \alpha_\ell \, \underline{u}_{j-\ell} + \beta_0 \frac{\mathrm{d}\underline{u}_j}{\mathrm{d}t}\left(t_j - t_{j-1}\right) , \tag{3.13}$$

in which the integer q is the order of the BDF, and the real constants α_ℓ $(1 \leq \ell \leq q)$ and β_0 have values that depend on the order q. The method of determining the best set of values for these coefficients to optimize time integration accuracy and ensure integration stability, and a tabulated set of the resulting values for the cases of $1 \leq q \leq 6$, are both given in the original paper by Gear [13]. These tables are repeated in the literature and books by, for example, Brenan et al. [6] and Ascher and Petzold [2]. Note that for the first-order case when $q = 1$, the coefficients have values given by $\alpha_1 = \beta_0 = 1$ for the implicit Euler formula. For the second-order case when $q = 2$, the values of the coefficients are given by $\alpha_1 = 4/3$, $\alpha_2 = 1/3$, and $\beta_0 = 2/3$.

When the previous, more general formula is used to replace the vector of time derivatives in Equation (3.11), one then obtains

$$\underline{f}\left(\underline{u}_j, \ \frac{1}{\beta_0(t_j - t_{j-1})} \sum_{\ell=0}^{k} (-\alpha_\ell) \, \underline{u}_{j-\ell}, \ t_j\right) = \underline{0} \, , \tag{3.14}$$

in which ℓ now starts from zero and $\alpha_0 = -1$ is defined to include the $\underline{u}_j$ term. This is a higher order time-accurate set of nonlinear algebraic equations that can be solved for $\underline{u}_j$ at the jth time level by using a Newton iterative procedure. The

BDF of Equation (3.13) is used to obtain Equation (3.14) because the solution values for $\underline{u}_{j-1}$, $\underline{u}_{j-2}$, etc. from earlier time levels are known, whereas for centered and forward finite-difference formulae the solution values of $\underline{u}_{j+1}$, $\underline{u}_{j+2}$, etc. at future time levels are still unknown. The BDF is also used because the implicit nature of $\underline{u}_j$ occurring independently and as part of $\mathrm{d}\underline{u}_j/\mathrm{d}t$ in Equation (3.13) provides important stability properties for the time integration. The coefficients α_ℓ $(1 \leq \ell \leq q)$ and β_0 are determined in essence by a procedure that optimizes integration accuracy while ensuring good integration stability.

The previous discussion about solving DAEs according to Gear's original ideas and Equations (3.13) and (3.14) implies that the coefficients α_ℓ $(1 \leq \ell \leq q)$ and β_0 are all constants for a given order q of the BDF. This approach gives good solutions for DAEs when the solutions are computed with equal-sized time steps, and also when solutions for $\underline{u}$ have relatively smooth temporal variations, conditions for which the original time-integration method was designed. However, for solutions that vary relatively rapidly in time (i.e., high temporal gradients and curvatures in $\underline{u}$), the time steps should be reduced considerably and frequently to accurately capture any high temporal gradient and curvature phenomena, and in such cases the method of fixed coefficients can result in reduced solution stability. The remedy for this problem is to incorporate variable coefficients that depend inherently on the variable time steps which in turn depend on temporal solution gradients. Although the method of variable coefficients is the most stable implementation of the BDF methods, we do not recommend its implementation in the fullest form because of the disadvantage involving computational inefficiency. The full approach requires numerous, computationally laborious, Jacobian matrix evaluations or updates with time-step intervals of variable size. See Brenan et al. [6] for more details.

The scheme for integrating Equation (3.14) forward in time, one time step after another, should have the capability of using variable time steps to obtain accurate solutions to the EW equation, for the reasons just mentioned. The scheme should also have the capability of using variable orders of the BDF, for the following reasons. The accuracy of the time integration can be improved by using a higher order BDF (e.g., $q = 5$ instead of 3). However, the time-integration accuracy will be degraded temporarily during the first few time steps. During the first time step Δt_1 from $j = 1$ to 2, the integration must start with the order $q = 1$ of the BDF to determine $\underline{u}_2$, because only $\underline{u}_1$ is known. For the second time step Δt_2 from $j = 2$ to 3, one can then use $q = 2$, and so on and so forth, until $q = 5$ can be used at the fifth and subsequent time steps. The time-integration accuracy is also temporarily degraded when adaptive gridding is implemented at the end of any time step. This happens because the grid nodes are suddenly rearranged in the adaptive grid process, the numerical results from the previous grid are then interpolated onto the new grid, and most information related to the previous grid and earlier time integration process becomes obsolete. Hence, the time integration needs to be re-started with order $q = 1$, then $q = 2$, etc. until $q = 5$. These temporary degradations in the time integration can be partially overcome or minimized by using very small time steps when the order q is reduced to less than its maximum value (5 in the previous example). These remarks illustrate

that the integration method that will yield an accurate solution of the EW equation must solve a large set of implicit algebraic equations (one for each grid node) on an adaptive grid and include special features involving variable time stepping, variable BDF order sequencing and variable BDF coefficients.

The fully implicit DAE system given by Equation (3.11), which is approximated by the fully implicit set of nonlinear algebraic equations given by Equation (3.14), is integrated numerically in time in this study by using an advanced DAE solver called DASSL from Petzold [24]. This versatile solver has special features for solving stiff DAEs. It can be used to time integrate either ODEs or DAEs, such that different problems governed by the EW equation which involve differential or algebraic boundary conditions can be solved easily with minor computer-code modifications. This solver can be used readily with either uniform or adaptive grids. The integration time steps are changed automatically by the solver to capture high temporal gradient and curvature features of the solution and simultaneously maintain solution accuracy and integration stability. A combination of fixed and variable coefficients for α_ℓ and β_0, called the fixed leading-coefficient method by Ascher and Petzold [2], is used in DASSL. This is a compromise between the less stable but computationally efficient fixed coefficient approach and the more stable but computationally expensive variable coefficient approach, which results in less integration stability but still retains good computational efficiency by needing fewer Jacobian evaluations compared to the fully variable coefficient approach. The order q of the BDF is changed by the solver when necessary, and q is varied from 1 to 5 by the DASSL solver.

The solver DASSL includes a subroutine called RES that computes the residual of the system of equations resulting from the semidiscretization of the EW equation (or other similar PDEs). The primary inputs to RES are the independent variable t, dependent variable vector $\underline{u}$, and derivative vector $\underline{u}_t$, so that the residual vector having the form

$$R\left(\underline{u}, \underline{u}_t, t\right) = \underline{u}_t + \underline{u}^T \cdot \underline{u}_x - \mu\, \underline{u}_{xxt} \tag{3.15}$$

can be computed, with $\underline{u}_x$ and $\underline{u}_{xxt}$ both known from previous finite-difference approximations. The purpose of the solver DASSL is to compute the dependent variable vector $\underline{u}$ by using a modified Newton method such that the residual vector $R(\underline{u}, \underline{u}_t, t)$ approaches zero and Equations (3.11) and (3.14) are satisfied.

An adaptive grid is implemented in this study to facilitate the resolution of high spatial gradients and curvatures to reduce truncation errors thereat in the solutions of the EW equation. To help describe the adaptive grid scheme, consider the vector of grid nodes $\underline{x} = [x_1, x_2, \ldots, x_n]$ with $x_{i-1} < x_i < x_{i+1}$ and having known locations at the initial time level $t_1 = 0$ or some later time level t_j. Consider the corresponding solution $\underline{u} = [u_1, u_2, \ldots, u_n]$ as known initially at time t_1 or just computed at some later time level t_j. The node locations corresponding to the solution may not be optimal, and a method of moving these nodes to more optimal locations is a requirement of an adaptive grid.

The movement of grid nodes is based on the discrete function defined by

$$
\begin{aligned}
s_i &= \int_{x_1}^{x_i} (\alpha + a_1 b_1 + a_2 b_2)^{\gamma} \, \mathrm{d}x, \quad 1 \le i \le n \, , \\
b_1 &= \min \left(\beta, \, \left| u_x(x, t_j) \right| \right) \, , \\
b_2 &= \min \left(\beta, \, \left| u_{xx}(x, t_j) \right| \right) \, ,
\end{aligned}
\tag{3.16}
$$

and the values of s_i, one for each node at time level t_j, are computed by the trapezoidal rule. For this discrete function, $s_1 = 0$, $s_{i-1} < s_i < s_{i+1}$, and s_n is the maximum value. A continuous piece-wise linear function from s_i can be constructed when required.

The parameters a_1, a_2, α, and β in Equation (3.16) are user-defined positive constants for a particular solution to the EW equation. These parameters are normally adjusted or *tuned* for each EW solution (normally by solving part of a problem a few times), such that the adaptive grid will help produce a *good* solution to the EW equation for a specific problem, while using a *reasonable* number of grid nodes and a *moderate* computational effort, which is a trial-and-error process that involves *subjective* assessment. The values of a_1, a_2, α, β, and γ should be chosen to help equidistribute the truncation error throughout the grid, that is the truncation errors in regions of high gradients and curvatures with clustered nodes are roughly the same as those in regions of low gradients and curvatures with sparsely spaced nodes.

The values of s_i in the adaptive grid process depend on the first and second space derivatives, u_x and u_{xx}, which are approximated by finite differences (e.g., cfd5p4o and cfd7p6o) using the subroutine `WEIGHTS` of Fornberg [8]. In some previous studies, u_x and u_{xx} were obtained by differentiation of a cubic spline that was fitted to the discrete data u_i vs. x_i. This cubic-spline approximation was used instead of finite-difference approximations to help provide more smoothly behaved derivatives, which yield a more gradual or smoother variation in successive node spacings in the adaptive grid. However, this minor advantage is not worth the extra computational effort (i.e., solving a tridiagonal matrix system for the spline coefficients) during the procedure of solving the EW equation.

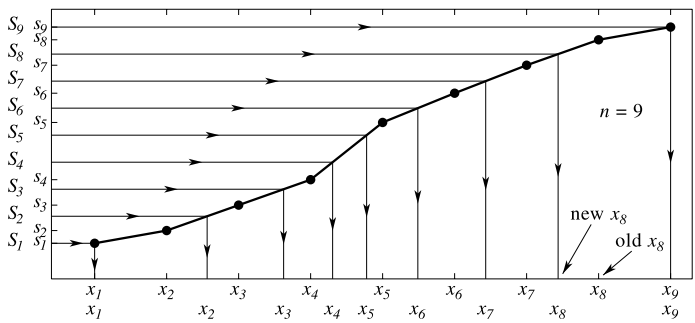


FIGURE 3.3
Grid adaptation using an equidistribution principle.

The previous locations of the grid nodes help define the piecewise linear curve $s(x_i)$ defined by Equation (3.16), which is depicted in Figure 3.3 for only $n = 9$ nodes for illustration purposes, in the form of a one-to-one mapping or projection $\underline{x} \rightarrow \underline{s}$. The new locations of the nodes are obtained by defining the equidistributed set $S_i = s_1 + (s_n - s_1)(i-1)/(n-1)$, or more simply by $S_i = s_n(i-1)/(n-1)$ because $s_1 = 0$. These S_i are shown equidistributed along the vertical axis in Figure 3.3. The inverse one-to-one mapping $\underline{S} \rightarrow \underline{x}$ provides the new locations of the nodes, as illustrated in Figure 3.3. The implementation of equal increments in the inverse mapping procedure, by using the equidistribution constant $\Delta S = S_i - S_{i-1} = s_n/(n-1)$, with $1 \leq i \leq n$, defines the essence of the equidistribution principle. Note that during grid adaptations the first and last nodes x_1 and x_n automatically remain fixed at the domain boundaries x_L and x_U.

Basic information that is relevant to understanding and specifying values for a_1, a_2, α, β, and γ are now provided. The parameters a_1 and a_2 are generally set either to zero or unity. For the combination $a_1 = 1$ and $a_2 = 0$, the adaptive grid movement is based in essence on the magnitude of the solution gradient, that is $|u_x(x,t)|$, whereas the reverse combination of $a_1 = 0$ and $a_2 = 1$ bases the adaptive grid movement in essence on the magnitude of the solution curvature that is typified by $|u_{xx}(x,t)|$. In this investigation, we normally set $a_1 = a_2 = 1$ such that the adaptive movement of the grid nodes is based on a combination of both the solution gradient and curvature.

The value of γ has historically been set equal to 1/2. This most likely stems from the earliest work in which the grid adaptation was based on the length of the curve $u(x,t_j)$ vs. x, which is related to the solution gradient, so that the discrete function was defined originally by $s_i = \displaystyle\int_{x_1}^{x_i} \sqrt{1 + u_x^2(x,t_j)}\, \mathrm{d}x$. In more recent publications, and also in this study, the values of b_1 and b_2 in Equation (3.16) are not squared, so retaining a value of γ equal to 1/2 no longer seems rational. Nonetheless, we set $\gamma = 1/2$ in this study because the use of other more appropriate values has not been explored.

For the following discussion about α and β, consider an important feature of the equidistribution principle. Although $\Delta S = s_n/(n-1)$ exactly, it is also given approximately by

$$\Delta S \approx \int_{x_i^{\text{new}}}^{x_{i+1}^{\text{new}}} (\alpha + a_1 b_1 + a_2 b_2)^\gamma \, \mathrm{d}x \approx (\alpha + a_1 b_1 + a_2 b_2)^\gamma \, \Delta x_i \tag{3.17}$$

in terms of the integrand and grid node spacing Δx_i. When the discrete solution for $u(x,t_j)$ is constant or devoid of waves in some region or regions of space (e.g., near boundaries or between solitary waves), that is when $u_x = 0$ and $u_{xx} = 0$ in such regions at the jth time level, then one can deduce from Equation (3.17) and the definitions of b_1 and b_2 from Equation (3.16) that the node spacings are a maximum for such conditions and given by $\Delta x_{\max} \approx \Delta S/\alpha^\gamma$. The parameter α, therefore, plays a fundamental role in controlling the maximum node spacing.

The maximum node spacing can be specified by $\Delta x_{\max} = g_{\max}(x_U - x_L)/(n-1)$ for many problems, which defines this node spacing at a value of $g_{\max}$ times the

average node spacing, where $g_{\max} \approx 3$ to 8 is reasonable based on our experience. Hence, we obtain

$$\alpha \approx \left(\frac{\Delta S}{\Delta x_{\max}} \right)^{1/\gamma} \approx \left(\frac{1}{g_{\max}} \frac{s_n - s_1}{x_U - x_L} \right)^{1/\gamma} , \tag{3.18}$$

which shows in essence that α^γ is a fraction of the overall slope $(s_n - s_1)/(x_U - x_L)$ of the piecewise linear curve $s(x)$, because $g_{\max} > 1$. This result also shows clearly that the parameter α is problem dependent by means of s_n, which is the total integral evaluated from x_1 to x_n in Equation (3.16). Of equal importance is that α is independent of the total number of grid nodes (n). Hence, once the value of α is known for a particular problem, it need not be changed when the same problem is re-solved using a different number of nodes.

One approach of specifying α for a new problem is to guess the value of s_n and estimate $\alpha \approx [s_n/g_{\max}(x_U - x_L)]^{1/\gamma}$. A better approach is to determine s_n directly by using the discretized initial conditions at time level t_1, which in many cases is a good estimation for the entire problem. A finer tuning of the value of α requires the study of numerical results of computations to determine if the final numerical solution can be judged satisfactory. Changes to the value of α can also be invoked to reduce numerical errors in the predicted invariants of motion of the conservation laws, if these laws are known.

For the discussion about β, let this parameter be set initially to the maximum magnitude of $u_x(x, t_j)$ when $a_1 = 1$ and $a_2 = 0$, the maximum magnitude of $u_{xx}(x, t_j)$ when $a_1 = 0$ and $a_2 = 1$, or the larger value of the maximum values of $|u_x(x_i, t_j)|$ and $|u_{xx}(x_i, t_j)|$ when $a_1 = a_2 = 1$. If these maximum values are not known before a problem is solved numerically, then a suitable value of β might be guessed. When $(\alpha + a_1 b_1 + a_2 b_2)^\gamma$ is a maximum, then the node spacing is a minimum, and one can deduce from Equation (3.17) and the definitions of b_1 and b_2 from Equation (3.16) that this minimum spacing is given by $\Delta x_{\min} \approx \Delta S/\,(\alpha + \kappa\beta)^\gamma$, in which $\kappa = 1$ for the first two cases $(a_1 = 0, a_2 = 1; a_1 = 1, a_2 = 0)$ and $1 \le \kappa \le 2$ for the last case $(a_1 = a_2 = 1)$. In this last and more complex case, $\kappa = 1$ might occur when $|u_x(x, t_j)|$ and $|u_{xx}(x, t_j)|$ are a zero and a maximum at the same spatial location, or conversely $\kappa = 2$ only if $|u_x(x, t_j)|$ and $|u_{xx}(x, t_j)|$ are both a maximum at the same spatial location (an impossibility). More generally, the value of κ is somewhat larger than but near unity.

The result $\Delta x_{\min} \approx \Delta S/\,(\alpha + \kappa\beta)^\gamma$ illustrates that $|u_x(x, t_j)|$, $|u_{xx}(x, t_j)|$, or β, which is normally much larger than α, plays a strong role in controlling the minimum node spacing. For example, when β is set to a value larger than both $|u_x(x, t_j)|$ and $|u_{xx}(x, t_j)|$, then these magnitudes help directly in concentrating nodes in regions of large solution gradients and curvatures, which produce one or more separated regions of minimum node separation. When β is set to a value somewhat smaller than the maximum values of both $|u_x(x, t_j)|$ and $|u_{xx}(x, t_j)|$, then some gradient and curvature values are said to be *clipped*, by means of the expressions for b_1 and b_2 in Equation (3.16), however some nodes are still concentrated in regions of large solution gradients and curvatures, but the concentration is somewhat less than the

previous unclipped case for a larger value of β. If β is set to zero, then a uniform grid will occur with an average node spacing $(x_U - x_L)/(n-1)$. These comments help illustrate that the so-called *clipping* parameter β plays a fundamental role in controlling the minimum node spacing, such that solution gradients and curvatures that are large are well resolved by node clustering and those that are small are also well resolved by means of sparsely spaced nodes. Consequently, the values of α and β control the node spacings which in turn help make the truncation error more equidistributed throughout the grid.

The minimum node spacing can be specified by $\Delta x_{\min} = g_{\min}(x_U - x_L)/(n-1)$ for many problems, which sets the minimum node spacing at a value of $g_{\min}$ times the average node spacing, where $g_{\min} \approx 1/2$ to 1/9 is reasonable based on our experience. Hence, we obtain

$$\beta \approx \frac{1}{\kappa}\left[\left(\frac{\Delta S}{\Delta x_{\min}}\right)^{1/\gamma} - \alpha\right] \approx \frac{1}{\kappa}\left(\frac{\Delta S}{\Delta x_{\min}}\right)^{1/\gamma} \tag{3.19}$$

or

$$\beta \approx \frac{1}{\kappa}\left(\frac{1}{g_{\min}^{1/\gamma}} - \frac{1}{g_{\max}^{1/\gamma}}\right)\left(\frac{s_n - s_1}{x_U - x_L}\right)^{1/\gamma} \approx \frac{1}{\kappa}\left(\frac{1}{g_{\min}}\,\frac{s_n - s_1}{x_U - x_L}\right)^{1/\gamma}, \tag{3.20}$$

which shows in essence that $(\kappa\beta)^\gamma$ is a multiple of the overall slope $(s_n - s_1)/(x_U - x_L)$ of the piecewise linear curve $s(x)$, because $g_{\min} < 1$. These results illustrate that the parameter β, like α, is problem dependent via s_n but independent of the total number of nodes.

The relationships for α and β can be combined to give $\Delta x_{\max}/\Delta x_{\min} \approx (1 + \kappa\beta/\alpha)^\gamma$ and $\kappa\beta \approx \alpha\left(g_{\max}^{1/\gamma}/g_{\min}^{1/\gamma} - 1\right)$, provided that β is set sufficiently small so that some clipping occurs. These results help illustrate the inter-relationships that exist between the parameters α and β, $\Delta x_{\max}$ and $\Delta x_{\min}$, and $g_{\max}$ and $g_{\min}$. If we take $\gamma = 1/2$, $g_{\max} = 4$, and $g_{\min} = 1/5$, we can then estimate $\kappa\beta \approx 399\alpha$, which illustrates that $\kappa\beta/\alpha \gg 1$ or $\kappa\beta \gg \alpha$.

One approach of specifying β for a new problem is to start from a previously guessed or calculated value of α and estimate β by means of Equation (3.20) or $\kappa\beta \approx \alpha\left(g_{\max}^{1/\gamma}/g_{\min}^{1/\gamma} - 1\right)$. On one hand, if solution gradients and curvatures are relatively large, then the value of β will be less than the maximum values of $|u_x(x_i, t_j)|$ and $|u_{xx}(x_i, t_j)|$, clipping of the gradient and/or curvature will occur, and the maximum and minimum node spacings will correspond closely to those expected from the specification of the values of $g_{\max}$ and $g_{\min}$. On the other hand, if solution gradients and curvatures are relatively small, then the value of β will be larger than the maximum values of $|u_x(x_i, t_j)|$ and $|u_{xx}(x_i, t_j)|$, clipping of both the gradient and curvature will not occur, and the maximum node spacing will not correspond closely to that expected from the specification of the value of $g_{\min}$. The nodes will be less concentrated than expected in regions of higher gradients and curvatures, but a higher node concentration will probably not be needed to obtain an accurate solution to the EW equation. A finer tuning of the value of β, like the tuning of α, requires the study of some numerical

solutions to determine if the final solution can be judged satisfactory. Changes to the value of β may also be invoked to help reduce numerical errors in the predicted invariants of motion from the conservation laws, if these laws are known.

The adaptive grid technique used in this study is based on the equidistribution principle described earlier. The nodes are held fixed during each time step. They are moved only after a specified time interval denoted by Δt_{grid}, which may include a large number of very small time steps. This is called the *static* grid method because the nodes are not moved simultaneously as the solution is computed, in contrast to the *dynamic* grid method. The static adaptive node movement is illustrated in Figure 3.4, where the grid adaptation is depicted after every four time steps. Although the grid adaptation is done at the end of a time step, the node shifts are shown to occur over an entire time step, but this is done for illustration purposes only.

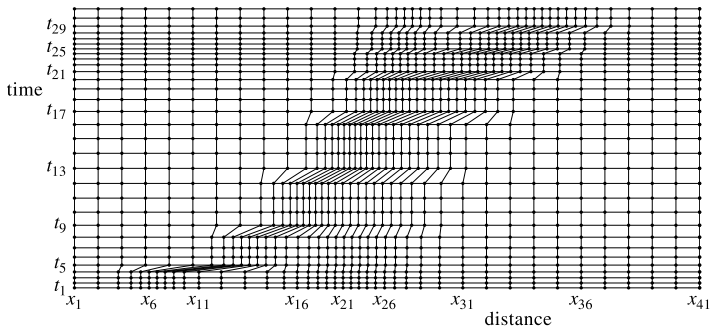


FIGURE 3.4
Adaptive grid movement after every four time steps.

The grid adaptation is done after a reasonably short preset time increment that may include a few or a large number of time steps, depending somewhat on the solution behavior. During this preset time increment Δt_{grid}, the solution $u(x, t)$ at the grid nodes should not change by more than a few percent, or a solitary wave should propagate only a short distance of a few nodes or less. In other words, the moving parts of solutions with high gradients and curvatures should not outrun their region of clustered nodes, if these high gradients and curvatures are to be properly resolved.

When the grid is adapted and updated, the solution from the previous grid needs to be mapped onto the new grid. In this study this mapping or interpolation is done by means of the quintic polynomial $u(x, t_j) = \sum_{k=0}^{5} b_k x^k$ between two successive nodes x_i and x_{i+1}, and the polynomial coefficients b_i are determined from our already known values of u_i, $u_x|_i$, and $u_{xx}|_i$ at node x_i and u_{i+1}, $u_x|_{i+1}$, and $u_{xx}|_{i+1}$ at node x_{i+1}. This interpolation is fairly consistent, but not entirely consistent, with our five- and seven-point finite-difference schemes cfd5p4o and cfd7p6o of fourth and sixth order, respectively. The quintic polynomial $u = \sum_{k=0}^{5} b_k x^k$ can be written in the

convenient prepackaged form

$$
\begin{aligned}
u \;=\; & u_i\,(1-\eta)^3\left(1+3\eta+6\eta^2\right)+u_{i+1}(1-\xi)^3\left(1+3\xi+6\xi^2\right) \\
& +\,u_x|_i\,(1-\eta)^3\eta(1+3\eta)\Delta x_i-u_x|_{i+1}\,(1-\xi)^3\xi(1+3\xi)\Delta x_i \\
& +\,\frac{1}{2}\,u_{xx}|_i\,(1-\eta)^3\eta^2\Delta x_i^2+\frac{1}{2}\,u_{xx}|_{i+1}\,(1-\xi)^3\xi^2\Delta x_i^2\,, \qquad (3.21)
\end{aligned}
$$

so that all coefficients are expressed directly in terms of u and its first and second space derivatives at adjacent nodes x_i and x_{i+1}. In this equation, the normalized distances $\eta = (x-x_i)/\Delta x_i$ and $\xi = 1-\eta$, and $\Delta x_i = x_{i+1}-x_i$. In previous research papers the interpolation was done with cubic and quintic splines. The cubic spline interpolation is simply insufficiently accurate for this study, as it was also for the previous studies that used high-order finite-difference schemes. The quintic spline may be sufficiently accurate but one disadvantage is that it alters the first and second derivatives of u of the computed solution to make u_{xxx} and u_{xxxx} continuous. A further disadvantage of cubic and quintic splines is the extra computational effort in solving tridiagonal and pentadiagonal matrix systems for the spline coefficients, respectively, in contrast to using a prepackaged interpolant like that given by Equation (3.21), for which the derivatives u_x and u_{xx} are readily available.

When the time integration is restarted after a grid adaptation, the solver `DASSL` is re-initialized by using the parameter `INFO`, which is set to a value of zero so that the order q of the BDF restarts from unity, because the previous obsolete solution values and Jacobian entries correspond to the previous rather than the new node locations. This procedure gave good results, so the more sophisticated, accurate, and computationally intensive method of remapping $u(x,t)$ and all related information from the previous q time levels and previous grid to the new grid were not implemented.

Numerical solutions for the EW equation always begin on a nonuniform grid. The initial conditions at time $t = 0$ are initially discretized on a uniform grid, then this uniform grid is adapted to a nonuniform grid by using these initial conditions. The initial conditions are then re-interpolated onto the nonuniform grid. This grid should be adapted once more to a new nonuniform grid, which will differ somewhat from the previous one because the mapping of $\underline{x} \rightarrow \underline{s}$ uses approximate trapezoidal integration and the mapping $\underline{S} \rightarrow \underline{x}$ uses a piecewise linear curve. The initial adaptive grid is then prepared for the numerical computations.

One important feature of the previously described adaptive grid technique is that this scheme is independent of, or uncoupled from, the PDE discretization and time integration procedures, because the grid adaptation is done between time steps when the discretization and integration procedures are halted. Hence, the adaptive grid algorithm is problem independent, and it can therefore be coded once for all problems. In this study, the adaptive grid subroutine called `AGE` is used, which was obtained from Saucez et al. [28]. Similar algorithms such as those by White [31], Sanz-Serna and Christie [27], and Revilla [25] can be implemented easily from the ideas presented in their papers.

It is important to realize that the use of more nodes (which results in a larger set of DAEs) with a larger variation in the node spacings, increases the computational

effort and related computer run times. Furthermore, closely spaced nodes make the set of DAEs computationally *stiff* and the problem solution by means of the solver DASSL requires an increased number of smaller time steps to produce a full solution, which leads to longer central processor run times. See the books by Brenan et al. [6] and Schiesser [30] for a detailed definition and explanation of the stiffness of DAEs, which is directly related to the separation of the eigenvalues of the Jacobian matrix of the DAE system, which in turn is adversely affected by large variations in node spacings. Because node spacings vary widely for problems solved numerically in this study, the double precision version of the stiff DAE solver DASSL is used to integrate Equation (3.14). Also, an approximate Jacobian is computed internally by this solver using finite differences. Small absolute and relative tolerances on local time steps are imposed by setting the values of ATOL and RTOL in DASSL to the small value 10^{-8}, and this suppresses the time integration errors to negligibly small values.

The banded structure of the Jacobian matrix is an important consideration for the numerical computations done in this study. The two spatial discretization schemes cfd5p4o and cfd7p6o used in the fully implicit DAE time integration by the DASSL solver lead to septagonal and endectagonal banded Jacobian matrices, respectively. Hence, for cfd5p4o and cfd7p6o, the half-bandwidths ML and MU in the DASSL solver are each set to 3 and 5, respectively, resulting in the Jacobian bandwidth defined by ML + MU + 1 to have values of 7 and 11, respectively. Note that higher order discretizations produce Jacobian matrices with larger bandwidths, which in turn increase the computational effort to produce the numerical solution. However, a lower order finite-difference scheme coupled to a grid with lots of nodes will yield a reasonably accurate solution, but the computations will be inefficient.

3.4 Numerical Results and Discussion

3.4.1 Single Solitary Waves

The EW equation, $u_t + uu_x - \mu u_{xxt} = 0$, is solved in this section to predict the motion of a single solitary wave in space and time. This problem is solved for the case of the parameter $\mu = 1/16$, by using the MOL whose subprocedures were outlined previously. Many solutions are obtained using uniform and adaptive grids with different numbers of grid nodes, and also using two different finite-difference schemes. The exact solution for this problem is known and given in general by Equation (3.2) as $u(x,t) = 3\,c\,\mathrm{sech}^2[k\,(x - x_0 - vt)]$. The wave number is $k = 1/\sqrt{4\mu} = 2$, the wave speed is $v = c = 1/10$, the peak amplitude is $u_{\mathrm{peak}} = 3c = 3/10$, and the wave is centered at $x_0 = 20$ at $t = 0$. Many exact and numerical results are compared for this benchmark problem to assess various parts of the solution procedure.

The initial conditions for the numerical computations are specified by using the exact solution at time $t = t_1 = 0$ as $u(x,0) = 3\,c\,\mathrm{sech}^2[k\,(x - x_0)]$ on the interval

$[x_L = 0, x_U = 45]$. The numerical solution is computed for times varying from $t = 0$ to 60 with various spatial discretization schemes on uniform and adaptive grids with the number of nodes varying from 51 to 401. During the time interval 0 to 60, the solitary wave is always far from the grid boundaries, so the Dirichlet boundary conditions $u(x_L, t) = u_L = 0$ and $u(x_U, t) = u_U = 0$ are applied because they are sufficiently accurate for the current problem.

The exact solution is given in Figure 3.5 for the spatial and temporal intervals $0 \le x \le 45$ and $0 \le t \le 60$, respectively. This problem becomes challenging to solve numerically on uniform and adaptive grids when the number of nodes is reduced sufficiently such that the solution gradients and curvatures become difficult to resolve accurately.

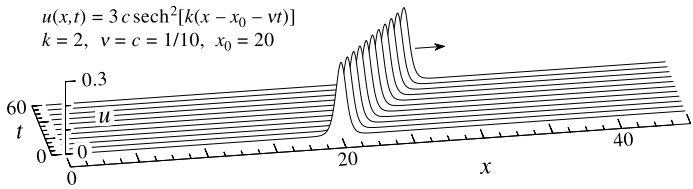


FIGURE 3.5
Solitary wave motion calculated with the exact solution.

The MOL solution of the EW equation for our problem with $\mu = 1/16$ gives numerical results for the motion of a single solitary wave with a nearly constant speed, peak amplitude, and shape. The discrete solution $u(x_i, t_j)$ is normally in close agreement with the exact solution shown in Figure 3.5, if the grid nodes are sufficiently numerous, such that the exact and numerical solutions overlap and are not readily distinguishable. Hence, the differences between the exact and numerical solutions, $u_i^{\text{exact}} - u_i^{\text{num}}$, as a function of distance at a given time should be used, and these errors are shown in Figure 3.6. Two piecewise linear solutions are displayed for the cases of $n = 101$ grid nodes, time $t = 60$, an adaptive grid with parameters $a_1 = a_2 = 1$, $\alpha = 0.01$, $\beta > 2.5$, grid adaptation implemented after the preset time interval $\Delta t_{\text{grid}} = 1$, and for the finite-difference schemes cfd5p4o and cfd7p6o defined earlier.

Both of the discretization schemes in the MOL give fairly accurate and acceptable solutions, with errors relative to the peak amplitude $u_{\text{peak}} = 3/10$ that are less than 2%. However, the finite-difference scheme cfd7p6o with the larger stencil and higher order gives more accurate results, as might be expected, with errors less than 0.3% (vs. 2% for the smaller stencil). If more grid nodes are used, the errors will then decrease. However, these numerical solutions were presented mainly to show that the scheme cfd7p6o with the wider stencil gives more accurate results and is a better choice for solving such problems, and increasing the number of nodes does not change this conclusion. Other results given later will further justify this remark. Our numerical solutions are also presented to show that these results are more accurate than those published earlier by Gardner et al. [10] for a much easier problem with $\mu = 1$, for which the wave is much smoother spatially in that it is four times wider. The errors

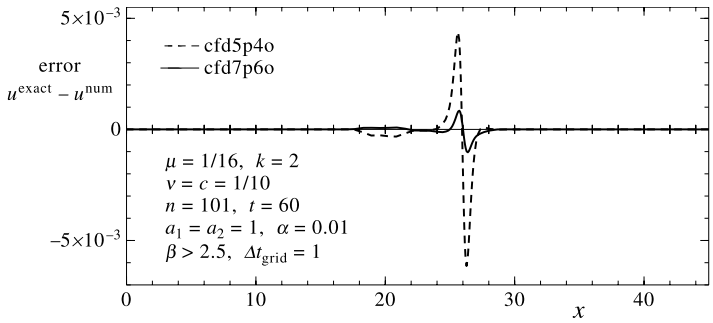


FIGURE 3.6
Solution errors for cfd5p4o and cfd7p6o.

for the case of our adaptive grid with 101 nodes are three times smaller than those of Gardner et al. [10] for the case of a uniform grid with 1001 nodes.

For our problem with $\mu = 1/16$ and $k = 2$ for a single solitary wave, any value of $\beta > 2.437$ yields the same results for the following reasons. The maximum value of $|u_x(x,t)|$ is 0.4619 at $x = x_0 \pm 0.3292$ (at which $u = 0.2\overline{0}$ and $u_{xx} = 0$), where the bar over the last digit of a number implies that this digit continues indefinitely. The maximum value of $|u_{xx}(x,t)|$ is $2.4\overline{0}$ at $x = x_0$ (where $u = 0.3\overline{0}$ and $u_x = 0$). The maximum value of $|u_x(x,t)| + |u_{xx}(x,t)|$ is 2.437 at $x = x_0 \pm 0.03111$. Values of β set larger than this last maximum do not invoke any clipping by means of the expressions for b_1 and b_2 given by Equation (3.16). Clipping will occur for lower β values, that is when $\beta < 2.437$, but this clipping is unnecessary because the grid nodes are clustered appropriately in regions of high gradients and curvatures, and they are not overly crowded anywhere in the numerical solutions in this section. In other words, the minimum node spacing was not overly small in regions of maximum gradients and curvatures such that a stiff set of DAEs became difficult to solve. For the current problem, $g_{\max} \approx 1.452$ and $g_{\min} \approx 1/10.61$, meaning that the maximum and minimum node spacings are about 3/2 times larger and 11 times smaller than the average node spacing.

The MOL solutions of the EW equation for the same problem using a coarse uniform grid and an adaptive grid with only 101 nodes are shown in Figures 3.7 and 3.8. Each set of solutions for $u(x_i, t_j)$ pertains to time levels t_j of 0, 20, 40, and 60, the solutions are given on the small spatial interval [15, 33] of the whole interval [0, 45] for clarity, and other problem and computational data are included as an inset in these figures. The node locations for the solutions at various times are shown in the lower part of each figure. For both sets of solutions the discrete data are connected by straight lines, which is a low-order solution reconstruction or interpolation. For the first set of results in Figure 3.7 the discrete data is also connected by a smooth curve using the quintic interpolation given by Equation (3.21). The piecewise linear solutions appear rather kinky for the case of the uniform grid and quite smooth for the nonuniform grid, illustrating that the adaptive grid that concentrates nodes in regions of higher

gradients and curvatures helps resolve or capture such steep solution gradients and curvatures more effectively. This is the primary reason for comparing these two sets of solutions, and other forthcoming results will also show that the adaptive grid yields superior solutions.

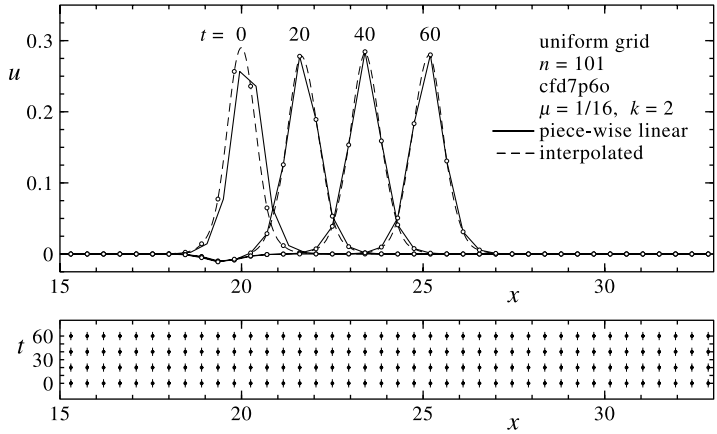


FIGURE 3.7
Single solitary wave solutions using a uniform grid.

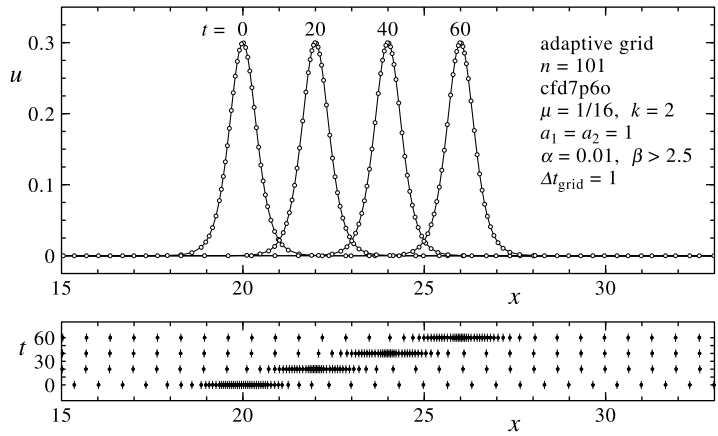


FIGURE 3.8
Single solitary wave solutions using an adaptive grid.

The MOL solution for the solitary wave on the uniform grid with only 101 nodes is quite inaccurate based on the results in Figures 3.7 and 3.8. For example, the peak amplitude of the wave on the uniform grid is less than the exact value of 3/10, the wave is propagating too slowly in that the center of this wave has moved to only $x \approx 25$ at time $t = 60$ instead of the exact value $x = 26$, and the dip below zero

in the solution between $x = 18$ to 21 should not occur. Solutions on a uniform grid with more nodes would, of course, be more accurate.

Although the piecewise linear reconstruction of the solutions in Figures 3.7 and 3.8 is a convenient stratagem to illustrate the benefits of using an adaptive grid in comparison to a uniform grid, this stratagem is unfair. The finite-difference scheme cfd7p6o, or the quintic equation given by Equation (3.21) for interpolating solutions from a previous to new adaptive grids, should be used instead to reconstruct a smooth solution for $u(x, t)$ between nodes. When this is done the solutions for both cases are no longer piecewise linear with kinkiness, especially apparent for the case of the coarse uniform grid, but they will be smooth as illustrated by the interpolated results shown in Figure 3.7. These interpolated sets of solutions represent much better the solutions obtained by the MOL. Nevertheless, the solution computed on the adaptive grid is still more accurate than that on the uniform grid, as we can see qualitatively from the results in Figures 3.7 and 3.8. However, the accuracy cannot be easily determined quantitatively from the results in these figures, so other results are now introduced.

The peak amplitude and its trajectory in space and time are given by the exact solution as $u_{\text{peak}}^{\text{exact}} = 3c = 3/10$ and $x_{\text{peak}}^{\text{exact}} = x_0 + vt = 20 + t/10$, respectively, because $v = c = 1/10$ and $x_0 = 20$ for our problem. The corresponding values of $u_{\text{peak}}^{\text{num}}$ and $x_{\text{peak}}^{\text{num}}$ from the MOL solution can be obtained by searching through the discrete data $u(x_i, t_j)$ at time level $t_j = 60$ for the maximum value and recording its corresponding node location. Such simple results for the peak amplitude and location are simply judged as being too crude for this investigation. Instead, an interpolation of $u(x_i, t_j)$ between grid nodes is done by using Equation (3.21), and the maximum (peak) amplitude and its spatial location are obtained by using a well-known iterative procedure due to Brent [7]. These numerical results are presented in Table 3.1 for the case of $\mu = 1/16$ and the cases of an adaptive grid ($a_1 = a_2 = 1$, $\alpha = 0.01$, $\beta > 2.5$), two finite-difference schemes cfd5p4o and cfd7p6o, and different numbers of nodes n equal to 51, 101, 201, and 401.

Table 3.1 Peak Amplitude and Its Location

	$u_{\text{peak}}^{\text{num}}$		$x_{\text{peak}}^{\text{num}}$	
n	cfd5p4o	cfd7p6o	cfd5p4o	cfd7p6o
51	0.28452	0.29824	25.8310	26.0129
101	0.29832	0.30005	25.9816	25.9968
201	0.29885	0.30001	25.9982	25.9998
401	0.29999	0.30000	25.9998	26.0000

For $\mu = 1/16, k = 2, t = 60$; adaptive grid with $a_1 = a_2 = 1, \alpha = 0.01, \beta > 2.5, \Delta t_{\text{grid}} = 1$.

The corresponding numerical errors in the peak amplitude and its location are given in Table 3.2 for the cases of uniform and adaptive grids, and for the number of nodes

varying from 51 to 401. These relative errors are defined by

$$e_{\text{amp}} = 1 - \frac{\max\limits_{x} u^{\text{num}}_{\text{peak}}(x,t)}{\max\limits_{x} u^{\text{exact}}_{\text{peak}}(x,t)} \quad \text{and} \quad e_{\text{phase}} = 1 - \frac{x^{\text{num}}_{\text{peak}}(x,t)}{x^{\text{exact}}_{\text{peak}}(x,t)}, \tag{3.22}$$

respectively, and the latter is the well-known phase error. The data in Tables 3.1 and 3.2 show, as one might expect, that the predicted peak amplitude and its location are more accurate for finite-difference scheme cfd7p6o than cfd5p4o, and also when a larger number of nodes is incorporated into the grid.

Table 3.2 Errors in Peak Amplitude and Its Location

	e_{amp}		e_{phase}	
n	cfd5p4o	cfd7p6o	cfd5p4o	cfd7p6o
51	0.0472	0.00586	0.00650	−0.000496
101	0.00559	−0.000161	0.000707	0.000125
201	0.000497	−0.0000207	0.0000679	0.00000777
401	0.0000385	−0.00000107	0.00000579	0.000000547

For $\mu = 1/16$, $k = 2$, $t = 60$; adaptive grid with $a_1 = a_2 = 1$, $\alpha = 0.01$, $\beta > 2.5$, $\Delta t_{\text{grid}} = 1$.

Additional results for numerical errors in the peak amplitude and its location are summarized in Table 3.3 for the cases of uniform and adaptive grids with 101 nodes, and at times varying from $t = 0$ to 60. The relative errors in amplitude and phase for the case of the uniform grid are rather large, because 101 nodes are simply insufficient to obtain a good MOL solution. However, these errors are considerably less and quite acceptable for the case of the adaptive grid. Such tabulated results illustrate clearly the advantages of using an adaptive grid over a uniform grid in the MOL for the case of the same number of nodes.

Table 3.3 Relative Errors of Peak Amplitude and Phase

	e_{amp}		e_{phase}	
Time	Uniform	Adaptive	Uniform	Adaptive
0	0.0680	0.000000169	0.00193	−0.000000105
10	0.0823	0.0000697	0.0137	−0.00000990
20	0.0870	−0.000174	0.0198	−0.00000738
30	0.0890	−0.000243	0.0249	0.0000320
40	0.0905	−0.000209	0.0293	0.0000671
50	0.0915	−0.000196	0.0330	0.0000970
60	0.0919	−0.000161	0.0362	0.000125

For $\mu = 1/16$, $k = 2$, $n = 101$; adaptive grid with $a_1 = a_2 = 1$, $\alpha = 0.01$, $\beta > 2.5$, $\Delta t_{\text{grid}} = 1$.

The constancy of the three invariants of motion given by Equation (3.7) were monitored during the MOL computations, and some of these results are presented after our analytical and numerical methods of evaluating C_1, C_2, and C_3 have been explained. The integrals for the three invariants of motion given by Equation (3.7) can be evaluated analytically for our problem involving a single solitary wave. The exact results are

$$C_1^{\text{exact}} = 6\tfrac{c}{k} = 12c\sqrt{\mu}\,, \quad C_2^{\text{exact}} = \tfrac{72}{5}\tfrac{c^2}{k} = \tfrac{144}{5}c^2\sqrt{\mu}\,,$$
$$C_3^{\text{exact}} = \tfrac{144}{5}\tfrac{c^3}{k} = \tfrac{288}{5}c^3\sqrt{\mu}\,, \tag{3.23}$$

at $t = 0$ and for $u_L = 0$ and $u_U = 0$, leading to $C_1^{\text{exact}} = 3/10 = 0.3\overline{0}$, $C_2^{\text{exact}} = 9/125 = 0.072\overline{0}$ and $C_3^{\text{exact}} = 9/125 = 0.0144\overline{0}$ for our problem with $\mu = 1/16$ and $c = 3/10$. In the numerical calculations the values of C_1^{num}, C_2^{num}, and C_3^{num} are evaluated in the following manner. The quintic interpolant given by Equation (3.21) is used to interpolate $u(x,t)$ between nodes. For $C_1^{\text{num}} = \int_{x_L}^{x_U} u(x,t_j)\,\mathrm{d}x$, this interpolant is first integrated for a general grid interval, and the results for all intervals are then summed to obtain C_1^{num}. For $C_2^{\text{num}} = \int_{x_L}^{x_U} \left[u^2(x,t_j) + \mu u_x^2(x,t_j)\right]\mathrm{d}x$, the interpolant is squared, the differentiated interpolant is squared, the first part is added to μ times the second part, these results are then integrated for a general grid interval, and the results for all intervals can then be summed to get C_2^{num}. For $C_3^{\text{num}} = \int_{x_L}^{x_U} u^3(x,t_j)\,\mathrm{d}x$ the interpolant is first cubed, integrated, etc. The integrations for a general grid interval can be done fairly easily by using well-known software such as Mathematica or Maple.

Table 3.4 Errors in Invariants of Motion for cfd5p4o and cfd7p6o

	C_1^{num}		C_2^{num}		C_3^{num}	
n	cfd5p4o	cfd7p6o	cfd5p4o	cfd7p6o	cfd5p4o	cfd7p6o
51	0.28452	0.30059	0.066297	0.071333	0.012701	0.014197
101	0.29821	0.29962	0.071280	0.071982	0.014184	0.014394
201	0.29983	0.29996	0.071934	0.072000	0.014380	0.014400
401	0.29999	0.30000	0.071995	0.072000	0.014440	0.014400

For $\mu = 1/16, k = 2, t = 60$; adaptive grid with $a_1 = a_2 = 1, \alpha = 0.01, \beta > 2.5, \Delta t_{\text{grid}} = 1$.

The numerically computed values of the three invariants C_1^{num}, C_2^{num}, and C_3^{num} are given in Table 3.4 for the cases of the different finite-difference schemes cfd5p4o and cfd7p6o and with grid nodes varying from 51 to 401. The corresponding relative errors are summarized in Table 3.5 for the same conditions. These two sets of results clearly show that the invariants of motion are better preserved in the MOL solutions, or the errors are smaller, for scheme cfd7p6o in contrast to cfd6p5o, and also when the number of nodes is increased. An additional set of the relative errors for the invariants of motion are given in Table 3.6 for the cases of uniform and adaptive grids

Table 3.5 Relative Errors in Invariants of Motion for cfd5p4o and cfd7p6o

	$\frac{C_1^{\text{exact}} - C_1^{\text{num}}}{C_1^{\text{exact}}}$		$\frac{C_2^{\text{exact}} - C_2^{\text{num}}}{C_2^{\text{exact}}}$		$\frac{C_3^{\text{exact}} - C_3^{\text{num}}}{C_3^{\text{exact}}}$	
n	cfd5p4o	cfd7p6o	cfd5p4o	cfd7p6o	cfd5p4o	cfd7p6o
51	0.0516	−0.00196	0.0792	0.00926	0.118	0.0141
101	0.00598	0.00126	0.0100	0.000255	0.0150	0.000384
201	0.000566	0.000140	0.000912	−0.00000336	0.00137	−0.00000503
401	0.0000472	0.0000120	0.0000722	0.000000143	0.000109	0.000000215

For $\mu = 1/16, k = 2, t = 60$; adaptive grid with $a_1 = a_2 = 1, \alpha = 0.01, \beta > 2.5, \Delta t_{\text{grid}} = 1$.

at various times from 0 to 60. These results clearly show, as expected, that the use of an adaptive grid in the MOL computations is superior to the uniform grid for the case of the same number of grid nodes.

Table 3.6 Relative Errors in Invariants of Motion for Uniform and Adaptive Grids

	$\frac{C_1^{\text{exact}} - C_1^{\text{num}}}{C_1^{\text{exact}}}$		$\frac{C_2^{\text{exact}} - C_2^{\text{num}}}{C_2^{\text{exact}}}$		$\frac{C_3^{\text{exact}} - C_3^{\text{num}}}{C_3^{\text{exact}}}$	
Time	Uniform	Adaptive	Uniform	Adaptive	Uniform	Adaptive
0	0.000712	0.0000509	0.0251	−0.000000276	0.0432	−0.000000125
10	−0.0253	0.000297	0.0345	0.0000317	0.0593	0.0000481
20	−0.0658	0.000489	0.0350	0.0000762	0.0611	0.0001152
30	−0.0772	0.000684	0.0354	0.000119	0.0621	0.000181
40	−0.0789	0.000872	0.0357	0.000166	0.0628	0.000250
50	−0.0792	0.00106	0.0358	0.000207	0.0633	0.000313
60	−0.0793	0.00126	0.0359	0.000255	0.0635	0.000384

For $\mu = 1/16, k = 2, n = 101$; adaptive grid with $a_1 = a_2 = 1, \alpha = 0.01, \beta > 2.5, \Delta t_{\text{grid}} = 1$.

One might think that the relative errors in the three invariants of motion at time $t = 0$ in Table 3.6 should be identically zero, because at $t = 0$ the solution for the solitary wave motion is started with initial conditions obtained directly from the exact solution, and the MOL solution procedure has yet to begin. However, the integration scheme using discrete data from the grid to evaluate C_1^{num}, C_2^{num}, and C_3^{num} and using the interpolant given by Equation (3.21) is approximate. In the present problem, these spatial integration errors associated with evaluating the invariants of motion at a given time are one to three orders of magnitude smaller than the time integration errors associated with the solution procedure by the MOL. Hence, we believe that the values of C_1^{num}, C_2^{num}, and C_3^{num} given in Table 3.4 and the relative errors given in Tables 3.5 and 3.6 primarily reflect the errors due to the MOL solution. In other

words, our reported errors for the MOL procedure are not dominated by the spatial integration errors from determining the invariants of motion. If we had implemented the spatial integrations to obtain the invariants of motion by using the trapezoidal rule, Simpson's rule, or a cubic spline, then the results would have been dominated by spatial integration errors. Most previous studies incorporate a low-order spatial integration scheme (e.g., Simpson's rule or a cubic spline) to evalue the invariants of motion, and their reported values of C_1^{num}, C_2^{num}, and C_3^{num} and their associated relative errors are inaccurate, being contaminated by the spatial integration error.

Error norms have also been computed for the current problem because the exact solution is known. We use the norm $L_2 = \|u^{\text{exact}} - u^{\text{num}}\|_2$, which is defined by

$$L_2 = \left[\frac{1}{x_U - x_L} \int_{x_L}^{x_U} \left(u^{\text{exact}} - u^{\text{num}} \right)^2 \, \mathrm{d}x \right]^{1/2} \tag{3.24}$$

$$\approx \left[\frac{1}{x_U - x_L} \sum_{i=1}^{n-1} \frac{1}{2} \left(x_{i+1} - x_i \right) \left(\left\{ u_i^{\text{exact}} - u_i^{\text{num}} \right\}^2 + \left\{ u_{i+1}^{\text{exact}} - u_{i+1}^{\text{num}} \right\}^2 \right) \right]^{1/2},$$

for the continuous and discrete error functions, respectively. The former is integrated by the trapezoidal rule to get the latter, in which u_i^{exact} and u_i^{num} are the exact and numerical solutions at the ith node x_i. We also use the norm $L_\infty = \|u^{\text{exact}} - u^{\text{num}}\|_\infty$, defined as

$$L_\infty = \max \left| u^{\text{exact}} - u^{\text{num}} \right| \approx \max_i \left| u_i^{\text{exact}} - u_i^{\text{num}} \right| \tag{3.25}$$

for the continuous and discrete error functions, respectively.

Results for these error norms for the cases of different finite-difference schemes and varying times from $t = 0$ to 60, and for uniform and adaptive grids with different numbers of grid nodes, are summarized in Tables 3.7 and 3.8. These tabulated data illustrate once again that the MOL using the larger stencil associated with cfd7p6o and an adaptive grid gives more accurate solutions than when the MOL uses a smaller stencil associated with cfd6p5o and a uniform grid with the same number of nodes.

Table 3.7 L_2 and L_∞ Errors for cfd5p4o and cfd7p6o

	L_2		L_∞	
n	cfd5p4o	cfd7p6o	cfd5p4o	cfd7p6o
51	0.00658	0.000636	0.0636	0.00439
101	0.000758	0.000139	0.00615	0.00103
201	0.0000725	0.00000861	0.000587	0.0000621
401	0.0000062	0.00000060	0.000050	0.0000043

For $\mu = 1/16, k = 2, t = 60$; adaptive grid with $a_1 = a_2 = 1, \alpha = 0.01, \beta > 2.5, \Delta t_{\text{grid}} = 1$.

Table 3.8 L_2 and L_∞ Errors for Uniform and Adaptive Grids

	L_2		L_∞	
Time	Uniform	Adaptive	Uniform	Adaptive
0	$0.\bar{0}$	$0.\bar{0}$	$0.\bar{0}$	$0.\bar{0}$
10	0.00810	0.0000195	0.0487	0.000121
20	0.0138	0.0000494	0.0889	0.000166
30	0.0185	0.0000494	0.128	0.000321
40	0.0229	0.0000764	0.159	0.000542
50	0.0273	0.000106	0.183	0.000774
60	0.0316	0.000139	0.206	0.001032

For $\mu = 1/16, k = 2, n = 101$; adaptive grid with $a_1 = a_2 = 1, \alpha = 0.01, \beta > 2.5, \Delta t_{\text{grid}} = 1$.

Some interesting information on the time integration of the EW equation with $\mu = 1/16$ for the motion of the solitary wave by the DAE solver called DASSL in the MOL is presented in Table 3.9. The total number of time steps to solve the problem by the MOL from time $t = 0$ to 60 ranges from 2894 to 2945 for the two finite-difference schemes cfd5p4o and cfd7p6o and grid nodes varying from 51 to 401. The size of each time step is selected by the solver to ensure that the time integration is done accurately, and these steps are for the most part independent of the number of nodes. These time steps are fairly constant at about 0.02 because the solitary wave propagates with a constant shape and speed, so the time integration is of the same degree of difficulty for each time step.

Table 3.9 Time Steps, Function and Jacobian Calculations, and CPU Times

	Total Number of Time Steps		Total Number of Function Evaluations		Total Number of Jacobian Calculations		CPU Time (s)	
n	cfd5p4o	cfd7p6o	cfd5p4o	cfd7p6o	cfd5p4o	cfd7p6o	cfd5p4o	cfd7p6o
51	2894	2930	4339	4376	1202	1202	62	85
101	2942	2945	4388	4391	1202	1202	124	169
201	2945	2945	4391	4390	1202	1202	246	341
401	2944	2944	4388	4330	1202	1202	491	668

For $\mu = 1/16, k = 2, t = 60$; adaptive grid with $a_1 = a_2 = 1, \alpha = 0.01, \beta > 2.5, \Delta t_{\text{grid}} = 1$.

The number of function evaluations used in the solver DASSL is related to the number of times the DAEs are used in the full solution procedure, whereas the number of Jacobian calculations are the number of times the entries of the Jacobian matrix are calculated or updated. The number of function and Jacobian calculations are also related to the time integration and not to the number of nodes, which explains why they are approximately constant for schemes cfd5p4o and cfd7p6o and also for the grid nodes varying from 51 to 401.

The central processor (CPU) time needed by the computer to determine a complete MOL solution to the current problem is directly proportional to the number of grid nodes and size of the grid-point stencil of the finite-difference scheme. The CPU times given in Table 3.9 are those for a Pentium III computer with a 500-MHz processor and LINUX operating system. From the tabulated data one can see that the CPU time almost exactly doubles as the number of nodes doubles for both cases of finite-difference schemes cfd5p4o and cfd7p6o. The CPU time for scheme cfd7p6o is about 37% larger than that for scheme cfd5p4o, for each case with the same number of nodes. This occurs mainly because scheme cfd7p4o has a seven-point stencil and an eleven-band Jacobian matrix compared to the five-point stencil and seven-band Jacobian matrix for cfd5p4o, which are respectively 40% and 57% larger for scheme cfd7p6o, making computations for the larger stencil cfd7p6o more tedious and longer.

Solutions to the current solitary wave problem by the MOL using the finite-difference scheme cfd7p6o as compared to cfd5p4o are obtained more efficiently by about 30%, in terms of the CPU time for numerical results obtained with the same accuracy. From Tables 3.2, 3.5, and 3.7 we observe that solutions for scheme cfd5p4o are about the same accuracy as those for scheme cfd7p6o when the number of grid nodes for scheme cfd7p6o are one half of those for scheme cfd5p4o. Hence, from Table 3.9 the CPU times of 85, 169, and 341 for 51, 101, and 201 grid nodes for scheme cfd7p6o should be compared directly to the CPU times of 124, 246, and 491 for 101, 201, and 401 grid nodes for scheme cfd5p4o. This observation leads to the conclusion that the CPU times are roughly 30% smaller for MOL solutions of the same accuracy with scheme cfd7po vs. scheme cfd5po. Hence, the additional computer programming effort in implementing a higher order finite-difference scheme (larger stencil width) provides a payoff in terms of a modest reduction in the CPU time for a given accuracy.

3.4.2 Inelastic Interaction of Solitary Waves

The interaction of two solitary waves, such as two waves that collide or one wave that overtakes a slower wave, are fascinating from the point of view that the transmitted waves may retain much of their original shapes and speeds after the interaction and additional waves can be generated by the interaction process, because the interaction is inelastic or unclean for the EW equation, as noted in the introduction. Such inelastic interactions are also interesting from the point of view of the capability of the MOL to obtain good solutions, because the predictions can be difficult in terms of accurately capturing the wave interaction process and generation of small secondary waves. Solutions to the EW equation, $u_t + uu_x - \mu u_{xxt} = 0$, with $\mu = 1$ by the MOL are presented for three different problems in this section. Two of these problems involve the collision of two waves and the other involves the overtaking of one wave by another. None of these problems have known analytical solutions to provide a guide to, or a check on, the numerical results that are presented in this section.

The first problem for the EW equation with $\mu = 1$ involves the collision of two waves, a problem studied originally for the RLW equation by Santarelli [26]. One

wave is specified initially at time $t = 0$ by the solitary wave solution given by Equation (3.2) as $3c_1\text{sech}^2[k_1(x - x_{01})]$ with $c_1 = \nu_1 = 1.7\overline{0}$, $k_1 = 0.5\overline{0}$, and $x_{01} = 35$, and the second wave is also specified initially by the solitary wave solution as $3c_2\text{sech}^2[k_2(x - x_{02})]$ with $c_2 = \nu_2 = -3.4\overline{0}$, $k_2 = 0.5\overline{0}$, and $x_{02} = 65$. These two waveforms are simply added together and used as initial conditions at $t = 0$. The MOL solution is obtained by using the finite-difference scheme cfd7p6o on an adaptive grid with 201 nodes over the spatial interval [0, 80]. Solutions are computed for the time interval [0, 16], and grid adaptation is done at regular time intervals of $\Delta t_{\text{grid}} = 2$. The other adaptive grid parameters are $a_1 = a_2 = 1$, $\alpha = 0.001$, and $\beta = 0.1$. This value of β causes a significant degree of clipping because the maximum value of $|u_x| + |u_{xx}|$ ranges from a value of 6 for the initial solitary waves to 40 during their interaction.

A time-distance diagram of the numerical computations is depicted in Figure 3.9. The rightward traveling wave with the positive amplitude ($3c_1 = 5.1\overline{0}$) and speed ($\nu_1 = c_1 = 1.7\overline{0}$) and the leftward propagating wave with the negative amplitude ($3c_2 = -10.2\overline{0}$) and speed ($\nu_2 = c_2 = -3.4\overline{0}$) eventually collide, and after their interaction they become transmitted waves with shapes and speeds that appear similar to their original ones. A closer examination shows that their peak amplitudes and speeds are about 2% smaller. During the collision these two interacting waves experience slight phase shifts, which are not readily seen in Figure 3.9. The collision process also produces a small stationary disturbance between the transmitted waves, which is easily seen. Santarelli [26] solved this problem first for the RLW equation and resolved the small disturbance, which he called a pair of daughter waves. His work was later confirmed by Lewis and Tjon [20] and Gardner and Gardner [12].

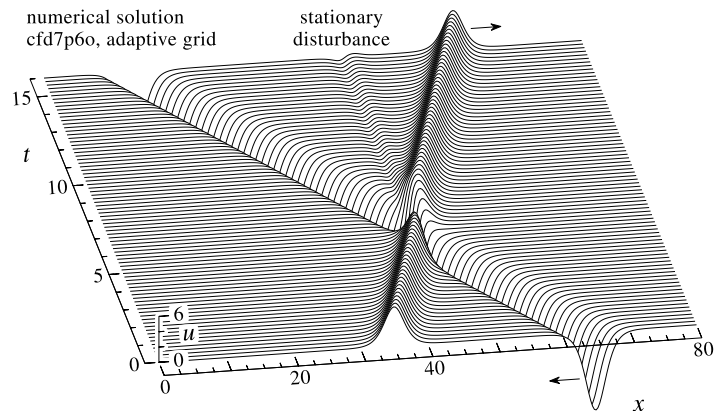


FIGURE 3.9
Collision of two solitary waves leaving a stationary disturbance.

Two spatial distributions at times $t = 0$ and 14 are shown more clearly in Figure 3.10, where the shape of the small stationary disturbance is enlarged for clarity (the amplitude is magnified by a factor of 15). The distributions of grid nodes at the two different times are also shown for interest in the lower part of the figure, to illus-

trate the node clustering in solution regions of larger gradients and curvatures. The solution by the MOL with only 201 nodes appears to provide a good solution to this problem, from the viewpoint that the grid adaptation seems to well resolve solution regions of large gradients and curvatures, including the transition through the small stationary disturbance. For interest, the maximum and minimum node spacings for this problem are about 5 and 1/2 times the average node spacing, respectively.

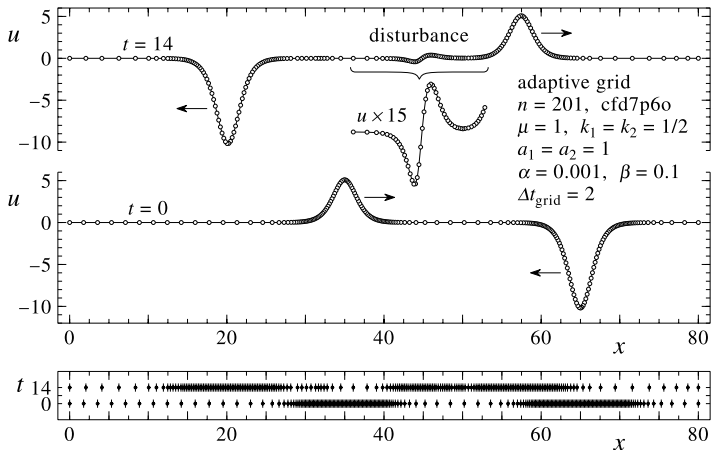


FIGURE 3.10
Collision of two solitary waves and details of the stationary disturbance.

The trajectories of the peak amplitudes of the two waves before, during, and after the collision are plotted as solid lines in Figure 3.11. The shifts in these trajectories from the extrapolated paths given by the dashed lines are shown clearly, whereas these phase shifts were not obvious in Figure 3.9, even though these shifts are substantial at -1.43 and 2.64 for the positive and negative amplitude waves, respectively.

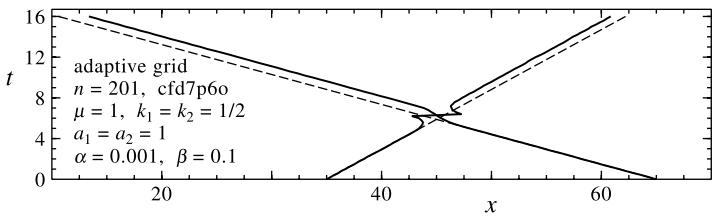


FIGURE 3.11
Collision of two solitary waves depicting their phase shifts.

The invariants of motion for spatially localized solitary waves and initial data are given in general by Equation (3.7), and for a single solitary wave the integrals can be evaluated analytically to obtain the specific results of Equation (3.23), with $u_L = 0$ and $u_U = 0$. For the present problem with two solitary waves whose

overlap in the spatial interval $[0, 80]$ is insignificant at the initial time $t = 0$, the invariants of motion can be estimated accurately by simply adding the results for each wave. This gives $C_1^{\text{est}} = 12(c_1 + c_2)\sqrt{\mu}$, $C_2^{\text{est}} = (144/5)(c_1^2 + c_2^2)\sqrt{\mu}$, and $C_3^{\text{est}} = (288/5)(c_1^3 + c_2^3)\sqrt{\mu}$. For this problem with $\mu = 1$ and the data for c_1 and c_2, we obtain $C_1^{\text{est}} = -20.4\overline{0}$, $C_2^{\text{est}} = 416.16\overline{0}$, and $C_3^{\text{est}} = -1980.9216\overline{0}$. The numerical values of the invariants of motion were evaluated and monitored at equal time intervals of 2, and the results for C_1^{num}, C_2^{num}, and C_3^{num} remained constant to five significant digits. Such results are good indications that the numerical computations by the MOL with cfd7po and an adaptive grid are fairly accurate.

The second problem for the EW equation with $\mu = 1$ also involves the head-on collision of two solitary waves having positive and negative amplitudes, but the magnitudes of these amplitudes are set equal. The first wave is specified at time $t = 0$ by the solitary wave solution given by Equation (3.2) as $3c_1\text{sech}^2\,[k_1\,(x - x_{01})]$ with the parameter values $c_1 = \nu_1 = 1.5\overline{0}$, $k_1 = 0.5\overline{0}$, and $x_{01} = 23$, and the second wave is also specified initially by the solitary wave solution as $3c_2\text{sech}^2\,[k_2\,(x - x_{02})]$ and the parameters are given by $c_2 = \nu_2 = -1.5\overline{0}$, $k_2 = 0.5\overline{0}$, and $x_{02} = 38$. These two results are added and used as the initial conditions at $t = 0$. The MOL solution is obtained by using the finite-difference scheme cfd7p6o on an adaptive grid with 601 nodes over the spatial interval $[-30, 90]$. Solutions are computed for the time interval $[0, 55]$, and grid adaptation is done at regular time intervals of $\Delta t_{\text{grid}} = 1$. The other adaptive grid parameters are $a_1 = a_2 = 1$, $\alpha = 0.001$ and $\beta = 10$. This value of β causes significant clipping because the maximum value of $|u_x| + |u_{xx}|$ ranges from 2.9 for the initial solitary waves to 750 during their interaction.

A time-distance diagram of the numerical computations is given in Figure 3.12 for the spatial interval $[-10, 70]$, which is the main part of the entire computational interval $[-30, 90]$. The rightward traveling wave with the positive amplitude ($3c_1 = 4.5\overline{0}$) and speed ($\nu_1 = c_1 = 1.5\overline{0}$) and the leftward propagating wave with the negative amplitude ($3c_2 = -4.5\overline{0}$) and speed ($\nu_2 = c_2 = -1.5\overline{0}$) eventually collide, and after their interaction they become transmitted waves with similar shapes but substantially lower peak amplitudes and speeds (by a factor of about 2.5). The collision process produces a train of non-negligible amplitude waves moving between the transmitted waves, and it also produces a strong stationary disturbance with very steep gradients and curvatures. Note that Santarelli [26] was the first to solve this problem with the RLW equation and resolve the large train of moving waves and strong stationary disturbance.

Two spatial distributions at times $t = 0$ and 50 are shown more clearly in Figure 3.13, where the profile of the train of disturbances is displayed and the shape of the stationary disturbance is enlarged for clarity (twofold in amplitude and ninefold in space). The distributions of grid nodes at the two different times are also shown for interest in the lower part of the figure. About 601 nodes are needed to provide a good solution for this problem because grid adaptation then appears to well resolve solution regions of large gradients and curvatures. Note that only every second node is shown in both parts of Figure 3.13 for reasons of clarity, and the results are shown

on the entire spatial interval $[-10, 90]$. The maximum and minimum node spacings are approximately 10 and 1/30 times the average node spacing, respectively.

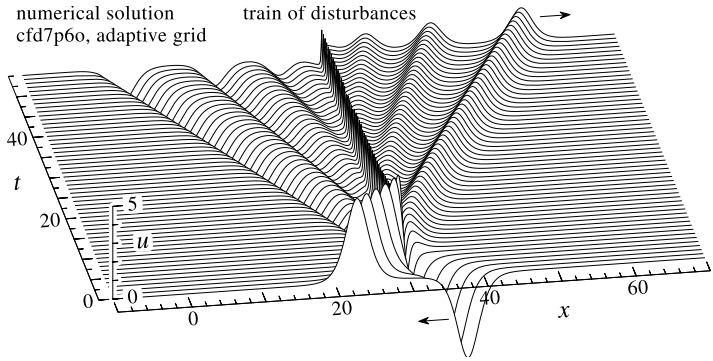


FIGURE 3.12
Collision of two solitary waves leaving a train of disturbances.

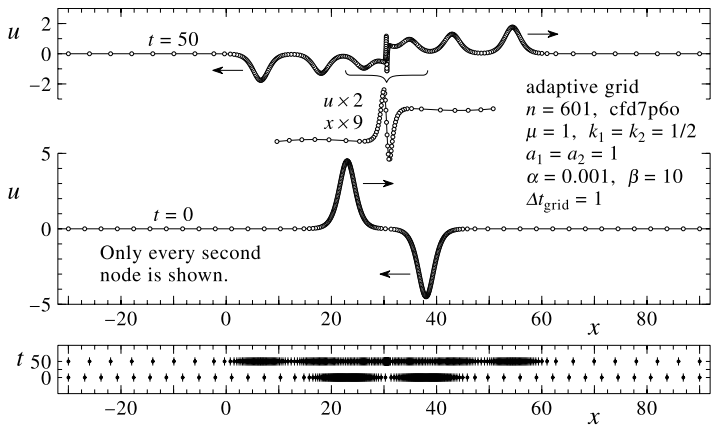


FIGURE 3.13
Collision of two solitary waves and details of the train of disturbances.

The invariants of motion at the initial time $t = 0$ are calculated in the same manner as for the previous problem, and the final results are given by $C_1^{\text{est}} = 0.\overline{0}$, $C_2^{\text{est}} = 129.6\overline{0}$, and $C_3^{\text{est}} = 0.\overline{0}$. The numerical values of the invariants of motion were evaluated and monitored during the MOL solution procedure, and the value of $|C_1^{\text{num}}|$ stayed below 2.81×10^{-6}, C_2^{num} remained constant to five significant digits, and $|C_3^{\text{num}}|$ stayed less than 13.9×10^{-6}. Such results provide some assurance that the numerical computations by the MOL with cfd7po and an adaptive grid are fairly accurate for this problem.

The third problem for the EW equation with $\mu = 1$ involves the overtaking of one solitary wave by another, a problem first studied by Abdulloev et al. [1] with the RLW

equation. The first wave is specified at time $t = 0$ by a solitary wave solution given by Equation (3.2) as $3c_1\text{sech}^2\,[k_1\,(x - x_{01})]$ with $c_1 = \nu_1 = 3.4\overline{0}$, $k_1 = 0.5\overline{0}$, and $x_{01} = 15$, and the second wave is also specified initially by a solitary wave solution as $3c_2\text{sech}^2\,[k_2\,(x - x_{02})]$ with $c_2 = \nu_2 = 1.7\overline{0}$, $k_2 = 0.5\overline{0}$, and $x_{02} = 35$. These two results are added and used as the initial conditions. The MOL solution is obtained by using the finite-difference scheme cfd7p6o on an adaptive grid with 301 nodes over the spatial interval $[-10, 130]$. Solutions are computed for the time interval $[0, 26]$, and grid adaptation is done at regular time intervals of $\Delta t_{\text{grid}} = 1$. The other adaptive grid parameters are $a_1 = a_2 = 1$, $\alpha = 0.0001$ and $\beta = 0.01$. This value of β invokes large amounts of clipping because the maximum value of $|u_x| + |u_{xx}|$ varies from 4 to 7 for the initial waves and their interaction.

A time-distance diagram of the numerical computations is shown in Figure 3.14 for the spatial interval $[0, 115]$, which is the main part of the entire computational interval $[-10, 130]$. The rightward moving wave with the amplitude ($3c_1 = 10.2\overline{0}$) and speed ($\nu_1 = c_1 = 3.4\overline{0}$) overtakes the rightward propagating wave with the smaller amplitude ($3c_2 = 5.1\overline{0}$) and speed ($\nu_2 = c_2 = 1.7\overline{0}$). After their interaction they become transmitted waves with almost their original shapes and speeds, but the trajectories of the large and small amplitude waves are shifted forward ($+3$) and backward (-3), respectively. The interaction produces a tiny disturbance whose location is indicated but its shape is not observable in Figure 3.14. This disturbance appears stationary from an examination of the numerical results.

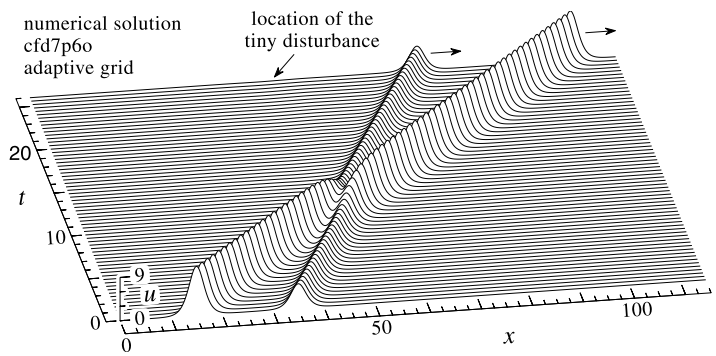


FIGURE 3.14
Overtaking of one solitary wave by another leaving a tiny disturbance.

Two spatial distributions at times $t = 0$ and 25 are shown more clearly in Figure 3.15, where the shape of the tiny stationary disturbance is magnified for clarity (50-fold in amplitude). The distributions of grid nodes at the two different times are also shown for interest in the lower part of Figure 3.15. The solution by the MOL for 301 nodes provides an excellent solution for this problem, although a good solution can be obtained with fewer nodes. Only every second node is shown in both parts of the figure for clarity. The maximum and minimum node spacings are approximately 4 and 1/3 times the average node spacing, respectively.

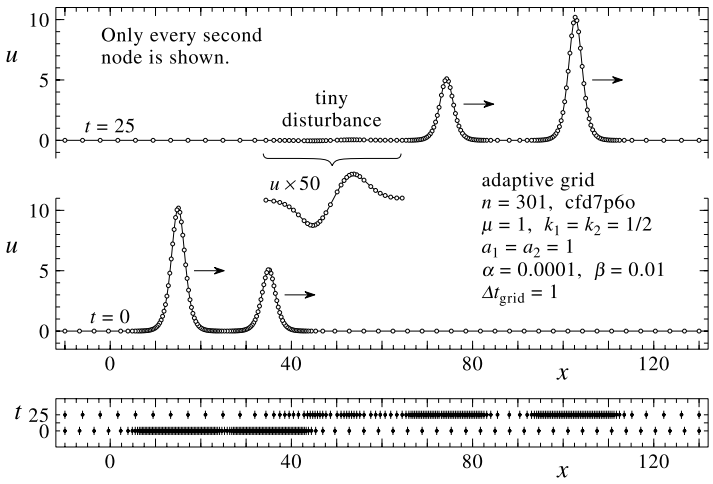


FIGURE 3.15
Overtaking of one solitary wave by another and details of the tiny disturbance.

The invariants of motion are calculated at the initial time $t = 0$ in the same manner as for the previous two problems, and the final results are given by $C_1^{\text{est}} = 61.2\overline{0}$, $C_2^{\text{est}} = 416.16\overline{0}$, and $C_3^{\text{est}} = 2546.8992\overline{0}$. The numerical values of the invariants of motion were calculated and monitored during the MOL solution procedure, and they remained constant to five significant digits. These results provide some assurance that the numerical computations by the MOL with cfd7po and an adaptive grid are fairly accurate for this problem.

The overtaking of the weaker solitary wave by the stronger solitary wave in this last problem produces a tiny stationary disturbance that is barely noticeable. Such a tiny disturbance could be easily overlooked in the results of numerical computations, or it could be mistaken as a numerical effect, and one might then conclude incorrectly that the interaction is elastic. Historically, this was the first type of inelastic wave interaction that was investigated, and the study was carried out by Abdulloev et al. [1] with the RLW equation, despite the numerical resolution difficulties with early numerical methods (e.g., low-order spatial discretization on a uniform grid and a low-order time integration scheme). Santarelli [26] focused his later studies with the RLW equation on the collision of two solitary waves, because the inelastic interaction resulted in a readily observable stationary disturbance in the first problem of this section and a strong train of disturbances in the second problem, such that the inelastic behavior could be exhibited clearly with modest computational tools of his time. Modern numerical methods based on high-order finite differences with an adaptive grid and a high-order time integration scheme used in this section clearly resolve all of these fascinating wave interactions, including the inelastic effects from a tiny disturbance to a train of strong disturbances.

3.4.3 Gaussian Pulse Breakup into Solitary Waves

The formation of a train of solitary waves from the breakup, dissolution, or decay of a single Gaussian shaped pulse specified initially (i.e., at time $t = 0$) has fascinated many researchers working on solutions of the KdV and RLW equations. For the KdV equation, the initial positive-amplitude Gaussian pulse does not propagate rightward as a single solitary wave but instead breaks into a train of rightward traveling positive-amplitude solitons when the value of μ is smaller than some critical value μ_{crit} defined by Berezin and Karpman [4]. For $\mu = \mu_{\text{crit}}$, the Gaussian pulse changes shape slightly and propagates as a positive-amplitude solitary wave and leaves behind a small oscillatory disturbance. For values of μ larger than μ_{crit}, the Gaussian pulse breaks into a set of alternating positive- and negative-amplitude waves, forming what might be a wave packet with one group velocity. Similar wave behavior was discovered for the RLW equation by Gardner et al. [10]. The initial Gaussian pulse and the formation of the train of solitary waves have sometimes been referred to as a *Maxwellian pulse* or *Maxwellian pulses*, terminology that is not adopted herein. In this section the EW equation, $u_t + uu_x - \mu u_{xxt} = 0$, is solved by the adaptive MOL for the case of an initial Gaussian pulse, and four different values for μ are chosen to provide a set of good illustrations of interesting Gaussian pulse breakups into solitary waves.

For all problems in this section the Gaussian pulse is used to provide the initial conditions at time $t = 0$, having the same symmetric profile $u(x, t = 0) = \exp\left[-(x - x_0)^2\right]$ centered at the spatial location $x_0 = 7$. This pulse is entirely positive, has an amplitude of unity, and features a width of 1.66511 at the one-half amplitude level. Although the EW equation is solved for four different problems corresponding to a wide range of values of $\mu = 1/100$, 1/25, 1/5, and 1, the solutions are obtained for all four problems for the same spatial domain $[-10, 40]$ and same temporal interval $[0, 50]$ by the MOL using the finite-difference scheme cfd7p6o and an adaptive grid with 401 nodes. The adaptive-grid parameters are fixed at $a_1 = a_2 = 1$, $\alpha = 0.00003$, $\beta = 0.01$, and $\Delta t_{\text{grid}} = 2$ for all four problems, which means they are not necessarily well *tuned* for each problem. The Dirichlet boundary conditions $u(x_L, t) = u_L = 0$ and $u(x_U, t) = u_U = 0$ are used in all four problems.

The integrals in the invariants of motion given by Equation (3.7) can be integrated analytically for the initial Gaussian profile specified at time $t = 0$. The resulting constants can be summarized as $C_1^{\text{exact}} = \sqrt{\pi}$, $C_2^{\text{exact}} = (1+\mu)\sqrt{\pi/2}$, and $C_3^{\text{exact}} = \sqrt{\pi/3}$ for later reference. These are evaluated at time $t = 0$ and for $u_L = 0$ and $u_U = 0$.

Numerical results from the MOL solution of the EW equation for the first problem with $\mu = 1/100$ are now presented. A time-distance diagram of the computations is given first in Figure 3.16 for the spatial interval $[0, 35]$, which is the main part of the computational interval $[-10, 40]$. One can see that the initial Gaussian pulse does not propagate as a single solitary wave. Instead, this pulse breaks up or evolves fairly rapidly into a set of rightward traveling waves that become more and more separated in space. The leading wave has the largest amplitude and speed, and the trailing waves successively decrease in amplitude and speed. These waves, when they are travelling separately, all appear to behave like solitary waves with the same theoretical

wave number $k = 1/\sqrt{4\mu} = 5$ and therefore the same width. Although seven waves can be counted in the computational time interval [0, 50], some additional smaller amplitude waves will likely occur at later times, until the continuously elongating and flattening left side of the Gaussian profile vanishes. An examination of the numerical results shows that the constant width of these solitary waves is about 0.35 at one-half amplitude. This is much smaller than the corresponding width 1.665 of the initial Gaussian profile, by the factor 0.21. Additionally, the leading wave has an amplitude of 1.53 and a speed of 0.51. This amplitude is larger than unity for the initial Gaussian profile.

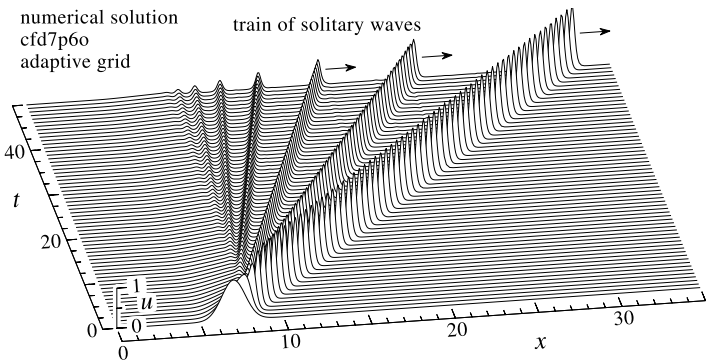


FIGURE 3.16
Solitary waves evolving from an initial Gaussian profile ($\mu = 1/100$).

Three spatial distributions at times $t = 0$, 10, and 20 are given in Figure 3.17, and they help illustrate the breakup and evolution of the Gaussian profile into a sequence of waves with decreasing amplitudes. The elongation and flattening of the left side of the Gaussian profile is seen more clearly in this figure. The waves appear to develop at the right end of this flattening section of the Gaussian profile, and a piece of this flattening section disappears in the process. From the upper two plots given in Figure 3.17 one can observe that the peak amplitudes of the train of waves lie closely on a straight line, and this projected line intersects the x-axis slightly to the left of the Gaussian profile center (i.e., $x_0 = 7$). An examination of the numerical data reveals that the leading wave is a solitary wave, based on its shape, peak amplitude ($3c \approx 1.53$), and speed ($c \approx 0.51$). When the peak amplitudes of the trailing waves lie closely on a straight line that includes the leading solitary wave, these trailing waves are then also solitary waves.

The results in Figure 3.17 show that the numerical solution by the adaptive MOL with 401 nodes appears to provide a fair solution for this first problem in the time interval [0, 50]. The invariants of motion should be constant at the exact values $C_1^{\text{exact}} = \sqrt{\pi}$, $C_2^{\text{exact}} = (1 + \mu)\sqrt{\pi/2}$, and $C_3^{\text{exact}} = \sqrt{\pi/3}$. The values of C_1^{num}, C_2^{num}, and C_3^{num} were calculated and monitored during the solution process, and the exact values were reproduced and remained accurate to three to four significant digits. This provides some assurance that the solution is fairly accurate. However, we note

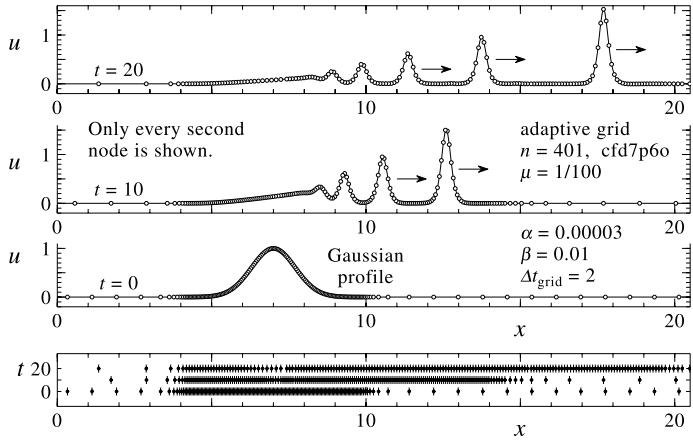


FIGURE 3.17
Profiles of solitary waves evolving from an initial Gaussian profile ($\mu = 1/100$).

that the nodes are too widely spaced away from the solitary waves, and the closest spacing of the nodes throughout the solitary wave train tends to become uniformly constant at later times and not sufficiently clustered (see the lowest diagram of the node spacings). This occurs because the values of $\alpha = 0.00003$ and $\beta = 0.01$ are somewhat too small for this first problem. For example, the maximum value of $|u_x| + |u_{xx}|$ ranges from 6 for the Gaussian profile to 80 for the train of narrow solitary waves at later times, so the low value of β produces too much clipping and prevents nodes from clustering more appropriately in localized regions of higher gradients and curvatures associated with individual solitary waves. Increases in the values of α, β and the number of nodes would provide a better solution resolution and accuracy.

Numerical results from the MOL solution of the EW equation for the second problem with $\mu = 1/25$ are now given. A time-distance diagram of the numerical computations appears in Figure 3.18 for the spatial interval [0, 32], which is the main part of the computational interval $[-10, 40]$. The initial Gaussian profile evolves fairly rapidly into a smaller number of rightward traveling waves than for the first problem, each successive wave with a smaller amplitude and a slower speed. These waves of decreasing amplitude again separate and behave like solitary waves, each with the same theoretical wave number $k = 1/\sqrt{4\mu} = 2.5$ and consequently the same width. Although only four waves can be counted in the computational time interval [0, 50], some additional smaller amplitude waves will likely occur at later times, until the elongating and flattening left side of the Gaussian profile vanishes. On examination of the numerical results, we find that the constant width of the waves is 0.71 at one-half amplitude. This width is again smaller than 1.665 for the initial Gaussian profile, by a factor of 0.42 (as compared to 0.21 for the first problem). The leading wave has an amplitude 1.31 that is larger than that of the initial Gaussian profile (in contrast to 1.53 for the first problem).

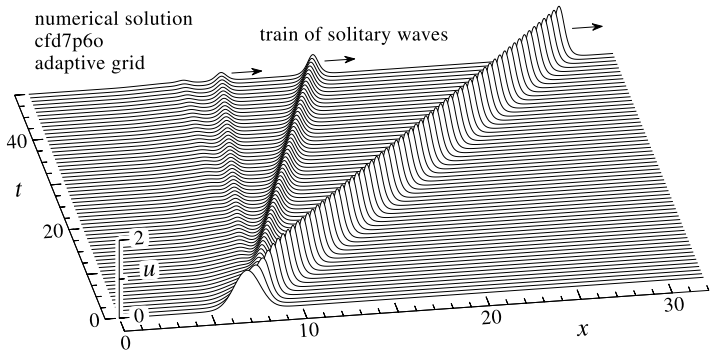


FIGURE 3.18
Solitary waves evolving from an initial Gaussian profile ($\mu = 1/25$).

Three spatial distributions at times $t = 0$, 16, and 32 are depicted in Figure 3.19, and they once again illustrate the evolution of the Gaussian profile into a sequence of waves with decreasing amplitudes and speeds. The elongation and flattening of the left side of the Gaussian profile can be seen in this figure. From the upper two plots in Figure 3.19 one can again observe that the peak amplitudes of the train of waves lie almost on a straight line, and the projected line intersects the x-axis between $x = 5$ to 6, to the left of the Gaussian profile center (i.e., $x_0 = 7$). When the peak amplitudes of the trailing waves lie closely on a straight line that includes the leading solitary wave, these trailing waves are then also solitary waves.

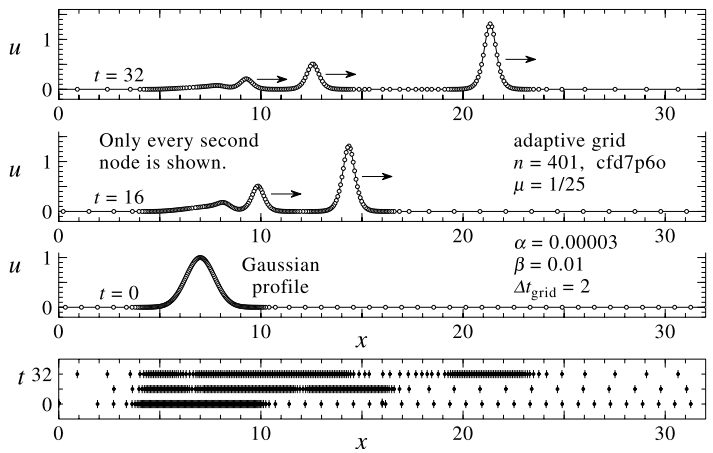


FIGURE 3.19
Profiles of solitary waves evolving from an initial Gaussian profile ($\mu = 1/25$).

From the numerical results displayed in Figure 3.19, the solution by the adaptive MOL with 401 nodes appears to provide a good solution for the second problem

in the time interval [0, 50]. This comment is supported by the calculations of the invariants of motion during the numerical computations. The exact values for C_1, C_2, and C_3 were reproduced and remained constant to five significant digits during the computations. From the upper two diagrams in Figure 3.19 we see that the nodes are better clustered in solution regions of high gradients and curvatures for this problem than the first one because the values of $\alpha = 0.00003$ and $\beta = 0.01$ are more appropriate, but they are still somewhat too small for this second problem. The small value of β causes too much clipping; the maximum value of $|u_x| + |u_{xx}|$ ranges from 6 for the Gaussian profile to 20 for the train of fairly narrow solitary waves at later times.

Numerical results for the third problem with $\mu = 1/5$ in the EW equation are now introduced. A time-distance diagram of the numerical computations is given first in Figure 3.20 for the reduced spatial interval [0, 26] and reduced time interval [0, 40]. The initial Gaussian pulse evolves into what appears to be a single rightward traveling solitary wave and a small stationary disturbance. On a closer examination of the numerical results, one finds that this wave behaves like a solitary wave with the theoretical wave number $k = 1/\sqrt{4\mu} = 1.118$. It has a constant width of 1.58 at one-half amplitude, which is only slightly smaller by 5% than the corresponding 1.665 for the initial Gaussian profile. Additionally, the solitary wave has an amplitude that is only 2.5% higher than unity for the initial Gaussian profile. As a consequence, the shape of the initial Gaussian pulse is a close match in amplitude, width, and shape to that of the solitary wave that develops from the pulse. Because of this, the solitary wave carries away the majority of the mass, momentum, and energy (pertaining to the conservation laws), and the small leftovers are contained in the small stationary disturbance.

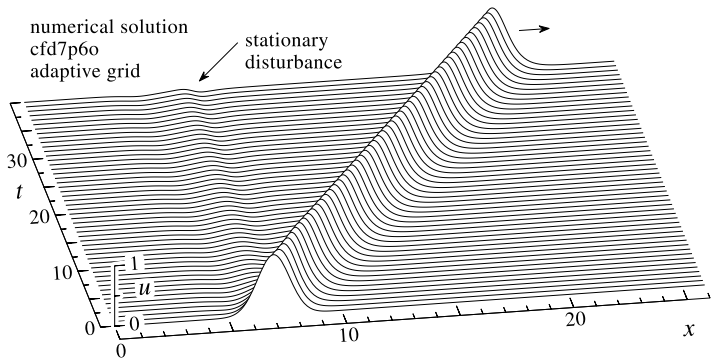


FIGURE 3.20
Solitary waves evolving from an initial Gaussian profile ($\mu = 1/5$).

Two spatial distributions at times $t = 0$ and 50 are given in Figure 3.21. The small disturbance is also shown separately as an inset, where it is amplified by a factor of ten for clarity. From the results given in this figure for $\mu = 1/5$, one can see more clearly the equivalence of the profiles of the Gaussian pulse and solitary wave.

From the results shown in Figure 3.21, the MOL solution with 401 nodes provides a good solution for this third problem in the time interval [0, 40]. This comment is supported by calculations of the invariants of motion during the MOL computations. The exact values were reproduced and remained constant to five significant digits. From the upper two diagrams in Figure 3.21, one can see that the grid nodes are well clustered in regions of high gradients and curvatures, better than for the two previous problems, because the values of $\alpha = 0.00003$ and $\beta = 0.01$ are now more appropriate. The maximum value of $|u_x| + |u_{xx}|$ ranges from 6 for the Gaussian profile to 3.5 for the smooth solitary wave and disturbance that evolve later.

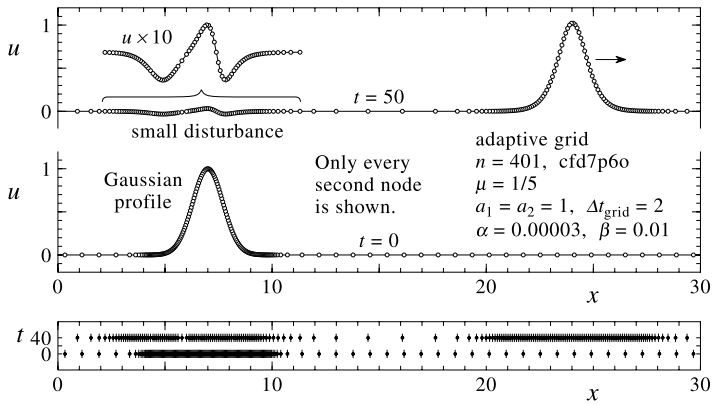


FIGURE 3.21
Profiles of solitary waves evolving from an initial Gaussian profile ($\mu = 1/5$).

The numerical results from the MOL solution of the EW equation for the fourth and last problem with $\mu = 1$ are now presented. A time-distance diagram of the numerical results is given first in Figure 3.22 for the spatial interval $[-10, 26]$, which is the main part of the computational interval $[-10, 40]$. The initial Gaussian pulse breaks into a

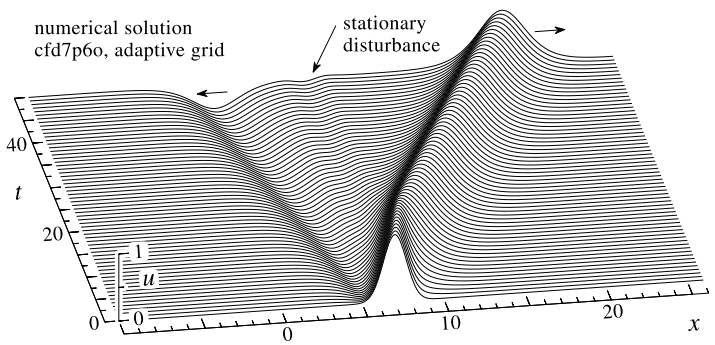


FIGURE 3.22
Solitary waves evolving from an initial Gaussian profile ($\mu = 1$).

rightward traveling wave with a positive amplitude and a leftward moving wave with a negative amplitude, and a small stationary disturbance that remains at $x_0 = 7$. The initial Gaussian pulse appears to move slowly at first as the leftward and rightward moving waves form, and then the rightward facing wave moves more quickly. The two waves appear to behave like solitary waves at later times, with the same theoretical wave number $k = 1/\sqrt{4\mu} = 1/2$ and hence the same width.

Three spatial distributions at times $t = 0$, 24, and 48 are displayed in Figure 3.23 to further illustrate the shape of the leftward and rightward traveling waves. From a close examination of the numerical data in Figures 3.22 and 3.23, the leftward and rightward moving waves have different amplitudes of 0.80 and -0.36, respectively, and they move at speeds of one-third of these values, typical of solitary waves computed from the EW equation. The widths of the rightward and leftward moving waves at the one-half amplitude level are the same at about 3.5, which is about twice as large as that corresponding to the initial Gaussian pulse.

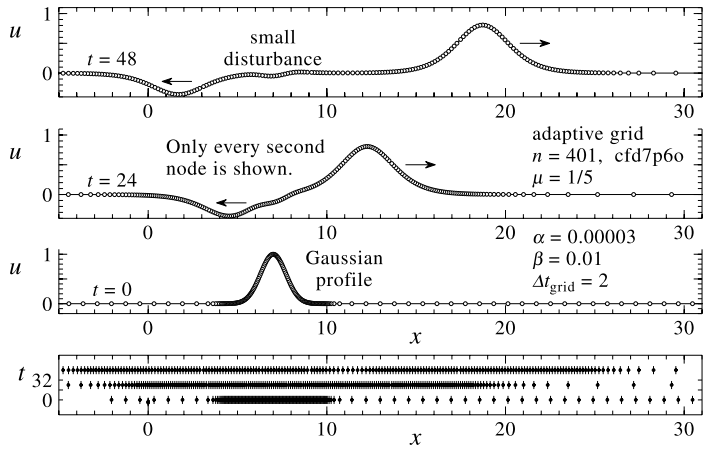


FIGURE 3.23
Profiles of solitary waves evolving from an initial Gaussian profile ($\mu = 1$).

The numerical solutions shown in Figures 3.22 and 3.23 for this last problem by the adaptive MOL with 401 grid nodes are quite good. The node spacings in the latter figure appear reasonable for the adaptive grid parameters selected for all four problems. The invariants of motion that were calculated and monitored during the numerical computations reproduced the theoretical values and remained constant to five to six significant digits. This enhanced accuracy over the previous three problems is likely due to the wider and smoother solitary waves and stationary disturbance that occur in this fourth problem.

Some additional observations can be made from all of the numerical results of the four problems. When $\mu \approx 1/5$ the width of solitary waves predicted by the EW equation is about equal to the width (1.665) of the initial Gaussian pulse. In this case, the Gaussian pulse evolves into one main solitary wave of about the same width

and leaves a small stationary disturbance behind. This solitary wave contains the majority of the mass, momentum, and energy of the initial Gaussian pulse, and the small remainder is in the small stationary disturbance. When $\mu < 1/5$ the width of solitary waves from the EW equation is smaller than that of the initial Gaussian pulse. In this case a train of narrow solitary waves must evolve from the initial Gaussian pulse to account for all of its initial mass, momentum, and energy. For smaller values of μ and correspondingly larger mismatches in the widths of the solitary waves and Gaussian pulse, the train of narrower solitary waves is more numerous and more closely spaced. When $\mu > 1/5$ the width of solitary waves from the EW equation is larger than that of the initial Gaussian profile. In this case the single, wide solitary wave contains more mass, momentum, and energy than the initial Gaussian pulse, and this seemingly results in a negative-amplitude pulse that becomes the provider of the extra mass and energy, although some may remain in the small disturbance remaining between the waves.

From the numerical results for the four problems, we deduce that the critical value of μ is roughly 1/5, and this criterion determines whether the breakup of the Gaussian pulse leads to a train of solitary waves ($\mu < 1/5$), only one solitary wave with a small stationary disturbance ($\mu \approx 1/5$), or a leftward moving solitary wave and a rightward moving solitary wave separated by a small disturbance ($\mu > 1/5$). However, this critical value of μ can be determined approximately by using the conservation laws. Let us equate the mass $\sqrt{\pi}$ and momentum $(1+\mu)\sqrt{\pi/2}$ in the Gaussian pulse to the corresponding mass $12c\sqrt{\mu}$ and momentum $(144/5)c^2\sqrt{\mu}$ of a solitary wave from the EW equation. These equivalences result in two nonlinear algebraic equations for the two unknowns μ and c, and the solution by means of an analytic or iterative method is given by $\mu_{\text{crit}}^{\text{est}} = 0.1804$ and $c_{\text{crit}}^{\text{est}} = 0.3478$ to four significant digits. If the mass and energy are used instead, then the critical values are very similar and given by $\mu_{\text{crit}}^{\text{est}} = (\pi/30)\sqrt{3} = 0.1814$ and $c_{\text{crit}}^{\text{est}} = (25/1728)^{1/4} = 0.3468$. These critical values are estimates only, because the solitary wave that evolves from the initial Gaussian pulse cannot contain exactly all of the mass, momentum, and energy of the initial Gaussian pulse.

3.4.4 Formation of an Undular Bore

A long wave with a gradually and monotonically sloped front can propagate in deep water without significant change in shape, when the nonlinear effects of steepening are balanced by dissipation and dispersion. However, as a long wave travels into shallow water, the smoothly varying front can steepen further, and this type of a steepening wave is called a bore. Ocean tides can produce large bores (e.g., amplitude of 3 m) that propagate upstream in river channels and attract bore watchers. When the surface elevation of the water behind a long bore is less than 0.28 times the water depth in front, the steepening front of a bore that is initially smoothly varying will develop surface ripples that grow into a train of large oscillations or undulations, and this type of a wave is called an undular bore. See Peregrine [23] for more details. In this section the EW equation, $u_t + uu_x - \mu u_{xxt} = 0$, with $\mu = 1/6$ is solved by the MOL

with an adaptive grid for a problem involving the development of an undular bore. The value of 1/6 is taken for μ so that the EW equation becomes applicable to the case of water waves and bores.

The initial conditions for the numerical computations are specified at time $t = 0$ by $u(x, t = 0) = (1/2)u_0[1 - \tanh(X)]$, with the nondimensional distance $X = (x - x_0)/d$. The spatial distribution of this initial bore is monotonically decreasing from an asymptotically constant amplitude $u = u_0$ as $x \to -\infty$ to the one-half amplitude level $u = u_0/2$ at $x = x_0$, where the surface slope $u_x = -(u_0/2)/d$, and then decreasing to the asymptotically constant amplitude $u = 0$ as $x \to +\infty$. The parameters for this initial bore shape are given by the bore amplitude $u_0 = 1/10$, the initial location $x_0 = 0$ of the center of the bore (at half amplitude) and slope control parameter $d = 5$. These are the same bore parameters that were used in a previous solution by Gardner et al. [10].

The problem involving the formation of an undular bore is solved in this study by the MOL using the finite-difference scheme cfd7p6o and an adaptive grid with 401 nodes, and for the spatial and temporal intervals $[-20, 55]$ and $[0, 800]$, respectively. The adaptive grid parameters are $a_1 = a_2 = 1$, $\alpha = 0.001$, $\beta = 10$ (no clipping), and $\Delta t_{\text{grid}} = 1$. The Dirichlet boundary conditions $u_L = u_0$ and $u_U = 0$ are applied at the lower and upper grid boundaries.

The integrals in the invariants of motion given by Equation (3.7) can be integrated analytically or numerically for the initial smooth bore profile specified at $t = 0$. The resulting invariants of motion, for the case of $u_L = u_0$ and $u_U = 0$, are summarized for later reference as $C_1^{\text{exact}} = 2.000084$, $C_2^{\text{exact}} = 0.1751279$, and $C_3^{\text{exact}} = 0.01625252$.

Numerical results from the MOL solution of the EW equation with $\mu = 1/6$ are given first in the form of a time-distance diagram in Figure 3.24 for the reduced spatial interval $[-20, 46]$, which is the main part of the entire computational domain $[-20, 55]$. The front of the initially smooth bore begins to steepen as it propagates to the right, and this front eventually breaks into an ever increasing number of undulations, one after the other, forming what is called an undular bore. This bore continues to advance in space and time as a train of oscillatory waves. A close observation of the formation of each undulation will reveal that its peak amplitude increases from an initial value of 0.10 for the initial bore to a somewhat larger amplitude at larger distances and times. This train of undulatory waves carries the mass, momentum, and energy of the initial bore forward in space and time.

A clearer view of the cross-section of the undular bore is presented in Figure 3.25, where three spatial distributions at the times $t = 0$, 300, and 600 are depicted. A large number of nodes is required to obtain an accurate solution at later times because of the increasing number of undulations. For the computational time interval of $[0, 800]$, the use of 401 grid nodes is more than sufficient, as can be seen by the node distribution in the undulations at the later time of $t = 600$. The grid nodes are well clustered in regions of large gradients and curvatures. Note that only every second node is shown for clarity. The invariants of motion were calculated during the numerical computations to help indicate if the solution is computed accurately. The

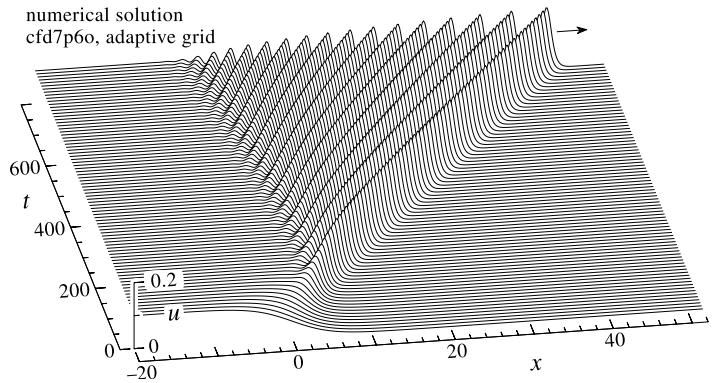


FIGURE 3.24
Formation of an undular bore.

exact values were reproduced and remained constant to four significant digits. This provides more assurance that the solution is computed accurately.

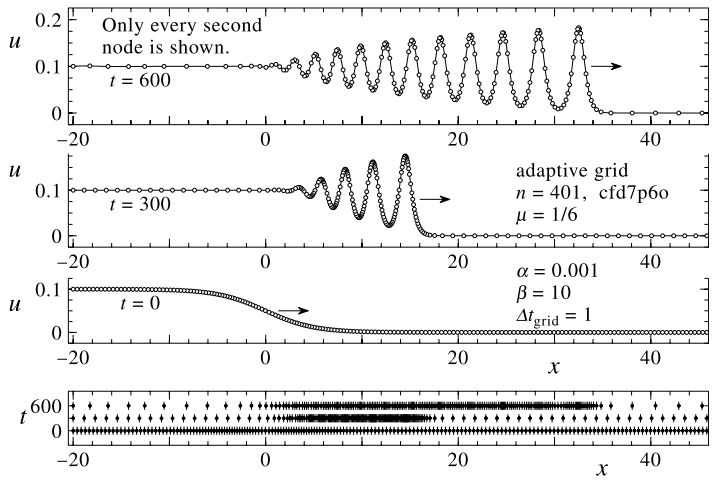


FIGURE 3.25
Cross-sections through an undular bore.

The increase in peak amplitude with time of the leading wave of the undular bore and the corresponding slightly concave trajectory are both shown in Figure 3.26. The peak amplitude of this leading wave can be clearly seen to increase rapidly at first from the initial value of 0.10 of the original bore, and then it rises more slowly to what appears as an asymptotic limit that is at least 0.182. The trajectory of the peak amplitude of the leading wave accelerates, more quickly at smaller times and then asymptotically to a final speed. The slope of the trajectory at later times, which is the

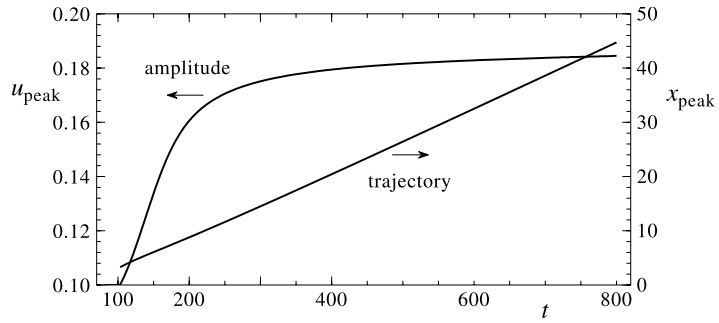


FIGURE 3.26
Amplitude growth and trajectory of the leading wave of an undular bore.

asymptotic speed of the leading wave, is about 0.061, which is about one-third of the peak amplitude.

A close inspection of the numerical data for the leading undulation shows that its shape and speed are consistent with those of a solitary wave. The following undulations have similar shapes and behaviors, but they are closely and uniformly spaced. These undulations do not separate as they propagate, as we observed for the earlier case of the breakup of an initial Gaussian pulse into a train of solitary waves.

The numerical computations by the MOL for the undular bore were rather computer intensive on the spatial and temporal domains of $[-20, 55]$ and $[0, 800]$, as compared to all of the previous problems in this study. The CPU time was 142 min on a Pentium III computer with a 500-MHz processor and LINUX operating system. For the adaptive grid with 401 nodes, the total number of time steps was 23430, the number of function evaluations was 35514, and the number of Jacobian evaluations was 12585.

3.5 Concluding Remarks

An advanced numerical MOL for solving the EW equation on uniform and adaptive grids with different finite-difference schemes (cfd7p6o and cfd5p4o) and various numbers of nodes was explained in detail in this study, primarily to help newcomers to the MOL learn these techniques more easily and quickly. Many interesting graphical solutions were computed by means of the MOL for the first time and presented to illustrate the fascinating behavior of a single solitary wave, the inelastic interaction of two solitary waves, the breakup of a Gaussian pulse into a sequence of solitary waves, and the formation of an undular bore. An equal emphasis was placed on the description of the numerical MOL subprocedures and on the illustration of interesting numerical results by means of informative diagrams.

The numerical MOL solutions presented in this study for the EW equation are more accurate and computed more efficiently than those in previous work on the EW and RLW equations for the same problems. These advantages stem primarily from the use of an adaptive grid with a high-order finite-difference scheme (cfd7p6o) and a high-order time integration (DASSL) of the DAEs, and also from enhanced techniques of interpolation of numerical data during grid adaptations.

Three objectives of this study were to implement improvements in some subprocedures of the numerical MOL. The first objective was to provide an improved technique of interpolating numerical data than the previous usage of cubic and quintic splines. The mapping of data by either spline interpolation from a previous to a newly adapted grid normally adversely reduces the accuracy of the numerical solution, although the solution may be smoothed to some advantage by the spline interpolation. This low-order spline interpolation is counterproductive to improving the solution accuracy by the implementation of high-order finite-difference schemes associated with large stencils in the MOL. In this study we interpolated numerical data by means of the quintic polynomial given by Equation (3.21), which we believe is an improvement in both accuracy and computer efficiency as compared to cubic and quintic splines. However, the underlying polynomial equation (e.g., Lagrange) associated with the finite-difference scheme should have been used for the interpolations. For example, the finite-difference scheme labeled cfd7po has a seven-point stencil with an associated sixth degree polynomial equation through seven data pairs (u vs. x). This underlying polynomial can be implemented readily for interpolation, and the accuracy of such an interpolation is then inherently consistent with the finite-difference scheme in the MOL procedure.

The second objective was to provide an improved technique of integrating numerical data than the previous approach of using Simpson's integration method. Accurate integrations are required, for example, to calculate the three invariants of motion, and Simpson's method is normally inadequate in terms of accuracy. In this study, the integrals in the invariants of motion were evaluated by using the quintic polynomial given by Equation (3.21), which yields sufficiently accurate invariants of motion. However, the underlying polynomial equation associated with the finite-difference scheme should have also been used for these integrations. This underlying polynomial can be used easily for the integrations, and the accuracy of the quadrature is then inherently consistent with the finite-difference scheme in the MOL procedure.

The third objective of this study was to improve the method of selection of values of some parameters for adaptive gridding. The use of a static adaptive grid technique in the MOL was found to be both cumbersome and time consuming for the user of the computer code. The selection of optimal values for the parameters a_1, a_2, α, β, γ, and Δt_{grid} for a particular problem was not only tedious, but it also depended to a large extent on subjective judgement. The description presented in this study on how to select appropriate values for α and β helps to simplify the selection process by illustrating their interdependence and also their relationship to $g_{\max}$ and $g_{\min}$, which are multiples of the average grid node spacing.

The selection process for values of the parameters $a_1, a_2, \alpha, \beta, \gamma$, and Δt_{grid} should, regardless of our helpful improvements, be fully automated and simply made part of an adaptive grid package. Such an automated process should be linked, hopefully, to the equidistribution of truncation errors throughout the grid. Adaptive gridding is currently implemented after a given time interval Δt_{grid}, and then the time integration in the solver `DASSL` is restarted with the order $q = 1$, progressing to higher orders up to $q = 5$ at successive time steps. On one hand, if the value of Δt_{grid} is set too small, the numerical results are less accurate due to the degraded order of the time integration. On the other hand, if Δt_{grid} is set too large, then the waves outrun their regions of clustered nodes and the numerical results become less accurate. This problem can be overcome by implementing a high-order solver that advances the solution in time of the stiff and implicit DAEs by means of a one-step method.

Acknowledgments

The financial support from the Natural Sciences and Engineering Research Council of Canada for Professors J.J. Gottlieb (Grant No. OGP0004539) and J.S. Hansen (Grant No. OGP0003663) is gratefully acknowledged. We express our deepest appreciation to Professor W.E. Schiesser of Lehigh University at Bethlehem, Pennsylvania, and Professor J.L. Bona of the University of Texas at Austin, Texas, for helpful suggestions.

References

[1] Kh.O. Abdulloev, H. Bogolubsky, and V.G. Markhankov, One more example of inelastic soliton interaction, *Phys. Lett. A,* **56**, 427–428, 1976.

[2] U.M. Ascher, and L.R. Petzold, *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations,* SIAM, Philadelphia, 1988.

[3] T.B. Benjamin, J.L. Bona, and J.L. Mahoney, Model equations for long waves in nonlinear dispersive media, *Phil. Trans. Roy. Soc. Lond. A,* **272**, 47–78, 1995.

[4] Yu. Berezin and V.I. Karpman, Nonlinear evolution of disturbances in plasmas and other dispersive media, *Soviet Physics JETP,* **24**, 1049–1056, 1967.

[5] J.L. Bona, W.G. Pritchard, and L.R. Scott, Numerical schemes for a model for nonlinear dispersive waves, *J. Comput. Phys.,* **60**, 167–186, 1985.

[6] K.E. Brenan, S.L. Campbell, and L.R. Petzold, *The Numerical Solution of Initial Value Problems in Differential-Algebraic Equations,* Elsevier Science Publishing Co., 1989.

[7] R.P. Brent, *Algorithms for Minimization without Derivatives,* Prentice-Hall, Englewood Cliffs, NJ, 1973.

[8] B. Fornberg, Calculation of weights in finite difference formulas, *SIAM Rev.,* **40**(3), 685–691, 1998.

[9] G.E. Forsythe, M.A. Malcolm, and C.B. Moler, *Computer Methods for Mathematical Computations,* Prentice-Hall, Englewood Cliffs, NJ, 1977.

[10] L.R.T. Gardner, G.A. Gardner, A. Ayoub, and N.K. Amein, Simulations of the EW undular bore, *Commun. Numer. Meth. Engng.,* **13**, 583–592, 1997.

[11] L.R.T. Gardner, G.A. Gardner, and A. Dogan, A least-squares finite element scheme for the RLW equation, *Commun. Numer. Meth. Engng.,* **12**, 795–804, 1996.

[12] L.R.T. Gardner and G.A. Gardner, Solitary waves of the regularized long-wave equation, *J. Comput. Phys.,* **91**, 441–459, 1990.

[13] C.W. Gear, The simultaneous numerical solution of differential-algebraic equations, *IEEE Trans. Circuit Theory,* CT–18:89–95, 1971.

[14] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems,* Springer-Verlag, Berlin, 1991.

[15] S. Hamdi, J.J. Gottlieb, and J.S. Hansen, Computation of blow-up solutions of the generalized Korteweg-de Vries equation using adaptive method of lines, in J. Militzer, ed., *7th Annual Conference of the CFD Society of Canada,* 921–926, June 1999.

[16] S. Hamdi, J.J. Gottlieb, and J.S. Hansen, Numerical solution of the equal width wave equation using the method of lines, in *8th Annual Conference of the CFD Society of Canada,* 129–134. Centre de Recherche en Calcul Appliqué, 2000.

[17] A.C. Hindmarsh, ODEPACK: A systematized collection of ODE solvers, in R.S. Steplemman, ed., *Scientific Computing,* 55–64. North-Holland, Amsterdam, 1983.

[18] P.C. Jain, R. Shankar, and T.V. Singh, Numerical solutions of regularized long wave equation, *Commun. Numer. Meth. Engng.,* **9**, 579–586, 1993.

[19] D.J. Korteweg and G. de Vries, On the change of form of long waves advancing in a rectangular canal and a new type of long stationary wave, *Phil. Mag.,* **39**, 422–443, 1895.

[20] J.C. Lewis and J.A. Tjon, Resonant production of solitons in the RLW equation, *Phys. Lett. A,* **73**, 275–279, 1979.

[21] P.J. Morrison, J.D. Meiss, and J.R. Carey, Scattering of RLW solitary waves, *Physica D,* **11**, 324–336, 1984.

[22] P.J. Olver, Euler operators and conservation laws of the BBM equations, *Math. Proc. Camb. Phil. Soc.,* **85**, 143–159, 1979.

[23] D.H. Peregrine, Calculations of the development of an undular bore, *J. Fluid Mech.,* **25**, 321–330, 1966.

[24] L.R. Petzold, A Description of DASSL: A differential/algebraic system solver, in *Scientific Computing,* R.S. Stepleman et al., eds., North Holland, Amsterdam, 1983.

[25] M.A. Revilla, Simple time and space adaptation in one-dimensional evolutionary partial differential equation, *Int. J. Numer. Methods Eng.,* **23**, 2263–2275, 1986.

[26] A.R. Santarelli, Numerical analysis of the regularized long-wave equation: Anelastic collision of solitary waves, *Nuovo Cimento B,* **46**, 179–188, 1978.

[27] J.M. Sanz-Serna and I. Christie, A simple adaptive technique for nonlinear wave problems, *J. Comput. Phys.,* **67**, 348–360, 1986.

[28] P. Saucez, A.V. Wouwer, and W.E. Schiesser, Some observations on a static spatial remeshing method based on equidistribution principles, *J. Comput. Phys.,* **128**, 274–288, 1996.

[29] P. Saucez, A.V. Wouwer, and W.E. Schiesser, An adaptive method of lines solution of the Korteweg-de Vries equation, *Computers Math. Applic.,* **35**, 13–25, 1998.

[30] W.E. Schiesser, Method of lines solution of the Korteweg-de Vries equation, *Computers Math. Applic.,* **28**, 147–154, 1994.

[31] A.B. White, On the numerical solution of initial/boundary-value problems in one space dimension, *SIAM J. Numer. Anal.,* **19**, 683–697, 1982.

[32] G.B. Whitham, *Linear and Nonlinear Waves,* John Wiley & Sons, New York, 1974.

Chapter 4

Adaptive Method of Lines for Magnetohydrodynamic PDE Models

P. A. Zegeling and R. Keppens

4.1 Introduction

An adaptive grid technique for use in the solution of multi-dimensional time-dependent PDEs is applied to several magnetohydrodynamic model problems. The technique employs the method of lines and can be viewed both in a continuous and semi-discrete setting. By using an equidistribution principle, it has the ability to track individual features of the physical solutions in the developing plasma flows. Moreover, it can be shown that the underlying grid varies smoothly in time and space. The results of several numerical experiments are presented which cover many aspects typifying nonlinear magneto-fluid dynamics.

Many interesting phenomena in plasma fluid dynamics can be described within the framework of magneto-hydrodynamics (MHD). Numerical studies in plasma flows frequently involve simulations with highly varying spatial and temporal scales. As a consequence, numerical methods on uniform grids may be inefficient to use, since a very large number of grid points is needed to resolve the spatial structures, such as shocks, contact discontinuities, shear layers, or current sheets. For the efficient study of these phenomena, we require adaptive grid methods which automatically track and spatially resolve one or more of these structures.

Over the years a large number of adaptive grid methods have been proposed for time-dependent PDE models. Two main strategies of adaptive grid methods can be distinguished, namely, static-regridding methods and moving-grid or dynamic-regridding methods. In static-regridding methods (denoted by h-refinement) the location of nodes is fixed. A method of this type adapts the grid by adding nodes where they are necessary and removing them when they are no longer needed. The refinement or de-refinement is controlled by error estimates or error monitor values (which have no resemblance with the true numerical error). Recent examples of these methods are described in [16, 4, 20, 7]. In dynamic-regridding methods (denoted by

r-refinement) nodes are moving continuously in the space-time domain, like in classical Lagrangian methods, and the discretization of the PDE is coupled with the motion of the grid. Examples can be found in [3, 18, 19, 5, 8].

In this chapter we follow the second approach. The adaptive grid method is based on a semi-discretization of a fourth-order PDE for the grid variable and is being coupled to the original MHD model re-written in a new coordinate system. We use the so-called method-of-lines technique (MOL) [11]: first we discretize the PDEs in the space direction using a finite-difference approximation, so as to convert the PDE problem into a system of stiff, ordinary differential equations (ODEs) with time as independent variable. The discretization in time of this stiff ODE system then yields the required fully discretized scheme.

The layout of the chapter is as follows. In the next section we present the full set of MHD equations and their physical meaning. In Section 4.3 we describe the restriction to the one-dimensional situation and the adaptive grid method. The moving grid is defined as the solution of an adaptive grid PDE. Numerical experiments are shown for three different cases: an MHD-shocktube model, a problem describing Shear-Alfvén wave propagation, and an oscillating plasma sheet in vacuum surroundings. Section 4.4 discusses the essential elements for generalizing the MOL approach to multi-dimensional MHD simulations. We evaluate different means for 2D grid adaptation on kinematic magnetic field models, with particular attention paid to the solenoidal condition on the magnetic field vector. Section 4.5 lists our conclusions and presents an outlook to future work.

4.2 The Equations of Magnetohydrodynamics

The MHD equations govern the dynamics of a charge-neutral "plasma." Just like the conservative Euler equations provide a continuum description for a compressible gas, the MHD equations express the basic physical conservation laws to which a plasma must obey. Because plasma dynamics is influenced by magnetic fields through the Lorentz-force, the needed additions in going from hydrodynamic to magnetohydrodynamic behavior is a vector equation for the magnetic field evolution and extra terms in the Euler system that quantify the magnetic force and energy density.

Using the conservative variables density ρ, momentum density $\mathbf{m} \equiv \rho\mathbf{v}$ (with velocity $\mathbf{v}$), magnetic field $\mathbf{B}$, and total energy density e, the ideal MHD equations can be written as follows (cfr. [2, 13, 15]):
Conservation of mass:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0 \,. \tag{4.1}$$

Conservation of momentum:

$$\frac{\partial (\rho \mathbf{v})}{\partial t} + \nabla \cdot (\rho \mathbf{v}\mathbf{v} - \mathbf{B}\mathbf{B}) + \nabla p_{tot} = 0 \,. \tag{4.2}$$

Conservation of energy:

$$\frac{\partial e}{\partial t} + \nabla \cdot (e\mathbf{v} + \mathbf{v}p_{tot} - \mathbf{B}\mathbf{B} \cdot \mathbf{v}) = 0 \ \left[+ \epsilon_m (\nabla \times \mathbf{B})^2 \right]. \tag{4.3}$$

Magnetic field induction equation:

$$\frac{\partial \mathbf{B}}{\partial t} + \nabla \cdot (\mathbf{v}\mathbf{B} - \mathbf{B}\mathbf{v}) = 0 \ [+ \epsilon_m \Delta \mathbf{B}] . \tag{4.4}$$

In (4.2) and (4.3) the total pressure p_{tot} consists of both a thermal and a magnetic contribution as given by

$$p_{\text{tot}} = p + \frac{\mathbf{B}^2}{2}, \ \text{ where } \ p = (\gamma - 1)\left(e - \rho \frac{\mathbf{v}^2}{2} - \frac{\mathbf{B}^2}{2} \right) \tag{4.5}$$

is the thermal pressure. This set of equations must be solved in conjunction with an important condition on the magnetic field **B**, namely the non-existence of magnetic "charge" or monopoles. Mathematically, it is easily demonstrated that this property can be imposed as an initial condition alone, since

$$\nabla \cdot \mathbf{B}|_{t=0} = 0 \Longrightarrow \nabla \cdot \mathbf{B}|_{t \geq 0} = 0 . \tag{4.6}$$

In multi-dimensional numerical MHD, the combined spatio-temporal discretization may not always ensure this conservation of the solenoidal character of the vector magnetic field. When dealing with a two-dimensional model problem for **B** evolution in Section 4.4.2.3.3, we pay particular attention to this matter.

The terms between brackets in Equations (4.3) and (4.4) extend the ideal MHD model with the effects of Ohmic heating due to the presence of currents. With the resistivity $\epsilon_m \neq 0$, we then solve the resistive MHD equations. Likewise, extra non-conservative source terms may be added to the momentum and energy equation for describing viscous effects. In the numerical experiments, we resort to artificial diffusive terms which can be thought of as approximations representing these actual physical phenomena.

4.3 Adaptive Grid Simulations for 1D MHD

4.3.1 The MHD Equations in 1D

If we restrict the MHD model (4.1)–(4.6) to 1.5D, i.e., variations in one spatial x-dimension but possibly non-vanishing y-components for the vector quantities with $\partial/\partial y = 0$, we obtain a 5-component PDE system which is formally written as

$$\frac{\partial \mathbf{\Psi}}{\partial t} + \frac{\partial \mathbf{F}(\mathbf{\Psi})}{\partial x} = 0, \quad x \in [x_L, x_R] , \quad t > 0 . \tag{4.7}$$

Here, $\mathbf{\Psi} = (\rho, m_1, m_2, B_2, e)^T$ is the vector of conserved variables (m_1, m_2 are now the x- and y-components of the momentum vector and B_2 denotes the y-component of the magnetic induction), with the flux-vector $\mathbf{F} = (F_1, \ldots, F_5)^T$ given by

$$\begin{aligned}
F_1 &= m_1 \,, \\
F_2 &= \frac{m_1^2}{\rho} - \bar{B}_1^2 + (\gamma - 1)e - (\gamma - 1)\frac{m_1^2 + m_2^2}{2\rho} + (2 - \gamma)\frac{\bar{B}_1^2 + B_2^2}{2} \,, \\
F_3 &= \frac{m_1 m_2}{\rho} - \bar{B}_1 B_2 \,, \\
F_4 &= B_2 \frac{m_1}{\rho} - \bar{B}_1 \frac{m_2}{\rho} \,, \\
F_5 &= \frac{m_1}{\rho}\left(\gamma e - (\gamma - 1)\frac{m_1^2 + m_2^2}{2\rho} + (2 - \gamma)\frac{\bar{B}_1^2 + B_2^2}{2}\right) \\
&\quad - \bar{B}_1\left(\bar{B}_1 \frac{m_1}{\rho} + B_2 \frac{m_2}{\rho}\right) \,.
\end{aligned}$$

The constant γ is the ratio of specific heats and $\bar{B}_1$ is the constant first component of the magnetic induction vector. Indeed, in 1D model problems, the vanishing divergence of the magnetic field is thereby trivially satisfied. The remaining set of 5 PDEs given by (4.7) constitutes the physical model used for the 1D MHD simulations found below. We first indicate how this system is further manipulated and discretized to solve simultaneously for the adaptive grid with its corresponding solution.

4.3.2 The Adaptive Grid Method in One Space Dimension

4.3.2.1 Transformation of Variables

It is common practice in adaptive grid generation to submit the PDE model to a coordinate transformation. Ideally, the mapping should be chosen such that in the new coordinate variables, the discretization error in the numerical solution is much smaller than in the original variables. In the new variables the PDEs are then simply uniformly partitioned. In general, applying a transformation

$$\xi = \xi(x, t) \in [0, 1], \quad \theta = t \,, \tag{4.8}$$

to the system (4.7) gives after some elementary calculations

$$x_\xi \mathbf{\Psi}_\theta - \mathbf{\Psi}_\xi x_\theta + (\boldsymbol{F}(\mathbf{\Psi}))_\xi = 0 \,. \tag{4.9}$$

Different choices for the transformation are possible. The coordinate transformation used in this chapter is implicitly defined as the solution of a special partial differential equation (see Section 4.3.2.2). Even without knowing this mapping, we can already semi-discretize (4.9) by noting that in the ξ-variable a uniform grid ($\xi_i = i/N, i = 0, \ldots, N$) is imposed. Using central finite differences, the PDEs (4.9) become a

system of ODEs as follows:

$$(x_{i+1} - x_{i-1}) \frac{\mathrm{d}\boldsymbol{\Psi}_i}{\mathrm{d}\theta} - (\boldsymbol{\Psi}_{i+1} - \boldsymbol{\Psi}_{i-1}) \frac{\mathrm{d}x_i}{\mathrm{d}\theta} + \mathbf{F}(\boldsymbol{\Psi}_{i+1}) - \mathbf{F}(\boldsymbol{\Psi}_{i-1}) = 0 \qquad \forall i \ . \tag{4.10}$$

Note that we have multiplied (4.10) by the factor $2\Delta\xi$, which has a constant value by definition.

4.3.2.2 The Adaptive Grid PDE

We implicitly define the transformation $\xi(x, t)$, and thereby the grid distribution, as the solution of the following time-dependent "adaptive grid PDE"

$$\left[\left(\mathcal{S}\left(x_\xi\right) + \tau x_{\xi\theta}\right)\mathcal{W}\right]_\xi = 0 \ . \tag{4.11}$$

The parameter $\tau > 0$ in (4.11) is a temporal smoothing parameter, the operator $\mathcal{S}$ incorporates a spatial smoothing in a manner detailed below, while

$$\mathcal{W} = \sqrt{1 + \sum_{j=1}^{5} \alpha_j \left(\boldsymbol{\Psi}_x^{(j)}\right)^2}$$

is a weight function that depends on the derivatives of the different components $\boldsymbol{\Psi}^{(j)}$. The parameters α_j are termed "adaptivity parameters." Their values can be chosen to emphasize, if necessary, particular variables in the PDE model (such as the density or a magnetic field component for MHD problems). In full, the smoothing operator $\mathcal{S}$ in (4.11) is defined by

$$\mathcal{S} = \mathcal{I} - \sigma(\sigma + 1)(\Delta\xi)^2 \frac{\partial^2}{\partial\xi^2}, \tag{4.12}$$

where $\sigma > 0$ is a spatial smoothing parameter and $\mathcal{I}$ the identity operator. This specific choice of transformation has several desirable properties, which are briefly discussed in Section 4.3.2.3.

Since the adaptive grid PDE is fourth order in space, it is clear that we need four boundary conditions and one initial condition. An obvious choice is to take two Dirichlet and two Neumann conditions:

$$x|_{\xi=0} = x_L, \ x|_{\xi=1} = x_R, \ x_\xi|_{\xi=0} = x_\xi|_{\xi=1} = 0 \ . \tag{4.13}$$

At initial time $\theta = 0$, the grid is uniformly distributed and is thus given by $x|_{\theta=0} = x_L + (x_R - x_L)\xi$.

4.3.2.3 Properties of the Adaptive Grid

It can be shown that the determinant of the Jacobian of the transformation implied by (4.11) and (4.12) satisfies the mesh-consistency condition

$$\mathcal{J} = x_\xi > 0 \quad \forall\, \theta \in [0, T], \ \ \forall \xi \in [0, 1] \ , \tag{4.14}$$

which in discretized form reads (since $\Delta\xi$ is a constant)

$$\Delta x_i(\theta) := x_i(\theta) - x_{i-1}(\theta) > 0 \quad \forall\, \theta \in [0, T]\,. \tag{4.15}$$

In other words, relation (4.15) states that the grid points can never cross one another (see Chapter 4 in [18] and [5] for more details and proofs of these results). Another important property of the transformation satisfying (4.11) and (4.12) is the following:

$$\left|\frac{x_{\xi\xi}}{x_\xi}\right| \leq \frac{1}{\sqrt{\sigma(\sigma+1)\Delta\xi}}\,, \tag{4.16}$$

which may be translated in discrete terms as

$$\frac{1}{1+\frac{1}{\sigma}} \leq \frac{\Delta x_{i+1}(\theta)}{\Delta x_i(\theta)} \leq 1 + \frac{1}{\sigma} \quad \forall\, \theta \geq 0, \quad \forall i\,. \tag{4.17}$$

This property expresses "local quasi-uniformity" and means that the variation in successive grid cells can be controlled by the parameter σ at every point in time.

A reasonable choice for the temporal smoothing parameter is $0 < \tau \leq 10^{-3} \times$ {timescale in PDE model}, while the spatial smoothing parameter is typically $\sigma = \mathcal{O}(1)$. The adaptivity parameters are normally taken $\alpha_j = \mathcal{O}(1)$ (see also [18]), but may need re-scaling depending on the x-range and the magnitude of the individual $\boldsymbol{\Psi}^{(j)}$. Note that if we switch off all smoothing in (4.11), we obtain the well-known "equidistribution principle" which has both a continuous and discrete variant given by the formulae

$$\tau = \sigma = 0 \quad \Rightarrow \quad \left[x_\xi\, \mathcal{W}\right]_\xi = 0 \quad \forall\, \theta \in [0, T] \quad \Leftrightarrow \quad \xi(x,t) = \frac{\int_{x_L}^{x} \mathcal{W}\, \mathrm{d}\bar{x}}{\int_{x_L}^{x_R} \mathcal{W}\, \mathrm{d}\bar{x}}\,, \tag{4.18}$$

or in discretized form (using the midpoint rule for integration)

$$\Delta x_i \cdot \mathcal{W}_{i-1/2} = \text{constant} \quad \forall\, \theta \in [0, T]\,. \tag{4.19}$$

4.3.2.4 Semi-Discretization of the Adaptive Grid PDE

The adaptive grid PDE (4.11) is semi-discretized using central-differences to obtain

$$\left[\tilde{\Delta} x_{i+1} + \tau \frac{\mathrm{d}\Delta x_{i+1}}{\mathrm{d}\theta}\right] \mathcal{W}_{i+1/2} - \left[\tilde{\Delta} x_i + \tau \frac{\mathrm{d}\Delta x_i}{\mathrm{d}\theta}\right] \mathcal{W}_{i-1/2} = 0 \quad \forall i\,, \tag{4.20}$$

where

$$\tilde{\Delta} x_i = \Delta x_i - \sigma(\sigma+1)\left(\Delta x_{i+1} - 2\Delta x_i + \Delta x_{i-1}\right)\,, \tag{4.21}$$

which is a discretization of $\mathcal{S}(x_\xi)$ about the gridpoint $\xi_i = \frac{i}{N}$.

The discrete version of the weight function becomes

$$\mathcal{W}_{i-1/2} = \sqrt{1 + \sum_{j=1}^{5} \alpha_j (\Psi^{(j)}_{x,i-1/2})^2}, \quad \text{with} \quad \Psi^{(j)}_{x,i-1/2} = \frac{\Psi^{(j)}_i - \Psi^{(j)}_{i-1}}{\Delta x_i} \,. \tag{4.22}$$

The grid equations (4.20) and (4.21) are related to the adaptive grid described in [3]. The differences between (4.20) and (4.21) and [3] mainly consist of the use of cell-lengths instead of point concentrations and not applying the operator $\mathcal{S}$ to the $\frac{\mathrm{d}\Delta x}{\mathrm{d}\theta}$-terms. In compact notation the adaptive grid ODE system (4.20) reads

$$\tau \, \mathcal{B}(\mathbf{X}, \mathbf{\Psi}, \sigma, \boldsymbol{\alpha}) \frac{\mathrm{d}\mathbf{X}}{\mathrm{d}\theta} = \mathcal{H}(\mathbf{X}, \mathbf{\Psi}, \sigma, \boldsymbol{\alpha}) \,, \tag{4.23}$$

where

$$\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_5)^T \,,$$

and $\mathbf{\Psi}$ and $\mathbf{X}$ contain the discretized MHD components and the grid points, respectively. After coupling this system to the semi-discretized PDE system (4.10), a large, stiff, banded, nonlinear ODE system is obtained. System (4.23) has bandwidth 12. This can be derived easily by working out (4.20) in terms of the x_i's and realizing that the unknown vector of the complete ODE system is written as $(\ldots, \Psi^{(1)}_i, \Psi^{(2)}_i, \ldots, \Psi^{(5)}_i, x_i, \Psi^{(1)}_{i+1}, \ldots)^T$. For the time-integration of this system, the ODE-package DASSL [10] with the (implicit) BDF-methods up to order 5 will be used. DASSL uses a direct solver for the linear systems and exploits the banded form of the equations in the Jacobian formation and numerical linear algebra computations. Numerical differencing for Jacobians in the Newton-process is being used. The time-stepping error tolerance is denoted by *tol* and will be specified at the experiments.

4.3.3 Numerical Results

In what follows, we apply the adaptive MOL approach to three 1.5D MHD model problems which are chosen to cover significantly diverse challenges typically encountered in numerical MHD simulations. We solve a standard Riemann problem to address the performance of the MOL technique as a shock-capturing and shock-tracing method, we simulate linear shear Alfvén waves which are non-compressive perturbations with a specific polarization, and we model a plasma-"vacuum" configuration which poses numerical difficulties to keep density and pressure positive throughout the domain. We explicitly compare the obtained adaptive grid solutions with high resolution reference solutions on static, uniform grids. These reference solutions are all calculated with the Versatile Advection Code [14] (VAC, see `http://www.phys.uu.nl/~toth`), and if not stated otherwise, use 1000 grid points and the (approximate) Riemann-solver based total variation diminishing (TVD) scheme with "minmod" limiting. This shock-capturing, one-step TVD scheme is actually one out of six high resolution spatial discretization schemes available in VAC, and has demonstrated to be the most accurate and efficient discretization method on

a large variety of HD and MHD problems [13]. Specifically, the effects of dispersion and diffusion will essentially be minimal in the reference solutions, and this should be kept in mind when comparing them with the adaptive simulation results. For the latter, all experiments used values for the smoothing parameters $\tau = 10^{-5}$, $\sigma = 2$, and tol $= 10^{-6}$. The adaptivity constants will be specified and motivated per model.

4.3.3.1 MHD-Shocktube Model

This test problem by [2] and also used in [13] has evolved into a benchmark for MHD codes. The initial Riemann problem separates a high density and high thermal pressure left state from a low density and low pressure right state with the magnetic field lines reflected over the normal to the discontinuity line $x = 0.5$ in the $x - y$ plane. The sudden expansion of the left state produces a reversedly propagating fast rarefaction fan and a slow compound wave, a rightwardly advected contact discontinuity, and a right-moving slow shock and fast rarefaction fan. The compound wave is a combination of a slow shock with a slow rarefaction attached to it.

Specifically, the problem is set up in the space-interval $x \in [0, 1]$, while we simulate for times $t \in [0, 0.1]$. In the adaptive approach, we use 250 grid points, and added artificial diffusion terms $\epsilon \frac{1}{\mathcal{J}} \frac{\partial}{\partial \xi} [\frac{1}{\mathcal{J}} \frac{\partial \Psi^{(j)}}{\partial \xi}]$ to all but the mass-conservation law with diffusion coefficients $\epsilon = 0.0001$. Since all developing dynamic features have an associated density variation, we set the adaptivity parameter $\alpha_1 = 1$, while all other parameters $\alpha_i = 0, \quad (i = 2, \ldots, 5)$. Furthermore,

$$\gamma = 2, \quad \bar{B}_1 \equiv 0.75$$

$$\rho|_{t=0} = \begin{cases} 1 & \text{for } x \in [0, 0.5] \\ 0.125 & \text{for } x \in [0.5, 1] \end{cases}$$

$$m_1|_{t=0} = m_2|_{t=0} = 0$$

$$B_2|_{t=0} = \begin{cases} 1 & \text{for } x \in [0, 0.5] \\ -1 & \text{for } x \in [0.5, 1] \end{cases}$$

$$e|_{t=0} = \begin{cases} 1.78125 & \text{for } x \in [0, 0.5] \\ 0.88125 & \text{for } x \in [0.5, 1] \end{cases}$$

Homogeneous Neumann boundary conditions are used for all components. In Figure 4.1, we compare the density profile at $t = 0.1$ from three simulations: a VAC solution on a 250-point static grid, the adaptive solution with the same amount of grid points, and the true reference VAC solution exploiting 1000 points (both VAC solutions used a Courant number of 0.8). Clearly, the MOL technique is superior to the VAC solution that uses the same amount of grid points, and the accuracy of the adaptive method is identical to the high resolution reference solution. We thus save a factor of 4 in grid resolution as compared to a uniform grid. In Figure 4.2, we plot at left the $v_1 := m_1/\rho$ velocity profile at the same time for both the adaptive and the reference solution, while the grid history for $t \in [0, 0.1]$ is shown at right. Note that the adaptive solution is fairly dispersive for this particular variable. The grid history demonstrates how the initial discontinuity causes an immediate clustering of

grid points in the region of interest and that the emerging shock features are nicely traced individually.

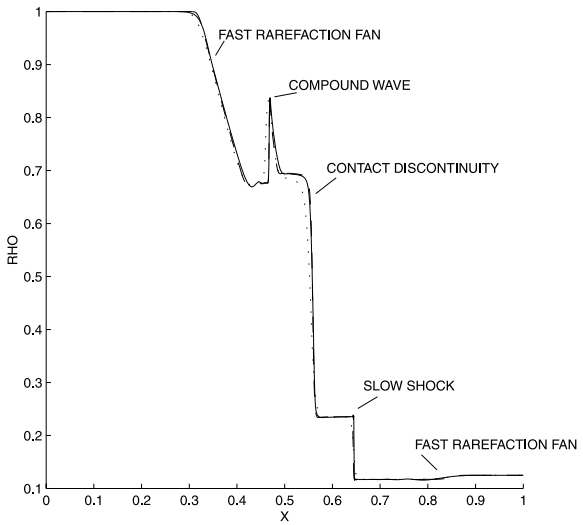


FIGURE 4.1
Density at $t = 0.1$ for the magnetic shocktube model. We compare two static grid reference solutions, one with 250 grid points (dots) and one for 1000 grid points (dashed), with an adaptive MOL solution exploiting 250 points (solid).

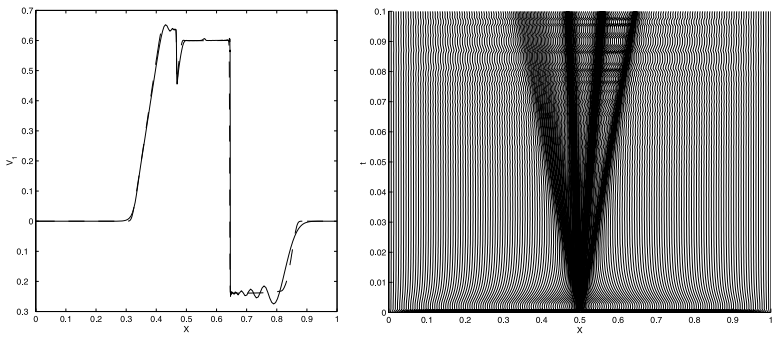


FIGURE 4.2
Left panel: v_1 component of the velocity $t = 0.1$ for both the reference (dashed) and the MOL solution (solid). Right panel: grid history (tracing x-positions of grid points as a function of time t) for the magnetic shocktube model.

4.3.3.2 Shear-Alfvén Waves

This test problem was described by [12] and also used in [13] for their evaluation of different discretization schemes. A homogeneous, uniformly magnetized plasma state is perturbed with a localized velocity pulse transverse ($v_2 := m_2/\rho \neq 0$) to the horizontal (x-direction) magnetic field. This evolves into two oppositely traveling Alfvén waves that only have associated $v_2 := m_2/\rho$ and B_2 perturbations. The complete problem setup is as follows.

We take $x \in [0, 3]$ and time-interval $t \in [0, 0.8]$, together with artificial diffusion coefficients (except for the mass-conservation law) $\epsilon = 0.0001$. Because we only expect transverse vector components, we set the adaptivity parameters $\alpha_3 = \alpha_4 = 1e + 8$ with all other $\alpha_1 = \alpha_2 = \alpha_5 = 0$. The high values for α_3 and α_4 are a consequence of a scaling effect in the weight function. Since $(\frac{\Delta B_2}{\Delta x})^2 = \mathcal{O}(10^{-8})$ occurs in $\mathcal{W}$, it is natural to choose the adaptivity parameter(s) $\mathcal{O}(10^8)$ to balance the different terms. The number of grid points for this model is taken equal to 250. Physical parameters and initial conditions for this model are:

$$\gamma = 1.4, \quad \bar{B}_1 \equiv 1$$

$$\rho|_{t=0} = 1$$

$$m_1|_{t=0} = 0$$

$$m_2|_{t=0} = \begin{cases} 10^{-3} & \text{for } x \in [1, 2] \\ 0 & \text{elsewhere} \end{cases}$$

$$B_2|_{t=0} = 0$$

$$e|_{t=0} = \begin{cases} 0.5000005025 & \text{for } x \in [1, 2] \\ 0.5000000025 & \text{elsewhere} \end{cases}$$

Homogeneous Neumann boundary conditions hold for all components.

Figure 4.3 shows the B_2 component of the magnetic induction at $t = 0.8$ from both the MOL and the reference solution. In the right panel, the grid history is shown. The solution again compares favorably to the high resolution static grid simulation, only slightly worsened by dispersion. The grid history shows how the original single pulse separates into two oppositely traveling signals. In Figure 4.4, we compare the errors present for both the reference solution and the adaptive one: ideally the density should remain constant. Noting the large difference in scales, the MOL approach succeeds better in minimizing the density variations. In fact, we used a Courant number of 0.4 for the reference solution in order to suppress these errors somewhat. For the reference result, they are due to the small thermal pressure ($p = 10^{-9}$) which creates roundoff problems within the Riemann solver used (see also [13]). Indeed, when switching to the non-Riemann solver based TVD Lax-Friedrichs discretization in VAC, these errors essentially disappear. Although the MOL solution seems better judged from the controlled density variations, it fails to maintain the positivity of the thermal pressure for this example.

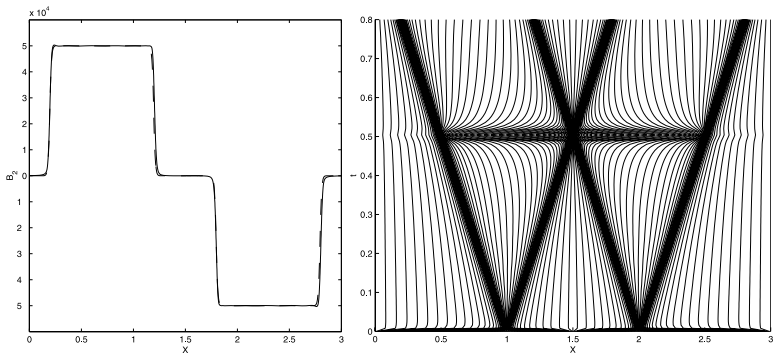


FIGURE 4.3
Left panel: y-component of magnetic induction for the Shear-Alfvén problem at $t = 0.8$, again from a 1000-point reference solution (dashed) with a 250 MOL solution. Right panel: grid history for the adaptive simulation.

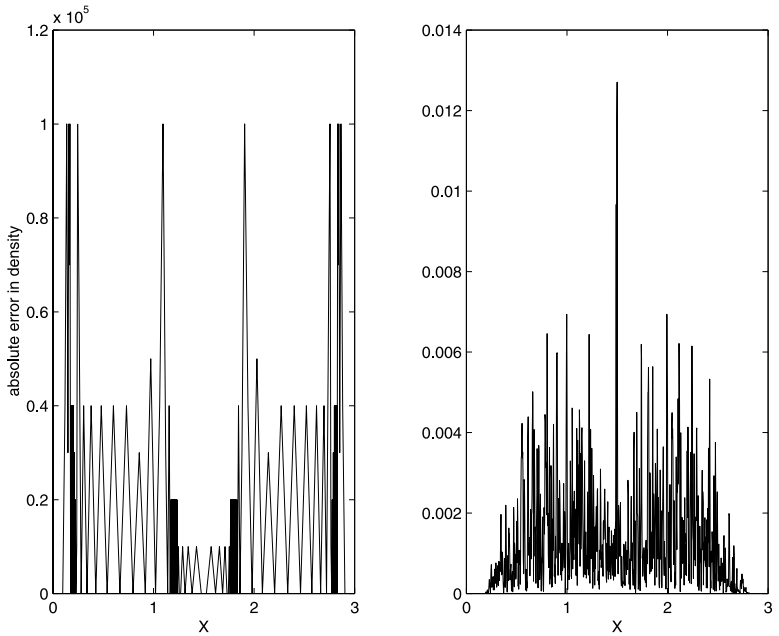


FIGURE 4.4
Comparison of the errors in the density profile for the reference MOL approach (left) and the TVD result (right). Note the different scales on the ρ-axes.

4.3.3.3 Oscillating Plasma Sheet

This test model was introduced recently in [15] as a typical case where an implicit time integration strategy is more efficient than explicit methods. A sheet of high density and pressure is surrounded by a magnetized "vacuum." The vacuum is modeled as a low density, low pressure plasma so that the plasma-vacuum interface is prone to introduce non-physical negative density and/or pressure fluctuations. We set up an initial total pressure imbalance across the sheet by prescribing a uniform, sheet-aligned magnetic field of different magnitude in the left and right vacuum region. With ideally conducting wall boundary conditions at some distance away from the sheet boundaries, this results in a magnetically controlled oscillation of the sheet as a whole due to alternate compressions and rarefactions of the vacuum magnetic fields on either side.

Specifically, for $x \in [0, 1]$, time $t \in [0, 2]$, we now use artificial diffusion coefficients $\epsilon = 0.001$ for momentum, energy, and magnetic field, while it was even necessary for stability reasons to introduce an artificial diffusion term in the mass-conservation law with $\epsilon = 10^{-5}$. We took as adaptivity parameters $\alpha_i = 1$ ($i = 1, \ldots, 5$) since there is no particular component which should be emphasized (we could perhaps take $\alpha_3 = 0$ since there will be no v_2 motion induced aligned with the sheet). The MOL solution employed 350 grid points. In summary

$$\gamma = 1.4, \quad \bar{B}_1 \equiv 0$$

$$\rho|_{t=0} = \begin{cases} 10^{-3} & \text{for } x \in [0, 0.45] \\ 1 & \text{for } x \in [0.45, 0.55] \\ 10^{-3} & \text{for } x \in [0.55, 1] \end{cases}$$

$$m_1|_{t=0} = m_2|_{t=0} = 0$$

$$B_2|_{t=0} = \begin{cases} 1.1 & \text{for } x \in [0, 0.45] \\ 0.6 & \text{for } x \in [0.45, 0.55] \\ 1.0 & \text{for } x \in [0.55, 1] \end{cases}$$

$$e|_{t=0} = \begin{cases} 0.60525 & \text{for } x \in [0, 0.45] \\ 0.98025 & \text{for } x \in [0.45, 0.55] \\ 0.50025 & \text{for } x \in [0.55, 1] \end{cases}$$

Homogeneous Neumann boundary conditions hold for all components, except for momentum in the x-direction for which $m_1|_{\partial\Omega} = 0$.

In Figure 4.5 the density at time $t = 2$, and the grid history until that time is shown. The density panel again compares the adaptive solution with a reference result (with Courant number 0.8), and it can be seen that the solution is somewhat influenced by the higher (artificial) diffusion imposed. From the grid history, we conclude that the timeframe shown is a little over two "periods" of the induced oscillation, which is in agreement with the estimated period 0.97 as listed in [15]. Note how the MOL technique nicely succeeds in tracing the waving motion of the sheet boundaries. In contrast with the previous example, the adaptive method is able to maintain the positivity of the thermal pressure for this case.

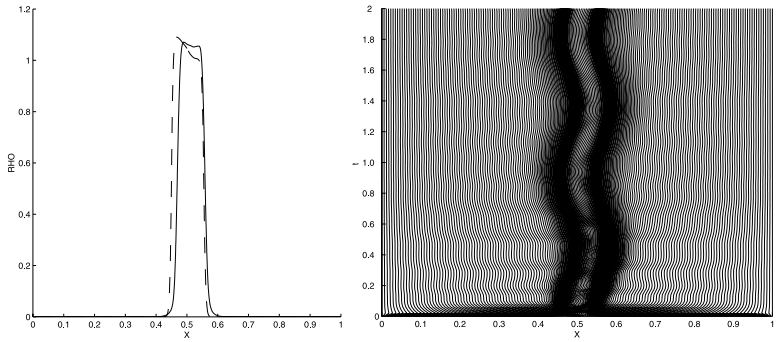


FIGURE 4.5
Density at $t = 2$ and grid history until that time for the oscillating plasma sheet. In the left panel, the MOL solution (solid) is again compared with a 1000-grid point reference solution.

4.4 Towards 2D MHD Modeling

4.4.1 2D Magnetic Field Evolution

In contrast to the 1D MHD case from above, multi-dimensional MHD simulations face a non-trivial task when advancing a magnetic field configuration forward in time while ensuring the property $\nabla \cdot \mathbf{B} = 0$. The core problem is represented by the induction equation (4.4), alternatively written as

$$\frac{\partial \mathbf{B}}{\partial t} = \nabla \times (\mathbf{v} \times \mathbf{B}) + \epsilon_m \Delta \mathbf{B} \tag{4.24}$$

with ϵ_m the resistivity $\epsilon_m \geq 0$. In two space dimensions, setting $\mathbf{B} = (B_1, B_2, 0)$, we obtain the following system of PDEs,

$$\frac{\partial B_1}{\partial t} = \epsilon_m \Delta B_1 + v_1 \frac{\partial B_2}{\partial y} - v_2 \frac{\partial B_1}{\partial y} + B_2 \frac{\partial v_1}{\partial y} - B_1 \frac{\partial v_2}{\partial y} , \tag{4.25}$$

$$\frac{\partial B_2}{\partial t} = \epsilon_m \Delta B_2 - v_1 \frac{\partial B_2}{\partial x} + v_2 \frac{\partial B_1}{\partial x} - B_2 \frac{\partial v_1}{\partial x} + B_1 \frac{\partial v_2}{\partial x} , \tag{4.26}$$

together with the property $\nabla \cdot \mathbf{B} = 0$. This system will be solved using a 2D adaptive grid method in Section 4.4.2.3.3, with particular attention paid to the solenoidal condition.

One way to ensure a divergence-free magnetic field at all times is to make use of a vector potential formulation where $\mathbf{B} := \nabla \times \mathbf{A}$. In two-dimensional applications, the system (4.25) and (4.26) is then equivalent to the single PDE for the scalar A_3

component

$$\frac{\partial A_3}{\partial t} = -\mathbf{v} \cdot \nabla A_3 + \epsilon_m \Delta A_3 \ , \tag{4.27}$$

with $\frac{\partial A_3}{\partial y} = B_1$, $-\frac{\partial A_3}{\partial x} = B_2$, while $\mathbf{A} = (0, 0, A_3)$. We will use this simpler model in Section 4.4.2.3.1 to compare different means for generating a 2D adaptive grid. Note that magnetic field lines are isolines of this A_3 potential.

Finally, we point out (cfr. [17]) that the partial problem posed by the system (4.25) and (4.26), or equivalently the PDE (4.27), can be relevant as a physical solution to the special case where we consider incompressible flow $\nabla \cdot \mathbf{v} = 0$, the momentum equation (4.2) under the condition that the magnetic energy $\mathbf{B}^2/2$ is much smaller than the kinetic energy $\rho \mathbf{v}^2/2$, and the induction equation itself. In those circumstances, the momentum balance decouples from the magnetic field evolution. In the model problems studied, we therefore impose an incompressible flow field $\mathbf{v}(x, y)$.

4.4.2 Adaptive Grids in Two Space Dimensions

4.4.2.1 Transformation in 2D

As in the 1D case we first make use of a transformation of variables

$$\xi = \xi(x, y, t), \quad \eta = \eta(x, y, t), \quad \theta = t \ , \tag{4.28}$$

that yields for the Equation (4.27) (a similar derivation can be made for the (B_1, B_2) system)

$$\begin{aligned} \mathcal{J} A_{3,\theta} + A_{3,\xi}(x_\eta y_\theta - x_\theta y_\eta) + A_{3,\eta}(x_\theta y_\xi - x_\xi y_\theta) \\ = A_{3,\xi}\left(-v_1 y_\eta + v_2 x_\eta\right) + A_{3,\eta}\left(v_1 y_\xi - v_2 x_\xi\right) \\ +\epsilon_m \left[\left(\frac{x_\eta^2 + y_\eta^2}{\mathcal{J}} A_{3,\xi} \right)_\xi - \left(\frac{x_\xi x_\eta + y_\xi y_\eta}{\mathcal{J}} A_{3,\eta} \right)_\xi \right. \\ \left. - \left(\frac{x_\xi x_\eta + y_\xi y_\eta}{\mathcal{J}} A_{3,\xi} \right)_\eta + \left(\frac{x_\xi^2 + y_\xi^2}{\mathcal{J}} A_{3,\eta} \right)_\eta \right] . \end{aligned} \tag{4.29}$$

Here, $\mathcal{J} = x_\xi y_\eta - x_\eta y_\xi$ is the determinant of the Jacobian of the transformation. In general, we then allow for truly two-dimensionally deforming grids.

If we restrict the grid adaptation in a 1.5D manner, i.e., when we impose the extra restriction $x_\eta = y_\xi = 0$, we get $\mathcal{J} = x_\xi y_\eta$, and Equation (4.29) simplifies to

$$\begin{aligned} x_\xi y_\eta A_{3,\theta} - A_{3,\xi} y_\eta x_\theta - A_{3,\eta} x_\xi y_\theta = -v_1 y_\eta A_{3,\xi} - v_2 x_\xi A_{3,\eta} \\ +\epsilon_m \left[\left(\frac{y_\eta A_{3,\xi}}{x_\xi} \right)_\xi + \left(\frac{x_\xi A_{3,\eta}}{y_\eta} \right)_\eta \right] . \end{aligned} \tag{4.30}$$

This dimensionally split approach for the grid adaptation will be compared with fully 2D deformations for the model problem from Section 4.4.2.3.1.

4.4.2.2 Adaptive Grid PDEs in 2D

Due to the 2D transformation, now two fourth-order PDEs are needed to define the grid and thereby the transformation. As an immediate extension of the 1D case (4.11) we set

$$\left[\left(\mathcal{S}_1\left(x_\xi\right) + \tau x_{\xi\theta}\right)\mathcal{W}_1\right]_\xi = 0\,, \tag{4.31}$$

$$\left[\left(\mathcal{S}_2\left(y_\eta\right) + \tau y_{\eta\theta}\right)\mathcal{W}_2\right]_\eta = 0\,,$$

with $\mathcal{S}_1$ and $\mathcal{S}_2$ direction-specific versions of the operator $\mathcal{S}$ defined in (4.12). The weight functions in (4.31) are now

$$\mathcal{W}_1 = \sqrt{1 + \alpha\ A_{3,x}^2}, \quad \mathcal{W}_2 = \sqrt{1 + \alpha\ A_{3,y}^2}\,, \tag{4.32}$$

for a fully 2D adaptive grid, while in the 1.5D case, we set

$$\mathcal{W}_1 = \sqrt{1 + \alpha\ \max_y A_{3,x}^2}, \quad \mathcal{W}_2 = \sqrt{1 + \alpha\ \max_x A_{3,y}^2}\,. \tag{4.33}$$

It can be shown (using 1D arguments in two directions), that with the latter choice

$$\mathcal{J} = x_\xi\, y_\eta > 0, \qquad \forall\, \theta \geq 0\,, \tag{4.34}$$

so that this restricted grid adaptivity maintains the desirable property that grid cells do not fold over. For the more general case (4.32), no guarantee can be given that grid points will not collide! This could be called the "battle between adaptivity and regularity." The method parameters τ, σ, α are chosen in a similar way as before and are specified per problem in the following sections. After semi-discretization of (4.30) and (4.31) we end up with a banded ODE system with bandwidth $= 6{*}npts{+}2$, where $npts \times npts$ denotes the total number of gridpoints in 2D. This ODE system is again time-integrated with DASSL [10].

4.4.2.3 Numerical Results

4.4.2.3.1 Kinematic Flux Expulsion

This model problem dates back to 1966 [17], as one of the first studies to address the role of the magnetic field in a convecting plasma. Starting from a uniform magnetic field, its distortion by cellular convection patterns was simulated numerically for various values of the resistivity ϵ_m. We use this model problem to compare the 2D with the 1.5D approach for two-dimensional moving grids.

Our 2D kinematic flux expulsion uses an imposed four-cell convection pattern with its incompressible velocity field given by

$$\mathbf{v}(x, y) = (\sin(2\pi x)\cos(2\pi y), -\cos(2\pi x)\sin(2\pi y))\,.$$

We solve for the scalar vector potential A_3 from (4.27) on the domain $(x, y) \in [0, 1] \times [0, 1]$ and for times $t \in [0, 0.5]$. We set the adaptivity parameter α to unity,

and $\tau = 10^{-3}, \sigma = 1$. The grid dimension is 25×25, while the resistivity is set equal to $\epsilon_m = 0.005$. In terms of A_3, the initial uniform vertical field is obtained through $A_3|_{t=0} = 1 - x$, while the boundary conditions are

$$A_3|_{x=0} = 1, \; A_3|_{x=1} = 0, \; A_3|_{y=0} = A_3|_{y=1} \; .$$

In Figure 4.6, we compare the grids and the obtained solution $A_3(x, y)$ at time $t = 0.5$ for three simulation results. The top row uses a full 2D grid deformation, the middle row takes the 1.5D adaptivity approach, while the bottom row shows a reference VAC solution on a 100×100 uniform, static grid. The solution is shown both as a surface and a contour plot, with the contour values varying between 0 and 1 with steps of 0.05. Note that the 1.5D deformation works well for this case, since the steep parts of the solution mostly vary in the x-direction. Although the 2D grid shows slightly sharper contour lines in the middle of the domain, the 2D deformation may break down at some point in time. For both cases we gain a factor of 16 in the total number of grid points compared with the reference solution.

4.4.2.3.2 Advection of a Current-Carrying Cylinder

To demonstrate the dimensionally split grid adaptation on a case where truly 2D deformations are required, we solve for the circular advection of a current-carrying cylinder (taken from [13]).

With a computational domain of size $(x, y) \in [-50, 50] \times [-50, 50]$, we embed an isolated magnetic "flux tube" in a circulatory flow. The cylinder is specified by

$$A_3|_{t=0} = \begin{cases} R/2 - [(x - x_0)^2 + (y - y_0)^2]/2R & \text{if } (x - x_0)^2 + (y - y_0)^2 < R^2, \\ 0 & \text{elsewhere} \; , \end{cases}$$

and is initially centered at $(x_0, y_0) = (0, 25)$ with radius $R = 15$ and the magnetic field strength increases radially from zero to one at the cylinder edge. In terms of a current $\mathbf{J} = \nabla \times \mathbf{B}$, the cylinder has a constant axial current throughout.

We simply rotate this current-carrying cylinder around (counterclockwise) by imposing

$$\mathbf{v}(x, y) = (-y, x) \; .$$

When we solve for times $t \in [0, 2\pi]$, we then follow one period of revolution of the cylinder, at which time the initial configuration must be regained. The method parameters are: adaptivity parameter $\alpha = 200$ (due to the scaling-effect), $\tau = 10^{-3}$, $\sigma = 1$. We now use the dimensionally decoupled adaptivity on a 25×25 grid and a dimensionless artificial diffusion of 0.5×10^{-4}. Boundary conditions do not play a role in this example, so we simply took homogeneous Dirichlet conditions $A_3|_{\partial\Omega} = 0$ everywhere. In Figure 4.7 we see the grids, solutions, and contour plots at $t = 2\pi$. The adaptive grid is nicely situated around the cylinder, although the solution is slightly smoothed by the artificial diffusion term, which can also be seen in the contour plot.

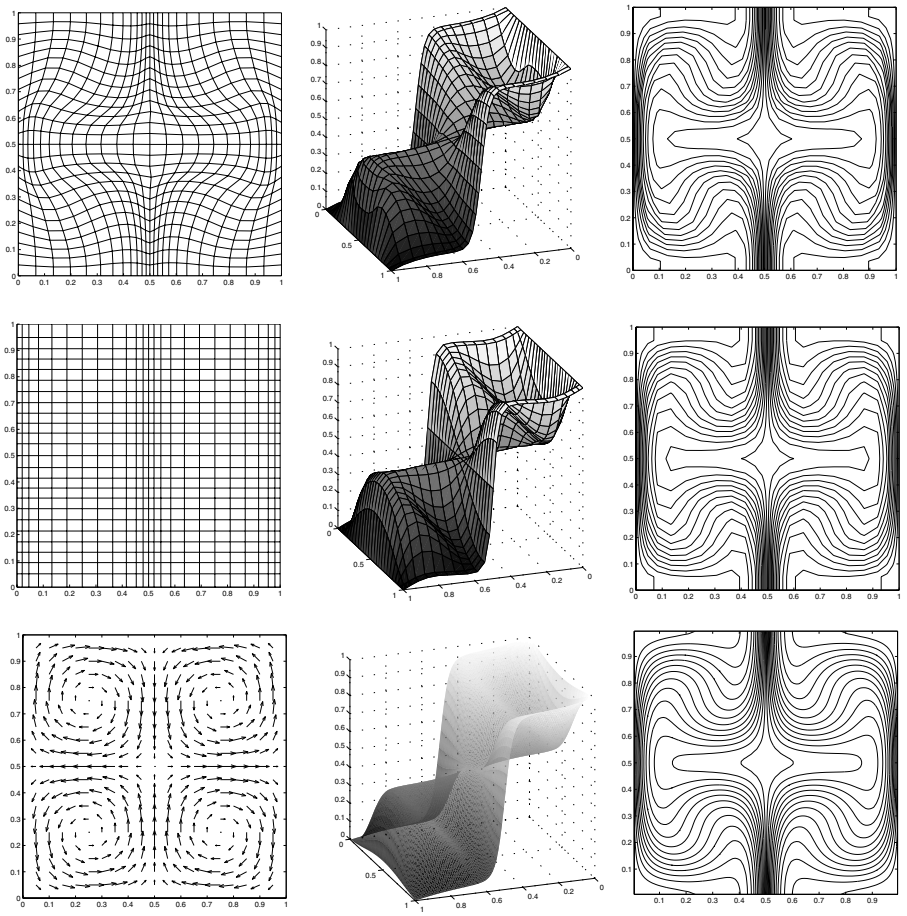


FIGURE 4.6

Three solution strategies for 2D kinematic flux expulsion compared: each row shows for time $t = 0.5$ the grid, the solution for the vector potential $A_3(x, y)$ as a surface plot, and as a contour plot (showing magnetic field lines) with fixed contour levels at $A_3 = 0 : 0.05 : 1$. The top row uses a 25×25 two-dimensionally deforming grid, the middle row uses restricted 1.5D adaptivity, and the bottom row is a 100×100 reference solution. The imposed velocity field is depicted at bottom left.

4.4.2.3.3 Conservation of $\nabla \cdot \mathbf{B} = 0$?

To investigate how the adaptive method copes with the important property $\nabla \cdot \mathbf{B} = 0$, we now take the full (B_1, B_2) system given by (4.25) and (4.26). Note that the current-carrying-cylinder model is not appropriate for this purpose, since the initial condition for the (B_1, B_2) system consists of piecewise linear parts (this follows from the initial condition for A_3 and $\mathbf{B} := \nabla \times \mathbf{A}$). As a consequence, *constant* weight functions W_1

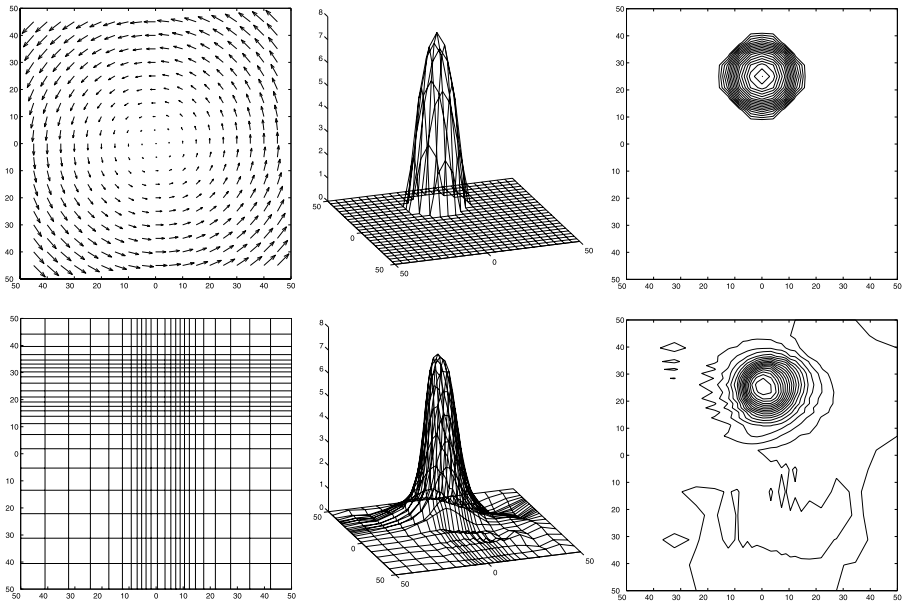


FIGURE 4.7
For initial time $t = 0$ (top row) and after one rotation at time $t = 2\pi$ (bottom row): Grid, surface plot of the potential A_3, and magnetic field lines for the current-carrying cylinder model with fixed contour levels at $A_3 = 0 : 0.325 : 7.5$. The top left frame shows the imposed circulation as a vector field.

and W_2 are obtained and therefore a uniform grid for all $t \geq 0$, independent of the choice of the adaptivity parameter α. For this reason, we examine the (B_1, B_2) version of the model in Section 4.4.2.3.1 with initial conditions $B_1|_{t=0} = 0$, $B_2|_{t=0} = 1$ and periodic boundary conditions. For simplicity we take the 1.5D approach. In Figure 4.8, we show a plot of $\nabla \cdot \mathbf{B} = 0$ on a 30×30 adaptive grid at $t = 0.1$, as evaluated from a central difference discretization:

$$[\nabla \cdot \mathbf{B} = 0]_{i,j} \approx \frac{B_{1,i+1,j} - B_{1,i-1,j}}{x_{i+1} - x_{i-1}} + \frac{B_{2,i,j+1} - B_{2,i,j-1}}{y_{j+1} - y_{j-1}} .$$

Numerical values of $||\nabla \cdot \mathbf{B}||_\infty$ for different grid sizes are: 0.2117 (on a 20×20 grid), 0.2104 (25×25 grid), and 0.1743 (30×30 grid). The main conclusion from these results is that, although the grid concentrates near areas of high-spatial activity, the solenoidal condition on the magnetic field is not preserved satisfactorily at all. This is a severe drawback of the current MOL implementation. A possible remedy for this could be adding a projection scheme after every time step, i.e., applying a Poisson solver to correct the divergence of the magnetic field [1].

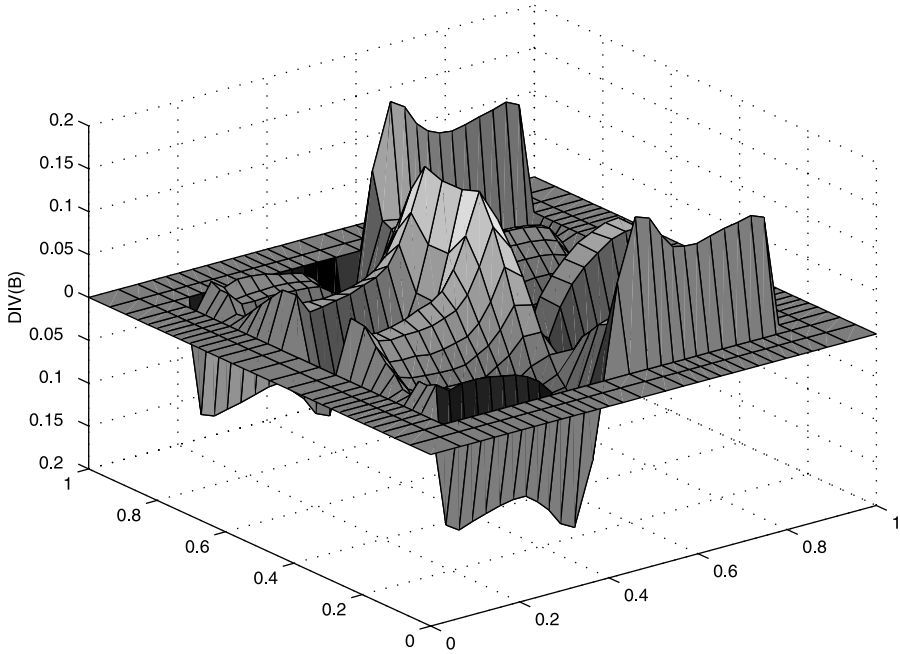


FIGURE 4.8
The divergence of the magnetic field for the solution in (4.4.2.3.3) at time $t = 0.1$ on a 30×30 adaptive grid. We evaluated the divergence from a centered difference formula.

4.5 Conclusions

In this chapter we applied the adaptive MOL technique to various 1D MHD problems and 2D magnetic field evolution simulations. In 1D, accurate numerical results were obtained for three important test cases. The method could further benefit from specific MHD properties that have not been exploited in the present implementation. For the 2D case, the adaptive method with restricted grid motion performed comparably to fully 2D adaptive simulations. This is of interest for easier generalizations to 3D calculations. Future work will consist of fully 2D MHD simulations and 3D applications (model problems could be taken from [6, 9]). From our results, it is clear that attention should be paid to means of maintaining pressure positivity in very low pressure situations, more physically based artificial diffusion terms, and an appropriate remedy for ensuring the solenoidal condition on the magnetic field vector in combination with the adaptive grid method for multi-dimensional applications. To allow for the latter applications, we will switch to the use of iterative methods for the linear systems behind the Newton process in the stiff ODE solver.

Acknowledgments

RK performed his work as part of the research program of the association agreement of Euratom and the "Stichting voor Fundamenteel Onderzoek der Materie" (FOM) with financial support from the "Nederlandse Organisatie voor Wetenschappelijk Onderzoek" (NWO) and Euratom. This work was partly performed in the project on "Parallel Computational Magneto-Fluid Dynamics," funded by the Dutch Science Foundation (NWO) Priority Program on Massively Parallel Computing.

References

[1] J.U. Brackbill and D.C. Barnes, The effect of nonzero $\nabla \cdot \mathbf{B}$ on the numerical solution of the magnetohydrodynamic equations, *J. Comput. Phys.,* **35**, (1980), 426–430.

[2] M. Brio and C.C. Wu, An upwind difference scheme for the equations of ideal magnetohydrodynamics, *J. Comput. Phys.,* **75**, (1988), 400–422.

[3] E.A. Dorfi and L.O.'Drury, Simple adaptive grids for 1-D initial value problems, *J. Comput. Phys.,* **69**, (1987), 175–195.

[4] H. Friedel, R. Grauer, and C. Marliani, Adaptive mesh refinement for singular current sheets in incompressible magnetohydrodynamic Flows, *J. Comput. Phys.,* **134**, (1997), 190–198.

[5] W. Huang and R.D. Russell, Analysis of moving mesh partial differential equations with spatial smoothing, *Research Report,* **93-17**, (1993), Mathematics and Statistics, Simon Fraser University, Burnaby, British Columbia.

[6] R. Keppens, G. Tóth, R.H.J. Westermann, and J.P. Goedbloed, Growth and saturation of the Kelvin-Helmholtz instability with parallel and antiparallel magnetic fields, *J. Plasma Phys.,* **61**, (1999), 1–19.

[7] R. Keppens, M. Nool, P.A. Zegeling, and J.P. Goedbloed, Dynamic grid adaptation for computational magnetohydrodynamics, *Lecture Notes in Computer Science,* **1823**, (2000), Springer Verlag, Berlin, 61–70.

[8] R.L. LeVeque, D. Mihalas, E.A. Dorfi, and E. Müller, Computational methods for astrophysical fluid flow, *Saas-Fee Advanced Course,* **27**, (1998), lecture notes 1997, Swiss Society for Astroph. and Astron., O. Steiner and A. Gautschy, eds., Springer Verlag, Berlin.

[9] M.G. Linton, G.H. Fisher, R.B. Dahlburg, and Y. Fan, Relationship of the multimode kink instability to δ-spot formation, *Astrophys. J.,* **522**, (1999), 1190–1205.

[10] L.R. Petzold, A description of DASSL: A differential/algebraic system solver, *IMACS Transactions on Scientific Computation,* (1983), R.S. Stepleman et al., eds., North-Holland, Amsterdam, 65–68.

[11] W.E. Schiesser, *The Numerical Method of Lines, Integration of Partial Differential Equations,* Academic Press, (1991), San Diego, CA.

[12] J.M. Stone and M.L. Norman, ZEUS-2D: a radiation magnetohydrodynamics code for astrophysical flows in two space dimensions. II. The magnetohydrodynamic algorithms and tests, *Astrophys. J. Suppl.,* **80**, (1992), 791–818.

[13] G. Tóth and D. Odstrčil, Comparison of some flux corrected transport and total variation diminishing numerical schemes for hydrodynamic and magnetohydrodynamic problems, *J. Comput. Phys.,* **128**, (1996), 82–100.

[14] G. Tóth, A general code for modeling MHD flows on parallel computers: Versatile Advection Code, *Astrophys. Lett. & Comm.,* **34**, (1996), 245–250.

[15] G. Tóth , R. Keppens, and M.A. Botchev, Implicit and semi-implicit schemes in the Versatile Advection Code: numerical tests, *Astron. & Astroph.,* **332**, 1159–1170.

[16] R.A. Trompert, Local uniform grid refinement for time-dependent partial differential equations, *CWI-tract,* **107**, (1995), Centrum voor Wiskunde en Informatica, Amsterdam.

[17] N.O. Weiss, The expulsion of magnetic flux by eddies, *Proc. of Roy. Soc. A,* **293**, (1996), 310–328.

[18] P.A. Zegeling, Moving grid methods for time-dependent partial differential equations, *CWI-tract,* **94**, (1993), Centrum voor Wiskunde en Informatica, Amsterdam.

[19] P.A. Zegeling, r-refinement for evolutionary PDEs with finite elements or finite differences, *Appl. Num. Maths.,* **26**, (1998), 97–104.

[20] U. Ziegler, A three-dimensional Cartesian adaptive mesh code for compressible magnetohydrodynamics, *Comp. Phys. Comm.,* **116**, (1999), 65–77.

Chapter 5

Development of a 1-D Error-Minimizing Moving Adaptive Grid Method

Mart Borsboom

Abstract Instead of developing an adaptive grid technique for some discretization method, we develop a discretization technique designed for grid adaptation. This so-called compatible scheme allows to translate the leading term of the local residual directly in terms of a local error in the numerical solution (the numerical modeling error). An error-dependent smoothing technique is used to ensure that higher-order error terms are negligible. The numerical modeling error is minimized by means of grid adaptation. Fully converged adapted grids with strong local refinements are obtained for a steady-state shallow-water application with a hydraulic jump. An unsteady application confirms the importance of taking the error in time into account when adapting the grid in space. We discuss the shortcomings of the present implementation and the remedies currently under development.

5.1 Introduction

In moving adaptive grid methods, both the numerical solution and the grid on which that numerical solution is defined are considered unknown. This offers an enormous increase in numerical modeling flexibility, but also raises the far from easy question of how to couple the grid to the numerical solution. The standard answer is to apply an equidistribution principle: the grid is defined by the equidistribution of a solution-dependent error measure or monitor function over the grid cells. Many different error measures have been proposed in the literature, often chosen heuristically, with little or no justification [12]. Ideally, error measures include all important sources of numerical solution errors, i.e., terms that indicate how grid resolution, grid stretching, grid curvature, and grid skewness affect the accuracy of the numerical simulation.

If essential error information is missing or not used properly, grid adaptation may not give any improvement [27]. The use of an incomplete or incorrect error estimate

could even lead to an increase of solution errors due to the incorrect grid point distributions and/or the severe grid distortions that it may induce. This can be avoided by adding terms that ensure some form of grid regularity or that limit grid variation, thereby reducing at the same time the adaptability of the grid. The resulting adaptive grid technique will probably require problem-dependent fine tuning as well. These arguments clearly illustrate the importance of a reliable error measure when designing a moving adaptive grid algorithm [12, 17].

Residual-based, *a posteriori* error estimates for hyperbolic problems of practical interest (in particular nonlinear flow problems) still lack sharpness or are not generally applicable [11, 15, 17]. Using the residual as such is generally not a good idea. The model equations (and hence the residual) can be multiplied by any smooth positive function without changing the solution. Depending on this scaling, a large/small residual may therefore not correspond with a large/small solution error. The relation between the solution error and the residual can be estimated by considering a global dual problem, obtained after linearization, which is usually complex and expensive to solve. Solving a dual problem locally is practically more feasible [23], although it does indicate only the locally generated error and ignores solution errors that are the result of the accumulation during propagation of errors generated elsewhere [14]. However, a small local residual may be due to the cancellation of numerical errors originating from different modeling terms and may therefore not correspond with small errors as such. It is usually also not clear how the local residual depends on the local grid parameters (size, stretching, curvature, skewness), which makes this approach less suited for error-minimizing grid adaptation. Furthermore, it may be difficult to develop a meaningful local dual problem with appropriate local (inflow, outflow) boundary conditions.

Propagation, and hence error propagation, is typical of flow and transport problems; it makes the development of an error-minimizing adaptive grid technique for such applications extremely complicated because of the obscure and complex relation that exists between the regions where the solution errors are generated and the regions where the solution errors manifest themselves. This applies to numerical errors as well as to physical errors and errors due to incorrect data. Examples of physical modeling errors are the approximate description of turbulence or the neglect of certain aspects like a space dimension or viscous effects. Data errors may consist of uncertainties in, e.g., the initial and boundary conditions, geometry, and certain model parameters. When reducing numerical errors through grid adaptation, the presence of these other modeling errors should be taken into consideration as well [7]. In particular, refining the grid is not useful in regions where the solution error is mainly due to the applied physical model or caused by unreliable or incomplete input data. This strongly limits the usefulness of an adaptive grid technique that merely attempts to minimize the numerical solution error. In complex applications it is, however, virtually impossible to quantify the effect of physical modeling errors and data errors on the solution, let alone to take the relative importance of this effect into account in the grid adaptation.

We have, therefore, not opted for the development of a grid adaptation method that aims to reduce some upper error bound to a given tolerance level. Instead, our goal

is to minimize the part of the numerical solution error that is generated locally as a result of using a numerical approximation of the model equations. We will refer to this error as the *numerical modeling error*. Note that it may be feasible to compare the numerical modeling error with physical modeling errors and data errors. This could then be used to assess its relative importance and hence the usefulness of local (adaptive) grid refinements.

The basis of our method is the approximation of the local residual in terms of a local solution error. We want to avoid the use of the residual as such in combination with a local dual problem because of the disadvantages mentioned before. Investigating this issue more closely, we found that it is generally impossible to reformulate the residual directly in terms of local errors in the numerical solution. The reason for this seems to be the discrepancy that often exists between the discretization of the model equations and the way the numerical solution is represented. For example, the use of piecewise linear polynomials to reconstruct the numerical solution over the whole domain is consistent with the use of a second-order accurate discretization technique. This does not mean however that the second-order interpolation error is representative of the numerical modeling errors.

So instead of analyzing the residual of some numerical scheme, we decided to investigate numerical schemes for their suitability of rewriting the residual as a local solution error. This has led to the development of a discretization technique designed for error analysis and hence for grid adaptation. Using this technique, the residual of any flow or transport equation discretized on a non-uniform, moving grid can indeed be reformulated, at least in 1-D, in terms of a local solution error.

The moving adaptive grid equations are obtained by solving an optimization problem, minimizing the numerical modeling error in the L_1 norm. The L_1 norm has been selected because of its physical relevancy (see also [29]). In particular, the numerical approximation of a discontinuity spread over a fixed number of grid points shows only the expected first-order error behavior if it is measured in the L_1 norm. This is consistent with the local order of accuracy of the numerical scheme.

5.2 Two-Step Numerical Modeling

In the numerical error analysis that we will present we will make extensive use of truncated Taylor-series expansions. A smoothing technique is applied prior to the discretization to ensure that higher-order error terms are negligible. This makes the numerical modeling process essentially a two-step procedure. In the first step, the problem to be solved is regularized by adding a suitable form of smoothing; in the second step, the regularized problem with smooth solution is discretized. The amount of smoothing in the first step is controlled by the error that is made in the second step by an error feedback loop. Smoothing or regularization is used frequently in adaptive grid techniques [5, 8, 12, 17].

Figure 5.1 shows the elements of the two-step method. We discern a smoothing step and a discretization step connecting four different problems:

- the **difficult problem**, i.e., the original physical model problem, whose solution may include steep gradients and discontinuities
- the **easy problem**, i.e., the regularized problem, whose solution is smooth enough to be discretized with sufficient accuracy
- the **discretized problem**, obtained upon the discretization of the easy problem
- the **equivalent problem**, i.e., the differential problem equivalent with the discretized problem

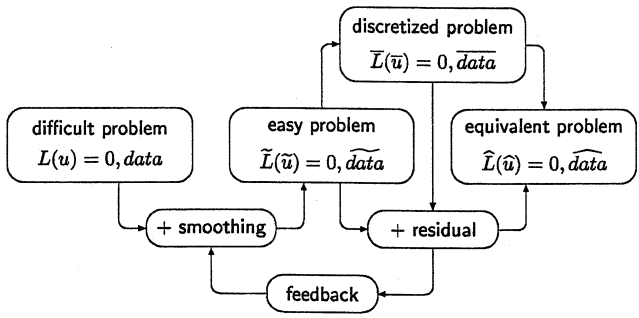


FIGURE 5.1
Outline of the two-step numerical modeling technique.

The smoothing step converts the difficult problem into the easy problem by explicitly adding suitable smoothing terms. The discretization step converts the easy problem into the equivalent problem by means of a suitable discretization of the easy problem. To obtain the second conversion in an explicit form, we determine the residual associated with the discretized problem. The residual can be viewed as the continuous equivalent of the discretization error.

The difficult, easy, and equivalent problem are all differential problems. They each consist of a system of partial differential equations ($L(u) = 0$, $\widetilde{L}(\widetilde{u}) = 0$, $\widehat{L}(\widehat{u}) = 0$) and a set of data ($data$, $\widetilde{data}$, $\widehat{data}$) containing, e.g., the boundary and initial conditions. Functions and parameters required to fully specify each problem (geometry and source terms, for example) are also included in the data. The discretized problem, on the other hand, consists of a system of algebraic equations ($\overline{L}(\overline{u}) = 0$) supplemented with a discrete data set ($\overline{data}$).

The discretized problem is usually the only problem that is solvable. However, we do not know the differential problem corresponding with its solution $\overline{u}$. In contrast, the difficult and easy problem are fully specified but their solution (respectively, u and $\widetilde{u}$) is largely unknown. The link between both is the equivalent problem. The equivalent solution $\widehat{u}$ and data $\widehat{data}$ are a close and smooth approximation of the

solution and data of the discretized problem, while the equivalent system of equations $\widehat{L}(\widehat{u}) = 0$ is an approximation of the system of partial differential equations satisfied by $\widehat{u}$ given $\widehat{data}$. If the discretization is consistent, the equivalent problem is also an approximation of the easy problem that in turn is an approximation of the difficult problem.

Some form of two-step modeling is used in virtually any discretization method for flow and transport problems of practical interest. Smoothing may have been added explicitly (i.e., prior to the discretization) in the form of artificial dissipation terms in the equations, or implicitly by incorporating a dissipative mechanism like flux limiters inside the discretization method. Usually, the purpose of adding dissipation is to reduce wiggles or to ensure monotonicity. Here we demand that it guarantees a sufficiently smooth solution, i.e., negligibly small higher-order discretization errors, to allow a meaningful error analysis that can be used in a grid adaptation procedure.

A simple example may illustrate the connection between smoothing and grid adaptation. We consider a smooth function with a discontinuity at $x = 0$ and approximate this function by means of a continuous piecewise linear discrete function defined on the grid $x = \pm 1, \pm 3, \ldots$. Because the discontinuity is at the center of a grid cell, its best possible numerical approximation is only one grid cell wide. We now discretize the function on the grid $x = 0, \pm 1, \pm 2, \ldots$ and observe that there is no improvement in accuracy, despite the fact that the resolution has been doubled. The reason for this is obvious: the discontinuity is now at a grid point and must be spread over two grid cells. It is even possible to get worse results on a finer grid, simply because the position of the discontinuity changes from a cell center to a grid point (consider the grid $x = 0, \pm 1.5, \pm 3, \ldots$). Such a function has been considered in [2]. The results presented in that paper show that although the global error behavior in the L_1 norm is indeed first order, without smoothing of the discontinuity the approximation error is an irregular function of the number of grid points. When the function is first properly smoothed, the L_1 approximation error decreases uniformly as the number of grid points increases, showing a clear first-order behavior independent of the position of the grid points.

The dependence of the error on grid point position is unacceptable in the context of the use of moving adaptive grid techniques where we need a monotone relation between the numerical accuracy and the grid resolution. The example shows that the behavior of higher-order error components can be rather erratic and becomes dominant if the solution is not smooth. This can be avoided by smoothing the function that is to be approximated (e.g., the solution of a differential problem) over a sufficient number of grid cells. It can be easily verified that the smoother the function, the less sensitive its numerical approximation is to the position of the grid points. However, smoothing introduces another form of numerical error, so in practice a compromise needs to be found between the amount of smoothing (should be as small as possible) and the amount of spreading (must be large enough). In particular, no additional smoothing is required if the function is already sufficiently smooth. These requirements are precisely the design criteria of the two-step method.

If designed correctly, explicitly added artificial dissipation weakens (infinitely) steep gradients by spreading them over enough but not too many grid cells. The numerical scheme should then be able to handle those gradients correctly, meaning that the effect of discretization errors should be sufficiently small. The mechanism is comparable with upwind techniques and flux limiters, but an essential difference is that flux limiters are mainly designed to spread steep gradients over the smallest number of grid cells possible without causing any over- or undershoots (see, e.g., [13]). They are not designed to keep numerical errors below an acceptable level. Using upwind methods, solutions can therefore be obtained that look reasonable and yet are completely wrong due to a lack of spatial resolution (see, e.g., [1, 19]). This is consistent with the conclusion from the above example and indicates that the preservation of monotonicity is no guarantee for numerical accuracy. On the other hand, explicitly added artificial dissipation generally does not give that guarantee either.

The design of a suitable form of artificial dissipation has been the subject of many publications (see, e.g., [13] for an overview). We prefer the approach proposed half a century ago by Von Neumann and Richtmyer who suggested to "utilize the well-known effect on shocks of dissipative mechanisms, such as viscosity" [26]. The advantage of the artificial viscosity concept is that no physically unrealistic terms are introduced. In particular, no dissipation terms that would destroy mass conservation are added to the continuity equation.

There remains the problem of how to scale the artificial viscosity. Since its purpose is to regularize the difficult problem, it must depend in some way on the discretization error. This feedback should ensure that the easy problem will always be easy enough for the discretization error to be sufficiently small: the discretization error should be such that the neglected higher-order error terms are indeed negligible.

Once these elements have been designed properly, the development of an error-minimizing moving adaptive grid technique is relatively straightforward. The discretization error is a function of the grid, and can be minimized by moving the grid points. When the discretization error is minimized, the artificial viscosity is minimized as well because of the feedback loop. It will be shown that this minimizes all coefficients in the equivalent differential equation that are different from the original differential equation of the difficult problem. In other words, the perturbations introduced in the difficult problem by the numerical solution technique are minimized and thereby their local effect on the solution (the numerical modeling error).

Notice that increasing the viscosity artificially allows direct comparison with the true physical viscosity, to assess the relative importance of the artificial dissipation. This may prove to be very useful in view of physical modeling errors. When numerical results are compared with measurement data, it is then easily verified whether deviations should be attributed to an incorrect physical viscosity model (e.g., a turbulence model) or to a large artificial viscosity. It may even be possible to incorporate knowledge about the validity of the physical model in the grid adaptation procedure, to avoid that grid points are moved to regions where the model is not very accurate anyway.

5.3 1-D Shallow-Water Equations

The physical problem that we consider is the modeling of free-surface flow through an open channel section. In many practical applications, the vertical and transversal length scales are small compared to the longitudinal scales and can be neglected. Assuming constant density ρ and restricting ourselves to channels of constant width W, the flow can then be modeled by the 1-D equations (see, e.g., [3]):

$$\frac{\partial d}{\partial t} + \frac{\partial q}{\partial x} = 0 \,, \tag{5.1}$$

$$\frac{\partial q}{\partial t} + \frac{\partial q^2/d}{\partial x} + gd\frac{\partial h}{\partial x} + g\frac{Pq|q|}{Wd^2C^2} = \frac{\partial}{\partial x}\left(\nu_{\text{art}} d \frac{\partial q/d}{\partial x}\right) \,. \tag{5.2}$$

Space coordinate x [m] is defined along the axis of the channel. The unknowns of the flow equations are the water depth d [m] and the depth-integrated flow velocity in x-direction q [m^2/sec]. Depth-averaged flow velocity u [m/sec] is equal to q/d. Gravitational acceleration g [m/sec^2] is a given constant parameter. Chézy coefficient C [m$^{1/2}$/sec] is a friction parameter to account for friction losses due to the bottom friction. Water level h [m] is the sum of d and the given bottom level of the channel z_b [m]:

$$h = d + z_b \,, \tag{5.3}$$

while wetted perimeter P [m] is defined as $P = W + 2d$.

Continuity equation (5.1) is a mass conservation law. It describes the balance between the rate of change of mass in a cross-section and the net mass flow entering that cross-section for constant ρ and W. The terms in the left-hand side of momentum equation (5.2) represent the rate of change of momentum, the net momentum flow, the hydrostatic pressure force, and the bottom friction force, respectively. In the right-hand side of (5.2) we have the added artificial smoothing term, so (5.1) and (5.2) form in fact the equations of an easy problem (cf. Figure 5.1). Since we will consider the easy problem only, we have omitted the tilde for convenience. The form of the smoothing term is equal to the physical viscosity term integrated over the water depth, neglecting non-conservative parts to avoid artificial loss of momentum. Its viscosity coefficient ν_{art} [m^2/sec] is of course artificial, although the same formulation could also be used to improve the modeling of physical effects [18].

Equations (5.1) and (5.2) can be recombined to the characteristic equations:

$$(c \mp u) \;\text{ cont. eq. (5.1)} \;\; \pm \text{ mom. eq. (5.2)} \,, \tag{5.4}$$

with $c = \sqrt{gd}$ the wave celerity. These equations describe the propagation of the Riemann variables $(c \mp u)\partial d \pm \partial q$ along the characteristics $dx/dt = u \pm c$. The flow behavior depends to a large extent on the direction of the characteristics. Introducing

the Froude number $\mathrm{Fr} = |u|/c$ we make a distinction between subcritical flow ($\mathrm{Fr} < 1$; propagation in both directions), critical flow ($\mathrm{Fr} = 1$; one propagation direction vanishes), and supercritical flow ($\mathrm{Fr} > 1$; only propagation in downstream direction). Note that in the shallow-water *model* there will always be some information moving upstream (even if $\mathrm{Fr} > 1$), due to the diffusive transport introduced by the artificial viscosity. This is perfectly acceptable if the effect is small enough.

In regions where the solution of (5.1) and (5.2) is differentiable, the equations can be recombined to the energy equation:

$$\begin{aligned} &\frac{\partial}{\partial t}\left(d\left(\frac{1}{2}u^2 + gh - \frac{1}{2}gd\right)\right) + \frac{\partial}{\partial x}\left(ud\left(\frac{1}{2}u^2 + gh\right)\right) \\ &\quad = -g\frac{Pu^2|u|}{WC^2} - \nu_{\mathrm{art}}d\left(\frac{\partial u}{\partial x}\right)^2 + \frac{\partial}{\partial x}\left(u\nu_{\mathrm{art}}d\frac{\partial u}{\partial x}\right) . \end{aligned} \tag{5.5}$$

Ignoring the dissipation terms in the right-hand side, the steady-state solution of Equation (5.5) reads (from (5.1) we obtain $ud = q = \text{constant}$):

$$h + \frac{u^2}{2g} = \text{constant} . \tag{5.6}$$

This Bernoulli equation shows that the energy head, $h + \frac{1}{2}u^2/g$, is constant in smooth stationary frictionless flow; no energy is dissipated. The equation is not valid across a steady discontinuous hydraulic jump where we have to apply the Rankine-Hugoniot relations obtained from (5.1) and (5.2):

$$\begin{aligned} (ud)_1 &= (ud)_2 , \\ \left(u^2d + g\frac{d^2}{2}\right)_1 &= \left(u^2d + g\frac{d^2}{2}\right)_2 . \end{aligned} \tag{5.7}$$

The indices 1 and 2 indicate the state left and right of the discontinuity. Solving (5.7) across a hydraulic jump gives the energy loss across the jump. Further details can be found in [4]. Using the steady-state solution of (5.1), discharge $q = ud = \text{constant}$, the solution of (5.6) (in regions where the solution is smooth) and (5.7) (across a hydraulic jump) can be determined analytically, given suitable boundary conditions. This provides a useful analytical bench-mark solution to compare numerical solutions with [10, 21].

We point out that the discontinuous hydraulic jump is only a solution of (5.1) and (5.2) if the artificial viscosity is set to zero, i.e., if the model equations are considered to form a difficult problem (cf. Figure 5.1). With the addition of the artificial smoothing term, it has become an easy problem; discontinuities are spread and "replaced" so to speak by gradients of limited steepness. It is easily verified that the spreading is proportional to the value of ν_{art}.

At some distance of a smooth but nevertheless steep gradient, the solution gradients will be small again, and the effect of the artificial viscosity negligible. Neglecting the variations in channel geometry, it follows from momentum equation (5.2) that

the jump relations (5.7) (or the unsteady equivalent) are still satisfied across the jump region. So if only a moderate amount of smoothing is applied, the behavior of discontinuities is still modeled accurately. The condition to be satisfied here is that within a smoothing region the channel variation must be small. This condition was formulated 50 years ago by Von Neumann and Richtmyer: " . . . for the assumed form of dissipation, and for many others as well, the Rankine–Hugoniot equations are satisfied provided the thickness of the shock layers is small in comparison with other physically relevant dimensions of the system" [26].

An interesting aspect of the easy problem is that because the solution is smooth and differentiable, energy equation (5.5) is valid everywhere, also across the jump regions. In fact, the sudden energy drop across the jump is replaced by a steep decrease of the energy head due to the artificial viscosity. Since jump conditions (5.7) are still valid, the net result must be the same.

White presents an analysis of the structure of a 1-D viscous aerodynamic shock wave [28]. The analysis can be applied to hydrodynamic shocks as well. His results confirm that the thickness of the shock is proportional to the size of the viscosity coefficient. A rather surprising result is the entropy overshoot in the shock which is due to the energy redistribution caused by the viscous term. In the inviscid limit, the overshoot becomes a peak of zero thickness and vanishes, leaving only the well-known discontinuous increase of entropy. In our model the entropy overshoot corresponds with an undershoot of the energy head. The last term in the right-hand side of (5.5) is responsible for this effect. It is first negative (u decreases, hence $\partial u/\partial x$ becomes strongly negative inside the shock), but becomes positive at the end of a viscous shock layer. Overall the third term does not affect the energy across a shock because of its conservative form. The energy across a shock decreases because of the negative-definite second term in the right-hand side that also prevents the development of non-physical expansion shocks (the entropy condition, cf. [13]). At the end of a shock, however, where the third term is strongly positive and dominant, the right-hand side of (5.5) becomes positive causing a small energy uplift. As explained before, the overall energy loss across a viscous shock can still be a very close approximation of the inviscid shock loss.

The conclusion that can be drawn from this discussion is that although the solutions are different locally, globally the solution of the easy problem and the difficult problem may be expected to be virtually the same provided that not too much artificial viscosity is added. This is precisely the purpose of grid adaptation; the amount of artificial dissipation required depends on both the solution and the grid, so by optimizing the grid as a function of the solution, the artificial viscosity can be minimized and the accuracy maximized. This too was perceived by Von Neumann and Richtmyer: "the qualitative influence of these terms (read: the artificial viscosity) can be made as small as one wishes by choice of a sufficiently fine mesh" [26].

A condition to be fulfilled by the numerical scheme is that the solution of the easy flow problem is calculated with sufficient accuracy. To be able to capture all details of the (artificially thickened) viscous shock, including the undershoot of the energy head, shocks must be spread over at least several grid cells. It seems that this argument

can be reversed: if a shock is not modeled as a discontinuity (any shock-capturing method), it should be modeled as a viscous shock of a certain thickness in order to be physically realistic. Upwind schemes that spread shocks over only one or two grid cells may not be capable of modeling the entropy overshoot, and hence the dynamics, of that shock correctly. This observation is in line with the discussion of Section 5.2 and emphasizes the importance of sufficient resolution and hence smoothing.

5.4 Compatible Discretization

Our investigations have revealed that there seems to be only one way to obtain a discretization of flow and transport equations with the property that the residual can be formulated in terms of errors in the numerical solution and other variables. The key element of this approach is the definition of unique approximations per grid cell of *all* variables. We have used piecewise polynomial approximations that are defined on a uniform grid in the parameter space or computational space (ξ, τ). It is convenient for the discretization of the model equations and for the error analysis to define the differential problem in this computational space as well.

Since we consider moving grids, the mapping of the computational space (ξ, τ) onto the physical space (x, t) is defined by the two functions:

$$x = x(\xi, \tau) \ , \ t = t(\tau) \ . \tag{5.8}$$

Actually, it is defined by several such functions because each problem that we consider in the two-step modeling technique (cf. Figure 5.1) requires its own coordinate transformation. This is only relevant for the two problems related to the numerical scheme: the discretized problem and the equivalent problem. We will see later that it is indeed essential to consider for each of these two problems a separate coordinate transformation.

The discretization in time that we will apply is a two-level method. As a consequence, each time step can be considered separately, i.e., each time step the transformation in time can be redefined including a change of the size of the time step. This permits us to restrict ourselves to transformation functions (5.8) that are linear in τ:

$$t_\tau = 1 \ , \ x_{\tau\tau} = 0 \ , \tag{5.9}$$

where we have used the convention that a subscript that is a coordinate denotes differentiation with respect to that coordinate.

The (moving) space-time grid that we use is shown in Figure 5.2. The circles in that figure indicate the coordinates (x_i^n, t^n) of the grid points (i, n) determine the coordinate transformation of the discretized problem. Using (bi)linear interpolations

per grid cell, the transformation is defined by:

$$\begin{aligned}\overline{x}(\xi,\tau) =& \frac{\xi_i - \xi}{\Delta\xi}\left(\frac{\tau^n - \tau}{\Delta\tau}x_{i-1}^{n-1} + \frac{\tau - \tau^{n-1}}{\Delta\tau}x_{i-1}^{n}\right) \\ &+ \frac{\xi - \xi_{i-1}}{\Delta\xi}\left(\frac{\tau^n - \tau}{\Delta\tau}x_{i}^{n-1} + \frac{\tau - \tau^{n-1}}{\Delta\tau}\Delta\xi x_{i}^{n}\right) , \\ \overline{t}(\xi,\tau) =& \frac{\tau^n - \tau}{\Delta\tau}t^{n-1} + \frac{\tau - \tau^{n-1}}{\Delta\tau}t^{n} ,\end{aligned} \tag{5.10}$$

with:

$$\begin{aligned}\xi_{i-1} \le \xi \le \xi_i , \quad i = 1,\ldots,I+1 , \\ \tau^{n-1} \le \tau \le \tau^n , \quad n = 1,\ldots,N .\end{aligned}$$

We have $\xi_i = \xi_{i-1} + \Delta\xi$, $i = 1,\ldots,I+1$, with $\Delta\xi$ the size of the uniform grid in computational space. The grid consists of $I+1$ grid cells, with I grid points inside the domain and 2 virtual grid points outside the boundaries (cf. Figure 5.2). The reason for this will become clear in Section 5.4.1. In order to satisfy (5.9), we take $\tau^n - \tau^{n-1} = \Delta\tau = \Delta t$ within each time step $[\tau^{n-1}, \tau^n]$. As explained before, this does not preclude the use of different values of Δt in different time steps. The total number of time steps is N.

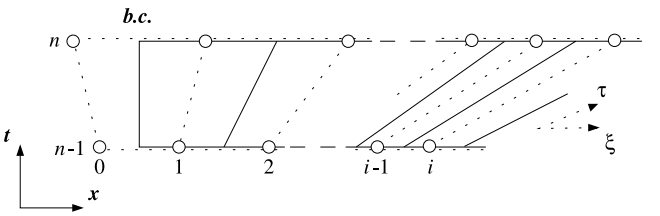


FIGURE 5.2
Grid in physical time and space.

Using (5.8) and (5.9), Equations (5.1) and (5.2) transformed to computational space can be written as:

$$\frac{\partial(x_\xi d)}{\partial\tau} + \frac{\partial(q - x_\tau d)}{\partial\xi} = 0 , \tag{5.11}$$

$$\frac{\partial(x_\xi q)}{\partial\tau} + \frac{\partial((q/d - x_\tau)q)}{\partial\xi} + gd\frac{\partial h}{\partial\xi} + x_\xi g\frac{Pq|q|}{Wd^2C^2} = \frac{\partial}{\partial\xi}\left(\frac{\nu_{\text{art}}d}{x_\xi}\frac{\partial q/d}{\partial\xi}\right) . \tag{5.12}$$

This result has been obtained upon multiplying the transformed equations by x_ξ and rearranging the terms to recover the conservative form of the original system

formulated in physical space. Notice that the coefficients t_τ have vanished because of (5.9).

For the discretization of (5.11) and (5.12) we introduce uniquely defined finite-dimensional function approximations of all variables: piecewise linear interpolations in computational space but backward piecewise constant (and hence discontinuous) extrapolations in computational time. The latter makes that the discretization in time will have the form of a backward Euler scheme. It provides the amount of dissipation in time required to ensure stability for any size of the time step, and also simplifies the analysis of the error in time. The accuracy obtained with this first-order scheme in time may still be quite acceptable, provided that we are able to design a moving adaptive grid method capable of aligning the grid properly with the solution. So we decided to use:

$$\overline{d}(\xi,\tau) = \frac{\xi_i - \xi}{\Delta\xi} d_{i-1}^n + \frac{\xi - \xi_{i-1}}{\Delta\xi} d_i^n \ , \tag{5.13}$$

$$\begin{aligned} \xi_{i-1} \le \xi \le \xi_i \ , &\quad i = 1, \ldots, I+1 \ , \\ \tau^{n-1} < \tau \le \tau^n \ , &\quad n = 1, \ldots, N \ . \end{aligned}$$

Similar expressions are used for unknown $\overline{q}$, wetted perimeter $\overline{P}$, and artificial viscosity coefficient $\overline{\nu}_{\text{art}}$. The expression is assumed to exist for discretized water level $\overline{h}$. As before, we have used the overbar to indicate discrete functions (cf. Figure 5.1).

All discrete functions, i.e., $\overline{x}$ [Equation (5.10)], $\overline{d}$ [Equation (5.13)], $\overline{q}$, $\overline{h}$, $\overline{P}$, and $\overline{\nu}_{\text{art}}$, must be (made) sufficiently smooth [the smoothness of $\overline{t}$ is guaranteed because of (5.9)]. This condition must be fulfilled in order that higher-order error terms can be neglected in the error analysis. The smoothing step with error feedback loop (cf. Figure 5.1) should take care of that. As for the unknowns $\overline{d}$ and $\overline{q}$, their smoothness is realized by the artificial viscosity term. To ensure smoothness of the artificial viscosity coefficient (an essential part of the method!), a separate equation is applied:

$$\nu_{\text{art}} - \alpha \Delta\xi^2 \frac{\partial^2 \nu_{\text{art}}}{\partial \xi^2} = c_\nu Err_\nu \ . \tag{5.14}$$

Function Err_ν is an error expression of dimension [m^2/sec] that will be specified later. For the moment it is sufficient to mention that Err_ν (and hence ν_{art}) is $O(\Delta x^3)$ in regions where the solution is smooth and $O(\Delta x)$ in regions where steep gradients develop. This makes that the artificial viscosity mechanism does not affect the formal second-order accuracy of the scheme, while discontinuities will be spread over a fixed number of grid cells depending on the value of constant scaling coefficient c_ν.

The amount of smoothing applied in (5.14) is determined by constant coefficient α. The smoothing of ν_{art} is required to reduce the unreliable higher-order error information that may be present in Err_ν to an insignificant level. These errors are partly due to the neglect of higher-order errors in the error analysis and partly introduced by approximating Err_ν by means of a discretization (details later). The smoothing of ν_{art} is in computational space because Err_ν will be determined and discretized

in computational space. The use of a constant smoothing coefficient α is sufficient because the artificial viscosity coefficient by itself is already a higher-order term.

An implicit smoothing equation like (5.14) is used frequently in moving adaptive grid methods. To see this, replace ν_{art} by smoothed grid-point concentration n, α by $\alpha(1+\alpha)$, and $c_\nu Err_\nu$ by non-smooth point concentration $\tilde{n}$, and discretize the equation by means of finite differences. The result is the equation that Dorfi and Drury use to ensure grid smoothness [5]. Huang and Russell show that smoothing of the monitor function and of the grid point concentration are equivalent [16]. Implicit smoothing of grid point concentration has been used in, e.g., [9, 25, 30, 31]. An explicit monitor smoothing technique approximating implicit smoothing has been used in [20, 22].

We will use an equation like (5.14) also for the smoothing of the error expressions that are used in the moving adaptive grid procedure, to eliminate higher-order errors that are not included properly. This automatically takes care of grid smoothness, i.e., no special measures are required to ensure that $\overline{x}$ is sufficiently smooth.

One variable may not be sufficiently smooth: $\overline{h}$. This is presently one of the main shortcomings of the method. Water level h is equal to the sum of bottom level z_b and water level d [cf. Equation (5.3)] and considered as a function of d. It is obvious that smoothness of $\overline{d}$ is no guarantee for smoothness of $\overline{h}$; that depends entirely on the given profile z_b that in practice may be highly irregular. Moreover, while $\overline{d}$ is a function that is piecewise linear in computational space and piecewise constant in computational time, the behavior of $\overline{h}$ can be anything depending on how z_b is specified. Usually, z_b is given as a piecewise linear function based on the available data of a channel's geometry. This does not make $\overline{h}$ a function that is linear per grid cell because the coordinates where the geometry is specified will rarely coincide with grid points. Discrete function $\overline{h}$ will generally not be piecewise constant in time either, even though z_b is constant in time. This is because z_b is constant in time in physical space, not in computational space.

All this is ignored at present. That is, also for $\overline{h}$ an expression like (5.13) is used, but a rather rough approximation is used to define the grid point values h_i^n:

$$h_i^n = d_i^n + z_b\left(x_i^{n-\frac{1}{2}}\right) , \tag{5.15}$$

with:

$$x_i^{n-\frac{1}{2}} = \overline{x}\left(\xi_i, \tau^{n-\frac{1}{2}}\right) = \frac{1}{2}\left(x_i^{n-1} + x_i^n\right) .$$

To obtain the best possible accuracy in time we evaluate the bottom level at the position of the moving grid in the middle of each time step.

5.4.1 Discretized Shallow-Water Equations

In the previous part we defined function approximations that are piecewise linear in computational space and piecewise constant in computational time. It is reasonable to assume that this is sufficient to construct a discretization that is second-order accurate

in space and first-order accurate in time, in agreement with the leading interpolation errors. However, the first derivative of a piecewise linear approximation is piecewise constant, and only second-order accurate at cell centers. The second derivative of a piecewise linear function is not defined. As a result, (5.11) and (5.12) cannot be discretized directly.

The obvious solution is to consider a weak formulation, integrating the model equations in space from cell center to cell center which are the only positions where the viscous fluxes can be evaluated with second-order accuracy. This automatically defines a finite volume technique, with volumes as indicated by the solid lines in Figure 5.2. Since all discrete functions have been specified, the discretization in space is now fully defined and straightforward to obtain.

It is important to evaluate the integrals in space of the different terms, with the functions replaced by their discrete approximations, with *at least* fourth-order accuracy. Second-order accurate approximations of the integrals, obtained by using, e.g., 1-point Gauss quadrature rules, are not allowed. Although that would lead to a second-order accurate discretization in space as well, it would *not* lead to a *compatible* discretization. It would introduce errors that are of the same order as the interpolation errors, thereby effectively modifying the defined discrete functions. In consequence, the interpolation error of these functions would not be the only source of discretization errors anymore. To make sure that the second-order interpolation errors are included with at least second-order accuracy, the integrals must be approximated fourth-order accurate or better.

We give two examples of how this compatible discretization in space works out in practice, keeping everything in time continuous for the moment (method of lines [MOL] approach). The time derivative of (5.12) is discretized in space according to:

$$\int_{\xi_{i-\frac{1}{2}}}^{\xi_{i+\frac{1}{2}}} \frac{\partial (x_\xi q)}{\partial \tau}\, d\xi \approx \int_{\xi_{i-\frac{1}{2}}}^{\xi_{i+\frac{1}{2}}} \frac{\partial (\overline{x}_\xi \overline{q})}{\partial \tau}\, d\xi$$
$$= \frac{\partial}{\partial \tau}\left((x_i - x_{i-1}) \frac{q_{i-1} + 3q_i}{8} + (x_{i+1} - x_i) \frac{3q_i + q_{i+1}}{8} \right) , \qquad (5.16)$$

while the space discretization of the artificial viscosity term of (5.12) reads:

$$\int_{\xi_{i-\frac{1}{2}}}^{\xi_{i+\frac{1}{2}}} \frac{\partial \left(\nu_{\text{art}} d / x_\xi \partial (q/d)/\partial \xi \right)}{\partial \xi}\, d\xi$$
$$\approx \left(\frac{\overline{\nu}_{\text{art}} \overline{d}}{\overline{x}_\xi} \frac{\partial \overline{q}/\overline{d}}{\partial \xi} \right)_{i+\frac{1}{2}} - \left(\frac{\overline{\nu}_{\text{art}} \overline{d}}{\overline{x}_\xi} \frac{\partial \overline{q}/\overline{d}}{\partial \xi} \right)_{i-\frac{1}{2}} , \qquad (5.17)$$

with:

$$\left(\frac{\overline{\nu}_{\text{art}} \overline{d}}{\overline{x}_\xi} \frac{\partial \overline{q}/\overline{d}}{\partial \xi} \right)_{i+\frac{1}{2}} = \left(\frac{\overline{\nu}_{\text{art}}}{\overline{x}_\xi} \frac{\partial \overline{q}}{\partial \xi} - \frac{\overline{\nu}_{\text{art}} \overline{q}}{\overline{x}_\xi \overline{d}} \frac{\partial \overline{d}}{\partial \xi} \right)_{i+\frac{1}{2}}$$
$$= \frac{\nu_{\text{art},i} + \nu_{\text{art},i+1}}{2} \left(\frac{q_{i+1} - q_i}{x_{i+1} - x_i} - \frac{q_i + q_{i+1}}{d_i + d_{i+1}} \frac{d_{i+1} - d_i}{x_{i+1} - x_i} \right) .$$

Note that this discretization has *not* been obtained by taking the average of $\overline{\nu}_{\text{art}}\overline{d}/\overline{x}_\xi$ and the difference of $\overline{q}/\overline{d}$ evaluated at the grid points. That would imply that certain combinations of variables rather than the variables themselves are approximated piecewise linearly. That is incompatible with the piecewise linear approximation of those variables used in the other terms [e.g., in time discretization (5.16)].

Our research has indicated that it is this mix of different variable approximations often encountered in numerical discretizations that makes it impossible to express discretization errors made in the equations in terms of errors in the numerical solution. One would expect this error to be some interpolation error, but if several interpolation errors have been mixed together during the discretization process, it is obviously not possible to recover a well-defined one afterwards. See also [2].

There are still a few details to be filled in. One concerns the discretization of (5.14) which reads:

$$\left(\frac{3}{4}+2\alpha\right)\nu_{\text{art},i}+\left(\frac{1}{8}-\alpha\right)\left(\nu_{\text{art},i-1}+\nu_{\text{art},i+1}\right)=\frac{c_\nu}{\Delta\xi}\int_{\xi_{i-\frac{1}{2}}}^{\xi_{i+\frac{1}{2}}} Err_\nu\, d\xi\ . \quad (5.18)$$

The discretization of the integral in the right-hand side is not critical; higher-order errors are irrelevant in the determination of ν_{art} and are damped by the smoothing anyway. The expression for Err_ν will be given in Section 5.5.3.

The discretization in time is simply obtained by evaluating the equations at the central time level $\tau^{n-\frac{1}{2}}=\frac{1}{2}(\tau^{n-1}+\tau^n)$ using the discrete function approximations in time. This is within $O(\Delta\tau^2)$ equal to the discretization obtained by integrating the equations over one time step, which is sufficient for compatibility since the time discretization is only first-order accurate.

As for the boundary conditions, they have to be applied at a cell center to allow the whole domain to be covered with finite volumes (cf. Figure 5.2). This is rather fortunate since both Dirichlet and Neumann boundary conditions can then be discretized with second-order accuracy using only two grid points. Compatibility is no problem either. We will consider only subcritical flow at boundaries, and apply Dirichlet conditions (either q or h imposed) supplemented with Equation (5.4) describing the behavior of the outgoing characteristic. The discretization of the characteristic equation is obtained in two steps:

- The flow equations (5.11) and (5.12) are discretized at the boundaries using the defined discrete approximation of the variables. Because the approximation is linear in space, second derivatives vanish, but the artificial viscosity term is still contributing to the discretization. To be able to discretize that term at the boundary, we write it as:

$$\frac{\partial}{\partial\xi}\left(\frac{\nu_{\text{art}}d}{x_\xi}\frac{\partial q/d}{\partial\xi}\right)=\frac{\nu_{\text{art}}}{x_\xi}\left(\frac{\partial^2 q}{\partial\xi^2}-\frac{q}{d}\frac{\partial^2 d}{\partial\xi^2}\right)+\frac{1}{x_\xi}\left(\frac{\partial\nu_{\text{art}}}{\partial\xi}-\frac{\nu_{\text{art}}}{d}\frac{\partial d}{\partial\xi}\right)\left(\frac{\partial q}{\partial\xi}-\frac{q}{d}\frac{\partial d}{\partial\xi}\right)\ .$$

Here, we have used $x_{\xi\xi} = 0$ since the grid is assumed to be uniform at the boundaries. The discretization of the second derivatives in the right-hand side vanishes at the boundaries; the compatible discretization of the other terms is straightforward.

- Using (5.4), the discretized flow equations at the boundaries are recombined to the equation for the outgoing characteristic.

We end this section with the boundary conditions for Equation (5.14) that takes care of the smoothing of ν_{art}. Assuming a constant error level and hence constant ν_{art} outside the domain, the discretization of this equation at the left boundary becomes [compare with (5.18)]:

$$\left(\frac{1}{2}+\alpha\right)\nu_{\text{art},0} + \left(\frac{1}{2}-\alpha\right)\nu_{\text{art},1} = c_\nu Err_{\nu,\frac{1}{2}} \ . \tag{5.19}$$

A similar condition is applied at the right boundary. It is not exactly compatible, but it ensures that the smoothing of Err_ν is extended up to and including the boundaries. Note that for $\alpha \to \infty$, condition (5.19) converges to a homogeneous Neumann condition for ν_{art}. This is the condition that is usually applied in combination with implicit smoothing [5, 16].

5.4.2 Iterative Solution Algorithm

With reference to Figure 5.1, we write the discretization of the flow equations per time step or steady state in the compact form $\overline{L}(\overline{u}^{n-1}, \overline{u}^n) = 0$. Solution vector $\overline{u}$, not to be confused with velocity u, consists of the two discrete functions $\overline{d}, \overline{q}$ whose vectors of grid point values $(d_i)^T$, $(q_i)^T$ are to be determined each time step or each steady state. We will omit in the description of the solution algorithm the solution at the previous time level $\overline{u}^{n-1} = (d_i^{n-1}, q_i^{n-1})^T$ since its contribution, only relevant in unsteady calculations, is known. Superscript n will be omitted as well. The determination of the grid-point coordinates x_i^n at the next time level or steady-state level will be considered later.

The basis of the iterative algorithm that we use to solve the algebraic system of nonlinear discretized flow equations is the Newton method. We employ a so-called pseudo or dual time-step technique to improve its robustness, solving each iteration:

$$\left(\frac{\Delta x}{\Delta t_{\text{pseu}}^{m-1}} + \left.\frac{\partial \overline{L}}{\partial \overline{u}}\right|^{m-1}\right)(\overline{u}^m - \overline{u}^{m-1}) = -\overline{L}(\overline{u}^{m-1}) \ . \tag{5.20}$$

Superscript m denotes the iteration level. The size of pseudo time step Δt_{pseu} is in [sec]. The added pseudo time-step coefficient $1/\Delta t_{\text{pseu}}^{m-1}$ is multiplied by local grid size Δx, reflecting the integration over the finite volumes of the flow equations. The residual in the right-hand side of (5.20) and the Jacobian $\partial\overline{L}/\partial\overline{u}$ (the block-tridiagonal linearization matrix) are evaluated at the previous iteration level $m-1$. Linearization of complicated coefficient ν_{art} is not feasible. Instead, we use underrelaxation with an

underrelaxation coefficient of 0.8 and multiply the part of $\partial\overline{L}/\partial\overline{u}$ stemming from the linearization of the artificial viscosity term by a factor 1.3. Without the latter, a much stronger underrelaxation of ν_{art} is usually required. We found that the combination of both measures is sufficient to stabilize the implicit iterative solver while maintaining a good performance, despite the fact that ν_{art}, which can be very important locally, is treated explicitly.

To ensure stability at the beginning of an iteration process and at the same time obtain a high convergence speed, we have made the pseudo time step per grid point inversely proportional to the local previous solution correction $(\overline{u}^{m-1} - \overline{u}^{m-2})$. The result is a pseudo time step that automatically compensates for the local Newton linearization error. To reduce the sensitivity of the method to irregular convergence behavior, the pseudo time steps are slightly underrelaxed. The iteration process is initialized by taking the initial pseudo CFL number $CFL^0_{\text{pseu}} = (|\overline{q}^0|/\overline{d}^0 + (g\overline{d}^0)^{1/2})\Delta t^0_{\text{pseu}}/\Delta x$ equal to 1. This gives a very conservative initial pseudo time-step value, which in fact is only required for steady-state problems where the initial condition $\overline{u}^0$ is generally strongly different from the final solution $\overline{u}^*$ satisfying $\overline{L}(\overline{u}^*) = 0$.

The discretization in space of the pseudo time-step term in (5.20) is partially upwind to stabilize the iteration process for high-speed flow calculations. To avoid stability problems due to sudden changes at the boundaries, we use solution-correction dependent underrelaxation of the boundary conditions that vanishes upon convergence. These two measures turn out to be very effective in practice.

The convergence behavior of the present method is as described in [24]: slow convergence in the first or *searching* phase where the algorithm tries to get in the neighborhood of $\overline{u}^*$ ($CFL_{\text{pseu}} = O(1)$); fast convergence in the second or *converging* phase where the algorithm feels the attraction of $\overline{u}^*$ ($CFL_{\text{pseu}} \gg 1$). We have observed that, depending on the difficulty of the system of equations to be solved, the searching phase can be virtually absent or can take up to several hundreds of iterations. The converging phase invariably takes only about 20 to 30 iterations to reduce the convergence error down to $(\max_i |d_i^m - d_i^{m-1}| < 10^{-11}, \max_i |q_i^m/d_i^m - q_i^{m-1}/d_i^{m-1}| < 10^{-13})$, the convergence criterion that we have used in all flow calculations. This type of convergence behavior is for problems with steep-gradient solutions where the underrelaxation of ν_{art} precludes quadratic convergence. Very easy problems with smooth solutions that require virtually no artificial smoothing converge faster.

An example of a system of equations that is difficult to solve is the calculation of a complex steady-state flow through a complex geometry, modeled on a grid consisting of 2000 grid points. The difficulty is entirely due to the fact that the initial condition (uniform discharge qW, constant water level h) is very different from the solution that is sought. In contrast, a system is easy to solve whenever a reasonable initial condition is available. This situation is encountered in unsteady calculations where there is hardly any searching phase; the initial condition is the solution of the previous physical time step and usually already very close to the solution of the next physical time step. Flow calculations inside the adaptive grid algorithm are also nearly all easy.

In adaptive grid calculations we solve within each time step or steady state alternately the grid equations and the flow equations until convergence of the coupled grid-flow problem (the outer grid iteration algorithm, cf. Section 5.6). The initial condition of the flow calculation on an updated grid is obtained by interpolation of the solution on the previous grid. As soon as the grid starts converging, this condition is reasonably to very close to the next solution, both for unsteady and steady-state calculations. Remark that the present iterative flow solver is relatively slow for easy systems of equations. The conservative initialization of the pseudo time step with $CFL^0_{\text{pseu}} = 1$ in combination with the applied underrelaxation of Δt_{pseu} prevents the occurrence of large pseudo time steps in the first few iterations, even in situations where that would be possible.

5.5 Error Analysis

The error analysis consists of the following steps (cf. Figure 5.1):

- define the close and smooth approximation $\widehat{u}$ of the numerical solution $\overline{u}$,
- determine the continuous differential operator $\widehat{L}$ satisfied by $\widehat{u}$,
- determine the difference between $\widehat{L}$ and the continuous differential operator $\widetilde{L}$ of the easy problem,
- express this difference in terms of the effect that is has on $\widehat{u}$.

The first step is fairly straightforward. The second step is more complicated and requires the correct application of Taylor-series expansions. This is realized by anticipating the result of the third step:

$$R(\widehat{u}) = \widehat{L}(\widehat{u}) - \widetilde{L}(\widehat{u}) \; . \tag{5.21}$$

Residual R indicates the numerical approximation error that is introduced in the easy system of partial differential equations by a particular numerical solution. The purpose of the fourth step is to transform residual R into the effect that it has locally on the numerical solution (the numerical modeling error).

The grid-dependent numerical modeling error is used in the right-hand side of (5.14) for the determination of ν_{art} (the feedback loop in Figure 5.1). Hence, minimizing the numerical modeling error by means of grid adaptation will not only minimize in some sense the difference between $\widehat{L}$ and $\widetilde{L}$ as far as is relevant for the accuracy of the numerical solution, but also the difference between $\widetilde{L}$ and L. Error minimization as such cannot be guaranteed, but the effect of all mechanisms responsible for the difference between the original solution u and the numerical solution $\overline{u}$ will be minimized.

The compatible scheme that we apply consists of the superposition of a discretization in space and a discretization in time (method of lines [MOL]). This permits analysis of the error separately in computational space and in computational time.

5.5.1 Error Analysis in Space

The first step is to define a suitable smooth and infinitely differentiable approximation $\widehat{u}$ of the piecewise linear numerical function approximation $\overline{u}$ that we have been using. In principle that is easy since the space of infinitely differentiable functions is dense in the space of piecewise polynomial functions. In practice it is slightly more complicated because we need an algebraic description of that smooth function.

There are two reasonable ways to construct such a function: one is to connect grid-point values by a smooth function; the other is to connect cell-center values. Both possibilities are illustrated in Figure 5.3 by, respectively, the function $\widehat{u}(\gamma = 0)$ and $\widehat{u}(\gamma = 1)$. It can be seen that the closest smooth fit to $\overline{u}$ lies somewhere in between these two functions.

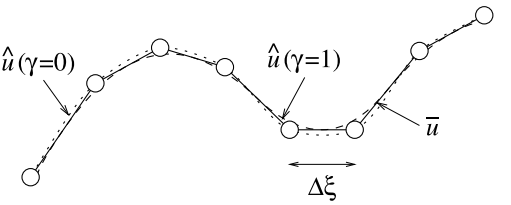


FIGURE 5.3
Smooth approximations of a piecewise linear function.

This suggests consideration of the one-parameter family of functions obtained by means of linear interpolation between $\widehat{u}(\gamma = 0)$ and $\widehat{u}(\gamma = 1)$, with γ the interpolation coefficient. It is not relevant to determine how these functions can be constructed from the grid-point values of $\overline{u}$. What matters is the inverse: $\overline{u}$ expressed in terms of $\widehat{u}$ and its derivatives at the grid points, since this is the information required to construct the Taylor-series expansions. Skipping the details of the construction, we will just present the result:

$$\overline{u}(\xi) = \overline{u}\,(\xi_{i-1} + \beta\Delta\xi) = f_i^-(1-\beta) - \gamma g_i^-(1-\beta) + O\left(\Delta\xi^6\right) , \qquad (5.22)$$

or:

$$\overline{u}(\xi) = \overline{u}\,(\xi_{i-1} + \beta\Delta\xi) = f_{i-1}^+(\beta) - \gamma g_{i-1}^+(\beta) + O\left(\Delta\xi^6\right) , \qquad (5.23)$$

with $\xi_{i-1} \le \xi \le \xi_i$, $i = 1, \ldots, I+1$, local coordinate $\beta = (\xi - \xi_{i-1})/\Delta\xi \in [0, 1]$,

and with:

$$f_i^{\pm}(\beta) = \widehat{u}_i + \beta\left(\pm\Delta\xi\widehat{u}_i^{i} + \frac{\Delta\xi^2}{2}\widehat{u}_i^{ii} \pm \frac{\Delta\xi^3}{6}\widehat{u}_i^{iii} + \frac{\Delta\xi^4}{24}\widehat{u}_i^{iv} \pm \frac{\Delta\xi^5}{120}\widehat{u}_i^{v}\right) , \tag{5.24}$$

$$g_i^{\pm}(\beta) = \frac{\Delta\xi^2}{8}\widehat{u}_i^{ii} \pm \beta\frac{\Delta\xi^3}{8}\widehat{u}_i^{iii} + (24\beta - 5)\frac{\Delta\xi^4}{384}\widehat{u}_i^{iv} \pm \beta\frac{\Delta\xi^5}{128}\widehat{u}_i^{v} . \tag{5.25}$$

The superscripts in (5.24) and (5.25) indicate the order of differentiation with respect to ξ while subscript i indicates the grid point (e.g., $\widehat{u}_i^{iii} = \partial^3\widehat{u}/\partial\xi^3\big|_{\xi=\xi_i}$). Each function in (5.22), (5.23), (5.24), and (5.25) is also a function of computational time coordinate τ, but for clarity this has not been indicated.

For $\beta = 1$ expression (5.24) is the well-known Taylor-series expansion up to $O(\Delta\xi^6)$ of grid-point value $u_{i\pm1}$ in terms of derivatives at $\xi = \xi_i$ when $\widehat{u}$ passes through the grid-point values ($\gamma = 0$). The expressions (5.22) and (5.23) for $\gamma = 0$ are obtained by combining this with $\overline{u}(\xi) = \overline{u}(\xi_{i-1} + \beta\Delta\xi) = (1-\beta)u_{i-1} + \beta u_i$ [cf. (5.13)].

The second term in the right-hand side of (5.22) and (5.23) can be viewed as a γ-dependent correction that takes care of the shift required when the smooth approximation is defined differently. From both (5.22) and (5.23) we obtain (expanding the right-hand sides at $\xi = \xi_{i-\frac{1}{2}}$):

$$\overline{u}\left(\xi_{i-\frac{1}{2}}\right) = \widehat{u}\left(\xi_{i-\frac{1}{2}}\right) + (1-\gamma)\left(\frac{\Delta\xi^2}{8}\widehat{u}_{i-\frac{1}{2}}^{ii} + \frac{\Delta\xi^4}{384}\widehat{u}_{i-\frac{1}{2}}^{iv}\right) + O\left(\Delta\xi^6\right) . \tag{5.26}$$

This shows that $\gamma = 1$ corresponds with a smooth function fit through the cell-center values. In fact, the equation $\overline{u}(\xi_{i-\frac{1}{2}}) = \widehat{u}(\xi_{i-\frac{1}{2}})$ has been used to derive the expression for $g_i^{\pm}$. Another equation that has been used in that derivation is $g_{i-1}^{+}(\beta) = g_i^{-}(1-\beta) + O(\Delta\xi^6)$. We also have $f_{i-1}^{+}(\beta) = f_i^{-}(1-\beta) + O(\Delta\xi^6)$. The latter two equations must hold for the two expressions (5.22) and (5.23) to be equivalent, and hence continuous at the grid points. This is trivial for $\gamma = 0$ when $\widehat{u}$ passes through the grid-point values, but it is a condition to be imposed when deriving the expression for $g_i^{\pm}$. It turns out that this, together with $\overline{u}(\xi_{i-\frac{1}{2}}) = \widehat{u}(\xi_{i-\frac{1}{2}})$ for $\gamma = 1$, fixes $g_i^{\pm}$ completely.

The compatible discretization in space of the shallow-water equations has been obtained upon the integration of the easy equations with added artificial smoothing term over the finite volumes $[\xi_{i-\frac{1}{2}}, \xi_{i+\frac{1}{2}}]$, replacing each continuous function by its piecewise linear approximation (Section 5.4.1). In other words, the discretized flow equations $\overline{L}(\overline{u}) = 0$ can also be written as:

$$\overline{L}_i(\overline{u}) = \int_{\xi_{i-\frac{1}{2}}}^{\xi_{i+\frac{1}{2}}} \widetilde{L}(\overline{u})\, d\xi = 0 , \ i = 1, \ldots, I , \tag{5.27}$$

where $\widetilde{L}$ stands for the easy shallow-water equations transformed to computational space [Equations (5.11) and (5.12)], while this time $\overline{u}$ represents the collection of

piecewise linear functions that have been used ($\overline{x}, \overline{d}, \overline{q}, \overline{h}, \overline{P}$, and $\overline{v}_{\text{art}}$). See Figure 5.1 and the construction of the discretization in Section 5.4.1.

It is important to realize that a straightforward Taylor-series expansion of the left-hand side of (5.27) at $\xi = \xi_i$ does *not* produce the second-order residual. This approach can only be used if the discretization represents an approximation of the model equations at a specific point (finite difference methods, collocation-point methods). Finite volume discretizations (and finite element discretizations), however, approximate a (weighed) *integral* of the model equations. A one-point Taylor-series expansion would then introduce an integral approximation error that is first order in general, and second order if the expansion is at the central quadrature point, as is the case here. This is one order lower or of the same order as the residual to be determined and hence unacceptable.

There is a simple way to avoid the problem of integration errors. The equivalence between the discretized problem and the equivalent problem implies that we have [cf. Figure 5.1 and (5.27)]:

$$\overline{L}_i(\overline{u}) = \int_{\xi_{i-\frac{1}{2}}}^{\xi_{i+\frac{1}{2}}} \widetilde{L}(\overline{u})\, d\xi = \int_{\xi_{i-\frac{1}{2}}}^{\xi_{i+\frac{1}{2}}} \widehat{L}(\widehat{u})\, d\xi \ , \ i = 1, \ldots, I \ . \tag{5.28}$$

Since the discretization is obtained by integration over finite volumes, the equivalence becomes an identity per finite volume. Using (5.28), we obtain from (5.21):

$$\int_{\xi_{i-\frac{1}{2}}}^{\xi_{i+\frac{1}{2}}} R(\widehat{u})\, d\xi = \overline{L}_i(\overline{u}) - \int_{\xi_{i-\frac{1}{2}}}^{\xi_{i+\frac{1}{2}}} \widetilde{L}(\widehat{u})\, d\xi \ , \ i = 1, \ldots, I \ . \tag{5.29}$$

The integral approximation error that makes it impossible to determine the residual from a Taylor-series expansion of (5.27) at a single point does not pose a problem here. By expanding $\overline{L}_i(\overline{u})$ as well as the finite volume integrals of R and $\widetilde{L}(\widehat{u})$ at the same point (not necessarily the quadrature point), the terms that would perturb the error analysis cancel out. Moreover, residual R can be determined with an accuracy of $O(\Delta\xi^2)$ by just developing the right-hand side of (5.29) at the quadrature point $\xi = \xi_i$ and approximating the left-hand side by $\Delta\xi R_i$.

There is yet another snag to be avoided. Analyzing (5.29) would imply that we consider a problem in computational space. The result would be a residual that indicates the error in computational space, not in physical space. However, we are dealing with a physical problem in physical space and are only concerned with the numerical accuracy in physical space. On the other hand, it is perfectly acceptable to formulate the numerical modeling error pertaining to the physical problem in computational space. The computational space is a convenient means to construct and analyze discretizations, but its use should not have a negative effect on the error analysis.

The error in (5.29) is a subtle one and related to the coordinate transformation. The coordinate transformation used for the discretized equations is defined by $\overline{x}(\xi, \tau)$, while the one of the equivalent equations is $\widehat{x}(\xi, \tau)$. Hence, integrating both over the *same* finite volume in computational space means that they are integrated over

finite volumes of *different* size in physical space. We see here why it is essential to consider separate coordinate transformations for the different problems with which we are dealing (see the beginning of Section 5.4). To make sure that the equivalent equations will be equivalent with the discretized equations in *physical* space, we should consider:

$$\int_{\overline{x}_{i-\frac{1}{2}}}^{\overline{x}_{i+\frac{1}{2}}} \frac{R(\widehat{u})}{\widehat{x}_\xi}\, d\widehat{x} = \int_{\overline{x}_{i-\frac{1}{2}}}^{\overline{x}_{i+\frac{1}{2}}} \frac{\widetilde{L}(\overline{u})}{\overline{x}_\xi}\, d\overline{x} - \int_{\overline{x}_{i-\frac{1}{2}}}^{\overline{x}_{i+\frac{1}{2}}} \frac{\widetilde{L}(\widehat{u})}{\widehat{x}_\xi}\, d\overline{x}\ ,\ i = 1, \ldots, I\ . \qquad (5.30)$$

$\widetilde{L}(\overline{u})/\overline{x}_\xi = 0$ and $\widetilde{L}(\widehat{u})/\widehat{x}_\xi = 0$ can be viewed as equations formulated in physical space, cf. the equations in computational space (5.11) and (5.12) that have been obtained upon multiplying the original equations (5.1) and (5.2) by x_ξ.

Using (5.26) to replace the integral boundaries $\overline{x}_{i-\frac{1}{2}}$ and $\overline{x}_{i+\frac{1}{2}}$ of the second term in the right-hand side, we obtain from (5.30) (recall that $\widehat{x}_\xi$ and $\widehat{x}^i$ both denote $\partial\widehat{x}/\partial\xi$):

$$\begin{aligned}
&\Delta\xi \left(R(\widehat{u})|_{\xi_i} + O\left(\Delta\xi^2\right)\right) \\
&= \int_{\overline{x}_{i-\frac{1}{2}}}^{\overline{x}_{i+\frac{1}{2}}} \frac{\widetilde{L}(\overline{u})}{\overline{x}_\xi}\, d\overline{x} - \int_{\widehat{x}_{i-\frac{1}{2}} + \frac{1-\gamma}{8}\Delta\xi^2 \widehat{x}^{ii}_{i-\frac{1}{2}} + O(\Delta\xi^4)}^{\widehat{x}_{i+\frac{1}{2}} + \frac{1-\gamma}{8}\Delta\xi^2 \widehat{x}^{ii}_{i+\frac{1}{2}} + O(\Delta\xi^4)} \frac{\widetilde{L}(\widehat{u})}{\widehat{x}_\xi}\, d\widehat{x} \\
&= \int_{\xi_{i-\frac{1}{2}}}^{\xi_{i+\frac{1}{2}}} \widetilde{L}(\overline{u})\, d\xi - \int_{\xi_{i-\frac{1}{2}}}^{\xi_{i+\frac{1}{2}}} \widetilde{L}(\widehat{u})\, d\xi \\
&\quad - \frac{1-\gamma}{8}\Delta\xi^3 \frac{\partial}{\partial\xi}\left(\frac{\widehat{x}^{ii}\widetilde{L}(\widehat{u})}{\widehat{x}^i}\right)\Bigg|_{\xi_i} + O\left(\Delta\xi^5\right)\ ,\ i = 1, \ldots, I\ . \qquad (5.31)
\end{aligned}$$

The essential difference between (5.29) and (5.31) is the physical-to-computational-space correction in the right-hand side of (5.31). It vanishes for $\gamma = 1$ when we have $[\overline{x}_{i-\frac{1}{2}}, \overline{x}_{i+\frac{1}{2}}] = [\widehat{x}_{i-\frac{1}{2}}, \widehat{x}_{i+\frac{1}{2}}]$.

Since everything is now formulated in computational space again, the determination of R is fairly straightforward. Term by term the finite volume integrals of $\widetilde{L}(\overline{u})$ and of $\widetilde{L}(\widehat{u})$ are expanded in a Taylor series at $\xi = \xi_i$, using (5.22) and (5.23) to replace $\overline{u}$ by $\widehat{u}$. For example, using the discretization in space (5.16), the evaluation of the right-hand side of (5.31) for the time derivative $\partial(x_\xi q)/\partial\tau$ reads:

$$\begin{aligned}
&\int_{\xi_{i-\frac{1}{2}}}^{\xi_{i+\frac{1}{2}}} \frac{\partial(\overline{x}_\xi \overline{q})}{\partial\tau}\, d\xi - \int_{\xi_{i-\frac{1}{2}}}^{\xi_{i+\frac{1}{2}}} \frac{\partial(\widehat{x}_\xi \widehat{q})}{\partial\tau}\, d\xi - \frac{1-\gamma}{8}\Delta\xi^3 \frac{\partial}{\partial\xi}\left(\frac{\widehat{x}^{ii}}{\widehat{x}^i}\frac{\partial(\widehat{x}_\xi\widehat{q})}{\partial\tau}\right)\Bigg|_{\xi_i} \\
&\quad = \Delta\xi^3 \frac{\partial}{\partial\tau}\left(\frac{2-3\gamma}{24}\widehat{x}^i\widehat{q}^{ii} + \frac{1}{24}\widehat{x}^{ii}\widehat{q}^i + \frac{1-\gamma}{8}\widehat{x}^{iii}\widehat{q}\right)_{\xi_i}
\end{aligned}$$

$$
\begin{aligned}
&- \frac{1-\gamma}{8}\Delta\xi^3\left(\frac{\partial}{\partial\tau}\left(\widehat{x}^{ii}\widehat{q}^{\,i}+\widehat{x}^{iii}\widehat{q}\right)-\frac{\partial}{\partial\xi}\left(\widehat{x}_\xi\widehat{q}\,\frac{\partial}{\partial\tau}\frac{\widehat{x}^{ii}}{\widehat{x}^i}\right)\right)_{\xi_i}+O\left(\Delta\xi^5\right)\\
&=\frac{2-3\gamma}{24}\Delta\xi^3\frac{\partial}{\partial\tau}\left(\widehat{x}^i\left(\widehat{q}^{\,ii}-\frac{\widehat{x}^{ii}}{\widehat{x}^i}\widehat{q}^{\,i}\right)\right)_{\xi_i}\\
&+\frac{1-\gamma}{8}\Delta\xi^3\frac{\partial}{\partial\xi}\left(\widehat{q}\left(\widehat{x}^{ii}_\tau-\frac{\widehat{x}^{ii}}{\widehat{x}^i}\widehat{x}^i_\tau\right)\right)_{\xi_i}+O\left(\Delta\xi^5\right)\ . \qquad (5.32)
\end{aligned}
$$

In the first term of the right-hand side of (5.32) one recognizes the second-order interpolation error of $\overline{q}$ in *physical* space integrated over the finite volume $[\overline{x}_{i-\frac{1}{2}},\overline{x}_{i+\frac{1}{2}}]$ and formulated in computational space:

$$
\begin{aligned}
\int_{\overline{x}_{i-\frac{1}{2}}}^{\overline{x}_{i+\frac{1}{2}}}(\overline{q}-\widehat{q})\,dx&=\int_{\xi_{i-\frac{1}{2}}}^{\xi_{i+\frac{1}{2}}}\overline{x}_\xi\overline{q}\,d\xi-\int_{\xi_{i-\frac{1}{2}}}^{\xi_{i+\frac{1}{2}}}\widehat{x}_\xi\widehat{q}\,d\xi-\frac{1-\gamma}{8}\Delta\xi^3\left.\frac{\partial\widehat{x}^{ii}\widehat{q}}{\partial\xi}\right|_{\xi_i}\\
&=\Delta\xi^3\left(\frac{2-3\gamma}{24}\widehat{x}^i\widehat{q}^{\,ii}+\frac{1}{24}\widehat{x}^{ii}\widehat{q}^{\,i}+\frac{1-\gamma}{8}\widehat{x}^{iii}\widehat{q}\right)_{\xi_i}\\
&\quad-\frac{1-\gamma}{8}\Delta\xi^3\left(\widehat{x}^{ii}\widehat{q}^{\,i}+\widehat{x}^{iii}\widehat{q}\right)_{\xi_i}+O(\Delta\xi^5)\\
&=\frac{2-3\gamma}{24}\Delta\xi^3\widehat{x}^i_i\left(\widehat{q}^{\,ii}-\frac{\widehat{x}^{ii}}{\widehat{x}^i}\widehat{q}^{\,i}\right)_{\xi_i}+O\left(\Delta\xi^5\right)\ . \qquad (5.33)
\end{aligned}
$$

Using $\partial^2\widehat{q}/\partial x^2=1/\widehat{x}_\xi\,\partial(\widehat{q}_\xi/\widehat{x}_\xi)/\partial\xi=(\widehat{q}^{\,ii}-\widehat{x}^{ii}\widehat{q}^{\,i}/\widehat{x}^i)/(\widehat{x}^i)^2$, this can also be written as:

$$
\int_{\overline{x}_{i-\frac{1}{2}}}^{\overline{x}_{i+\frac{1}{2}}}(\overline{q}-\widehat{q}\,)\,dx=\frac{2-3\gamma}{24}\Delta\overline{x}_i^3\left.\frac{\partial^2\widehat{q}}{\partial x^2}\right|_{\xi_i}+O\left(\Delta\overline{x}_i^5\right)\ ,
$$

with $\Delta\overline{x}_i=\overline{x}_{i+\frac{1}{2}}-\overline{x}_{i-\frac{1}{2}}$.

The space discretization of the other terms in the flow equations is analyzed in the same way. This shows for example that the second term in the right-hand side of (5.32) cancels against the part of the residual of the convection term $\partial(\overline{q}/\overline{d}-\overline{x}_\tau)\overline{q}/\partial\xi$ stemming from the interpolation error in physical space of $\overline{x}_\tau$. Skipping the lengthy derivation and reformulating the residual in terms of errors in the variables, we obtain for the momentum equation:

$$
\begin{aligned}
&\frac{\partial}{\partial\tau}\left[\widehat{x}_\xi\left(\widehat{q}+\left(\gamma'-\frac{1}{24}\right)D_\xi(\widehat{q})\right)\right]\\
&\quad+\frac{\partial}{\partial\xi}\left[\left(\frac{\widehat{q}+\gamma' D_\xi(\widehat{q})}{\widehat{d}+\gamma' D_\xi(\widehat{d})}-\widehat{x}_\tau\right)\left(\widehat{q}+\gamma' D_\xi(\widehat{q})\right)\right]
\end{aligned}
$$

$$
\begin{aligned}
&+ g\left(\widehat{d} + \gamma' D_\xi(\widehat{d})\right) \frac{\partial}{\partial \xi}\left[\widehat{h} + \gamma' D_\xi(\widehat{h})\right] + g\frac{\Delta\xi^2}{24}\left(\widehat{d}^{\,i}\widehat{h}^{\,ii} - \widehat{d}^{\,ii}\widehat{h}^{\,i}\right) \\
&= \frac{\partial}{\partial \xi}\Bigg[\left(\widehat{v}_{\text{art}} + \gamma' D_\xi(\widehat{v}_{\text{art}})\right) \\
&\quad \times \left(\frac{1}{\widehat{x}_\xi}\frac{\partial}{\partial \xi}\left(\widehat{q} + \left(\gamma' - \frac{1}{12}\right) D_\xi(\widehat{q})\right) + \frac{D_\xi(\widehat{q})}{12}\frac{\partial(1/\widehat{x}_\xi)}{\partial \xi}\right)\Bigg] \\
&\quad - \frac{\partial}{\partial \xi}\Bigg[\frac{\left(\widehat{v}_{\text{art}} + \gamma' D_\xi(\widehat{v}_{\text{art}})\right)\left(\widehat{q} + \gamma' D_\xi(\widehat{q})\right)}{\widehat{d} + \gamma' D_\xi(\widehat{d})} \\
&\quad \times \left(\frac{1}{\widehat{x}_\xi}\frac{\partial}{\partial \xi}\left(\widehat{d} + (\gamma' - \frac{1}{12}) D_\xi(\widehat{d})\right) + \frac{D_\xi(\widehat{d})}{12}\frac{\partial(1/\widehat{x}_\xi)}{\partial \xi}\right)\Bigg] \\
&\quad + O\left(\Delta\xi^4\right), \qquad (5.34)
\end{aligned}
$$

with $\gamma' = (1-\gamma)/8$ and with

$$
D_\xi = \Delta\xi^2\left(\frac{\partial^2}{\partial \xi^2} - \frac{\widehat{x}^{\,ii}}{\widehat{x}^{\,i}}\frac{\partial}{\partial \xi}\right) \qquad (5.35)
$$

the differential operator for the second-order interpolation error in physical space.

Equivalent partial differential equation (5.34) is an extended version of the result presented in [2]. For convenience we have omitted the bottom friction term $\overline{x}_\xi g(\overline{P}\overline{q}|\overline{q}|)(W\overline{d}^2C^2)$ whose equivalent expression is obtained by replacing $\overline{x}_\xi$ by $\widehat{x}_\xi$ and all other variables by their equivalent smooth approximation plus average interpolation error, as in the time derivative term in the left-hand side of (5.34).

One recognizes in (5.34) momentum equation (5.12) with the artificial viscosity term written in the form used for its discretization (5.17). The difference is due to discretization errors and consists, if error terms of $O(\Delta\xi^4)$ can be neglected, almost entirely of second-order interpolation errors in physical space (the D_ξ-terms). With a small exception in the viscosity term, these errors perturb the local value of variables just like piecewise linear interpolations would do. For example, the error in the time derivative equals the second-order interpolation error averaged over the finite volumes [cf. (5.33)] and vanishes for $\gamma' - 1/24 = 0 \rightarrow \gamma = 2/3$. The error in the convection term is determined by the second-order interpolation error at the cell centers $\xi_{i-\frac{1}{2}}$ which is where the fluxes are evaluated, and vanishes for $\gamma' = 0 \rightarrow \gamma = 1$. It seems that the optimal value of γ, defining the best possible smooth fit in space, is somewhere in between.

This result demonstrates the compatibility between the discretization of the flow equations and the applied discrete function approximations. If the compatible scheme is used and sufficient smoothing is applied, the leading part of the residual (5.21), i.e., the difference between equivalent equation (5.34) and easy equation (5.12), can be reformulated directly in terms of local errors in the numerical solution that are of the same form and size as the leading interpolation errors. There is no need for solving a local dual problem.

Only the equivalent form of the hydrostatic pressure term [the third line of (5.34)] contains an error term that does not depend on interpolation errors in physical space. This is because the discretization of this *non*-conservative term involves the *weighed* integration (by $\overline{d}$) of $\partial\overline{h}/\partial\xi$ over the volume $[\xi_{i-\frac{1}{2}}, \xi_{i+\frac{1}{2}}]$, while this term is only a second-order accurate approximation of $\partial h/\partial\xi$ at the volume boundaries. Using (5.3), we can write that error term as:

$$g\frac{\Delta\xi^2}{24}\left(\widehat{d}^i\widehat{h}^{ii} - \widehat{d}^{ii}\widehat{h}^i\right) = g\frac{\Delta\xi^2}{24}\left(\widehat{d}^i z_b^{ii} - \widehat{d}^{ii} z_b^i\right) . \tag{5.36}$$

The effect of this term is small for a nearly horizontal bottom, but can become quite large if the bottom is non-smooth. On the other hand, since z_b is usually given as a piecewise linear function in space we have that $z_b(x)$ is of the form $A + Cx$ almost everywhere, with $|C| \ll 1$ because of the assumptions underlying the shallow-water model (Section 5.3). Hence $\Delta\xi^2|\widehat{d}^i z_b^{ii} - \widehat{d}^{ii} z_b^i| \ll |\widehat{x}^i D_\xi(\widehat{d})|$, except near the locations where the value of z_b is specified. Some form of geometry smoothing (not included yet) is required to ensure that the effect of (5.36) is negligible everywhere. See also Section 5.4.

5.5.2 Error Analysis in Time

The analysis of the error in time is similar to the error analysis in space, but less complex. The accuracy of the scheme is lower (first order in time as opposed to second order in space), only two time levels are used (as opposed to the three-point stencil in space), the coordinate transformation is considered to be linear in time (5.9), and there are no second derivatives in time. It is fairly easy to derive that the residual in time, assuming components of $O(\Delta\tau^2)$ and higher can be neglected, can be rewritten in expressions of the form:

$$\frac{1}{2}D_\tau(\widehat{d}) = \frac{1}{2}\Delta\tau\frac{\partial\widehat{d}}{\partial\tau} . \tag{5.37}$$

This is the first-order interpolation error in *computational* time that can be reduced by decreasing the size of the time step but also by aligning the grid with the solution. The latter property is the result of using a higher-order piecewise linear approximation in time for the coordinate mapping [i.e., (5.10)] in combination with the backward piecewise constant discretization (5.13) for the other variables.

5.5.3 Error in Discretized Shallow-Water Equations

Combining the results of the previous two sections, we conclude that the local effect of the discretization error on the numerical solution of the shallow-water equations (the numerical modeling error) can be reasonably well approximated by the sum of the second-order interpolation error in physical space and the first-order interpolation error in computational time:

$$Err_d = Err_d^{\text{space}} + Err_d^{\text{time}} \approx |D_\xi(\widehat{d})| + c_\gamma|D_\tau(\widehat{d})| \approx |D_\xi(\overline{d})| + c_\gamma|D_\tau(\overline{d})| , \tag{5.38}$$

$$Err_q = Err_q^{\text{space}} + Err_q^{\text{time}} \approx |D_\xi(\widehat{q})| + c_\gamma |D_\tau(\widehat{q})| \approx |D_\xi(\overline{q})| + c_\gamma |D_\tau(\overline{q})| \ , \tag{5.39}$$

with D_ξ and D_τ as in (5.35) and (5.37). Since all functions are smooth (they should be, by construction), the derivatives of $\widehat{d}$, $\widehat{q}$, and $\widehat{x}$ in D_ξ and D_τ can be approximated very well by the derivatives of $\overline{d}$, $\overline{q}$, and $\overline{x}$ using simple finite differences. We see that, although the smooth fits of the piecewise polynomial numerical approximations are the basis of the error analysis, these smooth functions are actually not required. It suffices to know that they exist.

Scaling of the error expressions (5.38) and (5.39) with a constant coefficient is irrelevant. However, the relative weighing between the interpolation error in space and the one in time is important. It is determined by parameter c_γ that, in view of the coefficients of the errors in space in (5.34) and the coefficient of the error in time (5.37), should have a value of about $(1/2)/(1/12) = 6$.

The reliability of error approximations (5.38) and (5.39) depends to a large extent on the smoothness of the bottom, cf. the discussion above definition (5.15) and below equation (5.36). It may therefore be useful to consider instead of (5.38):

$$Err_h = Err_h^{\text{space}} + Err_h^{\text{time}} \approx |D_\xi(\widehat{h})| + c_\gamma |D_\tau(\widehat{h})| \approx |D_\xi(\overline{h})| + c_\gamma |D_\tau(\overline{h})| \ . \tag{5.40}$$

For the determination of the artificial viscosity coefficient ν_{art} and the adaptation of the grid, the error in water depth d or water level h and in depth-integrated velocity q should be combined in some way to a single error expression. A convenient option is to base that combination on an energy measure like the energy head. Since the energy head itself may be (nearly) constant [cf. expression (5.6)] we will consider the sum of the error in the potential part and in the kinetic part of the flow energy. From a physical point of view this seems to be a fairly meaningful choice. So for the grid adaptation we consider the error expression:

$$\begin{aligned} Err = & Err^{\text{space}} + Err^{\text{time}} \approx |D_\xi(\overline{d})| + \left| \frac{\overline{q} D_\xi(\overline{q})}{g\overline{d}^2} - \frac{\overline{q}^2 D_\xi(\overline{d})}{g\overline{d}^3} \right| \\ & + c_\gamma \left(|D_\tau(\overline{d})| + \left| \frac{\overline{q} D_\tau(\overline{q})}{g\overline{d}^2} - \frac{\overline{q}^2 D_\tau(\overline{d})}{g\overline{d}^3} \right| \right) \ , \end{aligned} \tag{5.41}$$

which has been obtained by linearizing $\frac{1}{2}\overline{q}^2/(g\overline{d}^2) - \frac{1}{2}\widehat{q}^2/(g\widehat{d}^2)$, the interpolation error in the kinetic part $\frac{1}{2}u^2/g$ of the energy head, with respect to $D_\xi(\overline{q})$, $D_\xi(\overline{d})$, $D_\tau(\overline{q})$, and $D_\tau(\overline{d})$.

The dimension of Err is [m]. Therefore, expression (5.41) cannot be used for Err_ν, the error in the right-hand side of artificial viscosity equation (5.14). Instead we use:

$$Err_\nu = \Delta\xi \overline{x}_\xi \left(\frac{1}{2}\sqrt{\frac{g}{\overline{d}}} |D_\xi(\overline{h})| + \sqrt{\frac{1}{2}} \left| \frac{D_\xi(\overline{q})}{\overline{d}} - \frac{\overline{q} D_\xi(\overline{d})}{\overline{d}^2} \right| \right) \ , \tag{5.42}$$

obtained by linearizing the interpolation error in the square root of the potential part and the kinetic part of the energy head, multiplied by $\sqrt{g}$ and by grid size $\Delta\xi\overline{x}_\xi$. Only the interpolation error in space is considered; it is not necessary to include the error in time in the error feedback mechanism because the dissipative backward Euler scheme that we use provides enough smoothing in time. It is easily verified that (5.42) is as described below Equation (5.14) and satisfies the artificial dissipation requirements mentioned in Section 5.2.

Notice that the interpolation error in the water *depth* is used in (5.41), while in (5.42) we look at the interpolation error in the water *level.* Ideally, these two should be nearly equivalent; at present they are not because of the absence of geometry smoothing. Since a smooth water surface implies a smooth hydrostatic pressure term in (5.2) and hence a smooth flow, the use of $D_\xi(\overline{h})$ is preferred in (5.42). This avoids unnecessarily large values of ν_{art} occurring in regions with large variations in water depth but with small flow velocities. However, numerical experiments have shown that it is best to use $D_\xi(\overline{d})$ and $D_\tau(\overline{d})$ in (5.41), probably because in this way the algorithm "feels" to some extent the nonsmoothness of the bottom. All this is rather a matter of compromising and certainly not ideal, which is confirmed by the results that we will present.

5.6 Error-Minimizing Grid Adaptation

In the previous section we derived expression (5.41): a physically meaningful approximation of the numerical modeling error. *Err* is a function of the numerical solution that is sought, as well as of the piecewise linear coordinate transformation $\overline{x}(\xi,\tau)$ defined by the grid point coordinates x_i^n [cf. (5.10)]. The grid points form a set of degrees of freedom that can be chosen in any convenient way. Here, we choose to determine them by considering the optimization problem:

$$\text{solve:} \quad \min_{\overline{x}(\xi,\tau)} \| \, Err \, \|_1 \; . \tag{5.43}$$

Numerical modeling error *Err* is measured in L_1-norm for reasons explained in Section 5.1. Putting (5.41) and (5.43) together, we obtain:

$$\begin{aligned} &\text{solve:} \quad \min_{\overline{x}(\xi,\tau)} \int_{t^0}^{t^N} \int_{x_{\text{left}}}^{x_{\text{right}}} \left(Err^{\text{space}} + Err^{\text{time}} \right) dx \, dt \\ = \; &\text{solve:} \quad \min_{\overline{x}(\xi,\tau)} \int_{t^0}^{t^N} \int_{\xi_{\frac{1}{2}}}^{\xi_{I+\frac{1}{2}}} \overline{x}_\xi \left(Err^{\text{space}} + Err^{\text{time}} \right) d\xi \, dt \; , \end{aligned} \tag{5.44}$$

with x_{left} and x_{right} the coordinates of the left and right boundary in physical space, and $\xi_{\frac{1}{2}}$ and $\xi_{I+\frac{1}{2}}$ the coordinates of these boundaries in computational space [cf. Figure 5.2 and (5.10)].

Solving the global optimization problem (5.44) as such would be prohibitively complicated and expensive. Following the usual approach in moving adaptive grid methods, we therefore determine the grid point coordinates x_i^n time level by time level, keeping the grid points x_i^{n-1} fixed at the previous time level. So instead of (5.44) we consider the sequence of optimization problems:

$$\text{solve: } \min_{\overline{x}^n(\xi)} \int_{t^{n-1}}^{t^n} \int_{\xi_{\frac{1}{2}}}^{\xi_{I+\frac{1}{2}}} \overline{x}_\xi \left(Err^{\text{space}} + Err^{\text{time}} \right) d\xi \, dt \ , \ n = 1, \ldots, N \ . \quad (5.45)$$

This way of grid optimization may however lead to very poor grid point redistributions. It forces the grid points to move in a direction that may be totally different from the direction of propagation of steep gradients, in order to get the points concentrated at the right locations. For example, after a sudden change of the condition imposed at a boundary, grid points have to move toward that boundary which may seriously conflict with other requirements. Even the "optimal" grid may then yield a poor error reduction. Small time steps should be taken to compensate for the loss in accuracy when the grid lines can no longer be aligned with steep gradients in the space-time domain. This strongly reduces the efficiency and hence the advantages of moving grid adaptation.

The above problem is caused by the fact that (5.45) can be a very poor approximation of (5.44). By minimizing the error only in subsequent time steps we do not minimize the error globally in time. This is a consequence of keeping the grid points fixed once they have been determined, which forces the grid to be continuous in time. As the results will illustrate, this drawback is amplified by the presence of a variable bottom geometry.

An elegant solution would be to optimize the grid per time step at both the previous and the next time level, and to allow the grid to be discontinuous in time. This would strongly augment the possibilities for error minimization. Such a procedure is quite feasible in the present approach where we use a two-level method for the discretization in time; the coordinate transformation per time step would then be truly independent [cf. (5.9)], i.e., also the number of grid points per time step could be optimized. Moreover, solving (5.45), minimizing the error per time step with respect to $\overline{x}^{n-1}(\xi)$ as well, would become virtually equivalent with solving (5.44).

From (5.45) we obtain the algebraic system of equations that defines the optimal value of the x_i^n. Because Err^{space} and Err^{time} are grid dependent, the equations are nonlinear and have to be solved iteratively. Each iteration would involve a complicated interpolation procedure to handle the changing grid, which is not convenient. We have therefore opted for a different approach, determining per time step the optimal coordinate transformation $\overline{x}(\xi', \tau')$ given the coordinate transformation $\overline{x}(\xi, \tau)$. By solving the flow equations on the grid defined by $\overline{x}(\xi, \tau)$ we obtain the flow solution and hence Err^{space} and Err^{time} on that grid. From this we calculate $\overline{x}(\xi', \tau')$, which then defines the next approximation of optimal coordinate transformation $\overline{x}(\xi, \tau)$ on which we solve again the flow equations, etc. This process, which defines the *outer grid iteration loop,* is repeated until convergence, i.e., until $\overline{x}(\xi', \tau')$ and $\overline{x}(\xi, \tau)$ are virtually identical. That is, in principle, because the present algorithm is not yet able to

get the grid fully converged under all circumstances (see the next section). With each outer grid iteration we solve the flow equations given the grid defined by $\overline{x}(\xi, \tau)$, and solve the grid equations to obtain $\overline{x}(\xi', \tau')$ given the flow solution. Both are iterative solution procedures; the former has been described in Section 5.4.2, while the latter is the *inner grid iteration loop* described below.

For clarity, we will explain the grid optimization procedure for a single unknown a. Approximating the integral in time by means of a one-point quadrature rule, the optimization problem to be solved per time step is:

$$\min_{\overline{x}^n(\xi')} \Delta t \int_{\xi'_{\frac{1}{2}}}^{\xi'_{I+\frac{1}{2}}} \overline{x}_{\xi'}^{n-\frac{1}{2}} \left(\Delta \xi'^2 |\overline{a}_{\xi'\xi'} - \frac{\overline{x}_{\xi'\xi'}}{\overline{x}_{\xi'}} \overline{a}_{\xi'}| + c_\gamma \Delta \tau' |\overline{a}_{\tau'}| \right)^{n-\frac{1}{2}} d\xi' \ , \qquad (5.46)$$

where we have used an expression like (5.40) for a, substituting (5.35) and (5.37). For the optimization of the grid in shallow-water applications, the more complicated error expression (5.41) is used, but the principle remains the same.

Optimization problem (5.46) has been formulated in the computational space (ξ', τ') corresponding with optimal coordinate transformation $\overline{x}(\xi', \tau')$. To be able to solve it, we transform it to the current computational space (ξ, τ), introducing $\xi'(\xi, \tau)$ and $\tau' = \tau$, the transformation from current to optimal computational space. It is most convenient to consider $\xi'(\xi, \tau)$ and not $\overline{x}(\xi', \tau')$ as the unknown function to be determined, since $\xi'(\xi, \tau)$ is defined (and hence can be approximated) on the current computational grid. The coordinates of the optimal computational space corresponding with the grid points in the current computational space will be denoted by $\xi_i'^{,n} = \xi'(\xi_i, \tau^n)$. Notice that $\xi_i'^{,n} - \xi_{i-1}'^{,n}$ is *not* a grid size and hence in general not equal to $\Delta\xi'$, the size of the uniform grid in the optimal computational space.

At this point we can take the size of the grid in computational space (any computational space) equal to any convenient value. We will use $\Delta\xi = \Delta\xi' = 1$ [recall that we have $\Delta\tau = \Delta\tau' = \Delta t$ because of (5.9)]. The position of the boundaries of a computational domain are fixed, $\xi'_{\frac{1}{2}} = \xi_{\frac{1}{2}} = \frac{1}{2}$ and $\xi'_{I+\frac{1}{2}} = \xi_{I+\frac{1}{2}} = I + \frac{1}{2}$, and so optimization problem (5.46) formulated in (ξ, τ) becomes:

$$\text{solve:} \quad \min_{\xi'^{,n}(\xi)} \Delta t \int_{\xi_{\frac{1}{2}}}^{\xi_{I+\frac{1}{2}}} \overline{x}_{\xi}^{n-\frac{1}{2}} \left(\frac{|D_\xi(\overline{a})|}{(\xi'_\xi)^2} + c_\gamma \Delta\tau |\overline{a}_\tau - \frac{\xi'_\tau}{\xi'_\xi} \overline{a}_\xi| \right)^{n-\frac{1}{2}} d\xi \ . \qquad (5.47)$$

In order to be able to solve this problem we assume that $\overline{a}$ is a function of the physical coordinates only, and independent of the grid size. This is obviously not true in general, especially in regions of steep gradients, but a reasonable approximation for small grid perturbations (in particular perturbations around the optimal grid) because of the smoothness of the solution. Since we have fixed momentarily coordinate transformation $x(\xi, \tau)$, this implies that $\overline{x}$ and $\overline{a}$ in (5.47) are both considered to be independent of ξ'.

Under this assumption it is straightforward to solve (5.47). The integral is approximated by a sum over the grid cells using straightforward discretizations, and

differentiated with respect to the $\xi_i'^{,n}$, $i = 1, \ldots, I$. Equating the result to zero, the system of equations is obtained that defines the optimal grid:

$$\frac{8\Delta x_{i+\frac{1}{2}}^{n-\frac{1}{2}}}{(\xi_{i+1}'^{,n} - \xi_i'^{,n} + 1)^3} e_{i+\frac{1}{2}}^{\xi,n-\frac{1}{2}} - \frac{8\Delta x_{i-\frac{1}{2}}^{n-\frac{1}{2}}}{(\xi_i'^{,n} - \xi_{i-1}'^{,n} + 1)^3} e_{i-\frac{1}{2}}^{\xi,n-\frac{1}{2}}$$
$$= c_\gamma \Delta\tau \left(\frac{2\Delta x_{i+\frac{1}{2}}^{n-\frac{1}{2}} (\xi_{i+1}'^{,n} - i)}{(\xi_{i+1}'^{,n} - \xi_i'^{,n} + 1)^2} e_{i+\frac{1}{2}}^{\tau,n-\frac{1}{2}} - \frac{2\Delta x_{i-\frac{1}{2}}^{n-\frac{1}{2}} (\xi_{i-1}'^{,n} - i)}{(\xi_i'^{,n} - \xi_{i-1}'^{,n} + 1)^2} e_{i-\frac{1}{2}}^{\tau,n-\frac{1}{2}} \right),$$
$$i = 1, \ldots, I \,, \tag{5.48}$$

with:

$$e^\xi = \text{smoothed } |\overline{a}_{\xi\xi} - \overline{a}_\xi \overline{x}_{\xi\xi} / \overline{x}_\xi| \,,$$
$$e^\tau = \text{smoothed } \overline{a}_\xi \tanh\left(\frac{\Delta\tau (\overline{a}_\tau - \overline{a}_\xi \xi_\tau' / \xi_\xi')}{c_\tau |\overline{a}|} \right) . \tag{5.49}$$

The latter is an approximation of $e^\tau = \text{sign}(\overline{a}_\tau - \overline{a}_\xi \xi_\tau' / \xi_\xi')$ that, not surprisingly, turns out to lead to serious stability problems. The argument of the tanh function is scaled with $|\overline{a}|$ to make it non-dimensional. For the shallow-water equations we scale with the energy-head-like expression $\overline{d} + \frac{1}{2}\overline{q}^2/(g\overline{d}^2)$ and use $c_\tau = 10$ for the scaling parameter. The smoothing of e^ξ and e^τ is the same as the one applied for ν_{art}, i.e., like Equation (5.14) whose discretization is given in (5.18), using the same value of constant smoothing coefficient α.

Equation (5.48) with boundary conditions $\frac{1}{2}(\xi_0'^{,n} + \xi_1'^{,n}) = \frac{1}{2}$ and $\frac{1}{2}(\xi_I'^{,n} + \xi_{I+1}'^{,n}) = I + \frac{1}{2}$ is solved iteratively by means of a Newton-type method, evaluating e^τ explicitly using a very strong underrelaxation. The diagonal of the implicit part of the linearized system of equations per iteration is increased for additional stability. The approach is similar to that used for the flow equations (see Section 5.4.2). Although we obtain convergence down to 10^{-10}, the convergence is usually slow and certainly needs to be improved.

Once Equation (5.48) has been solved, we use the grid point values $\xi_i'^{,n}$ to define the piecewise linear function $\overline{\xi}'^{,n}(\xi)$. The coordinates of the next approximation of the optimal grid are determined by $x_i'^{,n} = \overline{x}^n(\xi)$, with ξ the solution of $\overline{\xi}'^{,n}(\xi) = i, i = 1, \ldots, I$. In practice, we do not use $\overline{x}^n(\xi)$ here but a smooth approximation based on a monotonicity-preserving cubic Hermite interpolation [6]. This leads to smoother grid updates in the outer grid iteration loop without changing the final, fully converged grid. The coordinates of the virtual grid points are obtained from the grid boundary conditions $\frac{1}{2}(x_0'^{,n} + x_1'^{,n}) = x_{\text{left}}$ and $\frac{1}{2}(x_I'^{,n} + x_{I+1}'^{,n}) = x_{\text{right}}$.

Once the outer grid iteration loop has converged we have $\xi_i'^{,n} \to \xi_i = i$ in which

case Equation (5.48) reduces to:

$$\begin{aligned}&\Delta x_{i+\frac{1}{2}}^{n-\frac{1}{2}} e_{i+\frac{1}{2}}^{\xi,n-\frac{1}{2}} - \Delta x_{i-\frac{1}{2}}^{n-\frac{1}{2}} e_{i-\frac{1}{2}}^{\xi,n-\frac{1}{2}} \\ &\quad = \frac{1}{2} c_\gamma \Delta\tau \left(\Delta x_{i+\frac{1}{2}}^{n-\frac{1}{2}} e_{i+\frac{1}{2}}^{\tau,n-\frac{1}{2}} + \Delta x_{i-\frac{1}{2}}^{n-\frac{1}{2}} e_{i-\frac{1}{2}}^{\tau,n-\frac{1}{2}} \right) \ , \ i = 1, \ldots, I \ ,\end{aligned} \tag{5.50}$$

or, in the equivalent continuous form:

$$\frac{\partial}{\partial \xi} \left(e^{\xi} \frac{\partial x}{\partial \xi} \right) = c_\gamma \Delta\tau e^{\tau} \frac{\partial x}{\partial \xi} \ . \tag{5.51}$$

In the left-hand side one recognizes an equidistribution principle based on the numerical modeling error in space. It is corrected with the term in the right-hand side that takes account of the effect that the movement of the grid has on the numerical modeling error in time.

For steady-state problems the grid optimization boils down to the equidistribution of the second-order interpolation error $|\overline{a}_{\xi\xi} - \overline{a}_\xi \overline{x}_{\xi\xi} / \overline{x}_\xi| = |(\overline{x}_\xi)^2 \overline{a}_{xx}|$. Equation (5.48) solved inside the outer grid iteration loop simplifies in that case to:

$$\frac{\Delta x_{i+\frac{1}{2}}^{n}}{(\xi_{i+1}^{\prime,n} - \xi_i^{\prime,n})^3} e_{i+\frac{1}{2}}^{\xi,n} - \frac{\Delta x_{i-\frac{1}{2}}^{n}}{(\xi_i^{\prime,n} - \xi_{i-1}^{\prime,n})^3} e_{i-\frac{1}{2}}^{\xi,n} = 0 \ , \ i = 1, \ldots, I \ , \tag{5.52}$$

with n some steady-state level.

5.7 Results

5.7.1 Steady-State Application

We consider a steady-state shallow-water problem for which an exact solution is available. As explained in Section 5.3, exact steady-state solutions of (5.1) and (5.2) can be constructed easily if the dissipation terms (bottom friction and artificial viscosity) are set to zero. This has been realized by setting $WC^2 = 10^{18}$. Artificial viscosity coefficient ν_{art} is set to zero to obtain the exact solution but plays of course an important role in the computations.

Figure 5.4 shows the geometry that we have considered, the exact solution of the water level, and the numerical solution of water level, velocity, Froude number, and energy head on a uniform grid consisting of 100 finite volumes. It is a fully converged solution (see Section 5.4.2) and has been obtained by setting $\alpha = 3$ and $c_\nu = 10$ in (5.14). With these parameter values, steep variations in ν_{art} and the numerical solution are spread over about 5 to 6 grid cells. It can be shown by means of a Fourier-mode accuracy analysis of the compatible scheme that this is sufficient and approximately required to ensure that higher-order error terms are indeed negligible. Notice however that the geometry is *not* smooth.

The boundary conditions that we have applied are $q = 19.8656\ \mathrm{m}^2/\mathrm{sec}$ at the left inflow boundary ($\rightarrow$ water level at left boundary is $h = 12.0000$ m) and $h = 9.0000$ m at the right outflow boundary ($\rightarrow$ position of the hydraulic jump is $x = 439.2647$ m). Because of the critical flow condition on top of the bar, the flow upstream of the bar (and hence the water level at the inflow boundary) is independent of the flow downstream of the bar and of the water level imposed at the outflow boundary.

Small dashes in Figure 5.4 indicate the level $\mathrm{Fr} = 1$. It can be observed that the position of the hydraulic jump is predicted within a fraction of the grid size. This confirms that, although smoothing does affect the solution locally, global details like the position of the jump are modeled very accurately (cf. the theoretical considerations in Section 5.3).

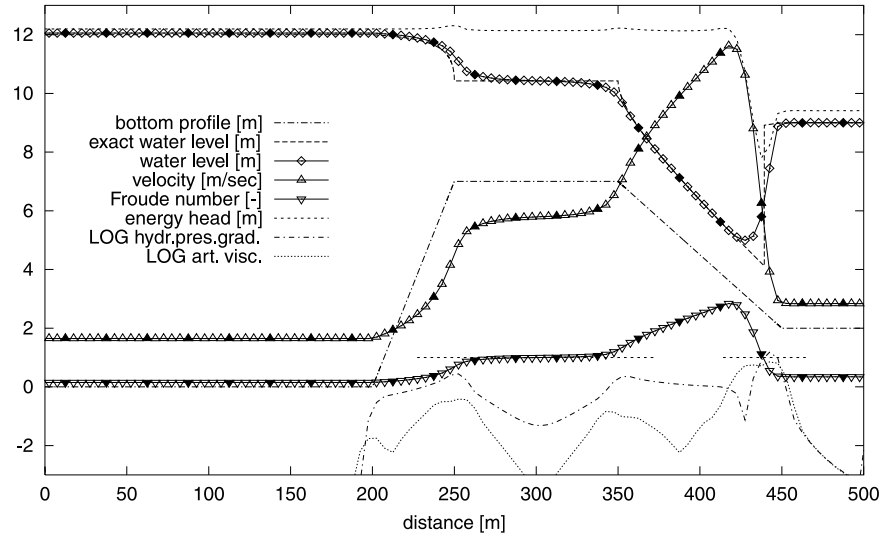


FIGURE 5.4
Shallow-water solution on uniform grid, 100 finite volumes.

Since the bottom friction has been set to zero and a physical viscosity model has not been included yet, the size of the artificial viscosity term cannot be compared with a physical dissipation term. Instead, we compare it in Figure 5.4 with the hydrostatic pressure term. The comparison shows, not surprisingly, that the artificial viscosity is fairly large in the neighborhood of the edges of the bar and equally important as the hydrostatic pressure gradient at the location of the hydraulic jump. The dominant character of the artificial viscosity around the hydraulic jump is explained by the fact that near discontinuities this term must balance the other terms in momentum equation (5.2). The fairly large value near the edges of the bar is mainly due to the lack of geometry smoothness.

Less artificial viscosity would have been required near the edges if geometry smoothing would have been applied. At first sight it may seem a matter of taste to smooth the solution near the edges indirectly by means of data smoothing rather

than directly by means of the artificial viscosity mechanism, but this is not true. This is because of the way the numerical solution method attempts to approximate the infinitely steep gradient at the top two edges (see the exact water level in Figure 5.4). The artificial viscosity takes care of these difficult details, but without additional geometry smoothing this mechanism is unable to provide enough smoothing, especially on a very fine grid when the details are felt strongly. The net result is that on very fine grids small wiggles tend to become important, thereby violating the assumption underlying the whole method that higher-order errors should be negligible. See also the adaptive grid results below.

The effect of the artificial viscosity on the solution is evidenced by the difference between the exact water level and its numerical approximation (see Figure 5.4). In practical applications, an exact solution would not be available. A comparison between the artificial viscosity and other modeling terms is however always possible. This information becomes especially useful once bottom friction and a turbulent viscosity model will be present, since it indicates immediately if deviations from measurement data are to be attributed to physical or numerical modeling errors. See also Section 5.2.

The energy head in Figure 5.4 downstream and upstream of the hydraulic jump is nearly constant, in agreement with the exact solution (see Section 5.3). Also the energy undershoot of a viscous hydraulic jump is predicted by the model. Not shown is the mass conservation "error". Because continuity equation (5.1) is discretized over the finite volumes $[\xi_{i-\frac{1}{2}}, \xi_{i+\frac{1}{2}}]$, the mass flow at the cell centers is constant within $O(10^{-12})$ which is the remaining convergence error. Perfect mass conservation at the discrete level does not prevent, however, a difference of 0.0352% between the value of q at the even-numbered grid points and the value of q at the odd-numbered grid points. This small mass flow wiggle is mainly due to the nonsmoothness of the geometry.

Solving the steady-state grid adaptation equation (5.52) and the flow equations alternately in 50 outer grid iterations gives the fully converged adaptive grid result shown in Figure 5.5. The outer grid iteration loop starts with a maximum relative grid correction $\max_i |\xi_i' - \xi_i| = \max_i |\xi_i' - i|$ of 23.5 in the first iteration, and reduces it to 4.72×10^{-6} in the last iteration. With such small grid corrections, there is of course no longer a gain in numerical accuracy. Comparing the exact solution and numerical solution in L_1 norm, it appears that the grid can be considered fully converged when $\max_i |\xi_i' - i| < 0.1$. This grid convergence criterion is reached after 13 outer grid iterations. The numerical solution error is then within 1% of its value on the fully converged grid.

A comparison between Figure 5.4 and Figure 5.5 clearly shows the significant accuracy improvement. The energy head upstream and downstream of the hydraulic jump is virtually constant, there is hardly any difference visible between the exact solution of the water level and its numerical approximation, and the level of the artificial viscosity is very low except at the locations where the solution is not smooth. However, the mass conservation error is 0.0156% which is barely a factor 2 better than on the uniform grid.

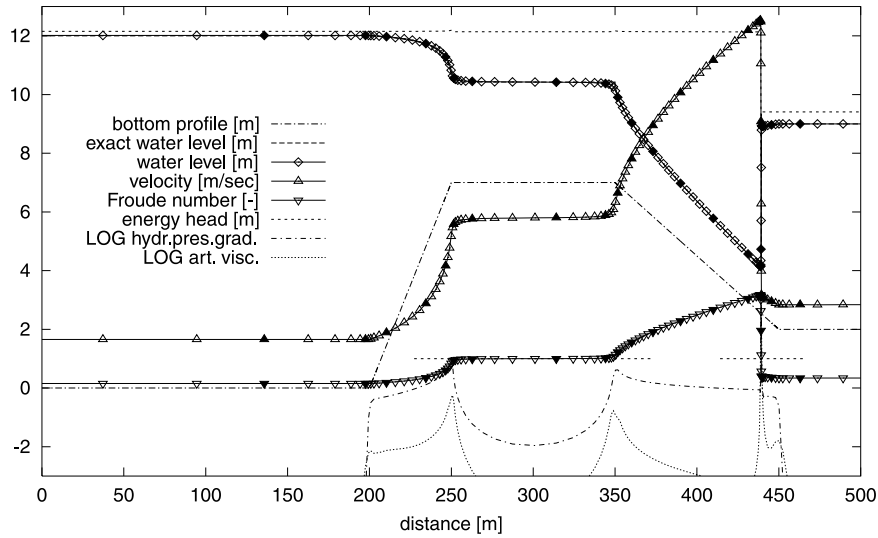


FIGURE 5.5
Shallow-water solution on adapted grid, 100 finite volumes.

A detail of the adaptive grid solution is shown in Figure 5.6 together with the uniform grid solution. It can be observed that the structure of the numerical approximation of the hydraulic jump on the adapted grid is identical to the one shown in Figure 5.4. The jump is again spread over about 5 to 6 grid cells while the levels of the undershoot of the energy head are the same, indicating that the artificial viscosity mechanism has been dimensioned correctly.

Parameter value $\alpha = 3$ has also been used for the smoothing of the error Err^{space} [see (5.41)] that has been minimized. With this value, the grid stretching is limited to 78.9% or $|\ln(\Delta x_{i+\frac{1}{2}}/\Delta x_{i-\frac{1}{2}})| \leq 0.582$. This result is obtained from discretization (5.18) of smoothing equation (5.14) that limits the rate of decay of the smoothed interpolation error to a factor $(\frac{3}{4}+2\alpha+2\sqrt{\frac{1}{8}+\alpha})/(2\alpha-\frac{1}{4})$ per grid cell. Substituting $\alpha = 3$ in this factor gives 1.789 ($\ln 1.789 = 0.582$), which is also the maximum grid stretching since the grid size is inversely proportional to the smoothed error [the equidistribution principle, Equation (5.52)].

The grid size and grid stretching as a function of computational space coordinate ξ are shown in Figure 5.7, with the grid size non-dimensionalized by the size of the uniform grid $\Delta x_{\text{unif}} = 5$ m. One recognizes the moderate grid refinement near the four edges of the bar (a stronger refinement near the top two edges) and the large refinement near the hydraulic jump.

The maximum stretching is indeed reached at a number of places. However, the maximum stretching is not reached at the boundaries where we would have expected it. The solution near the boundaries is uniform, the interpolation error becomes zero, and so the decay of the smoothed interpolation error and hence the grid stretching

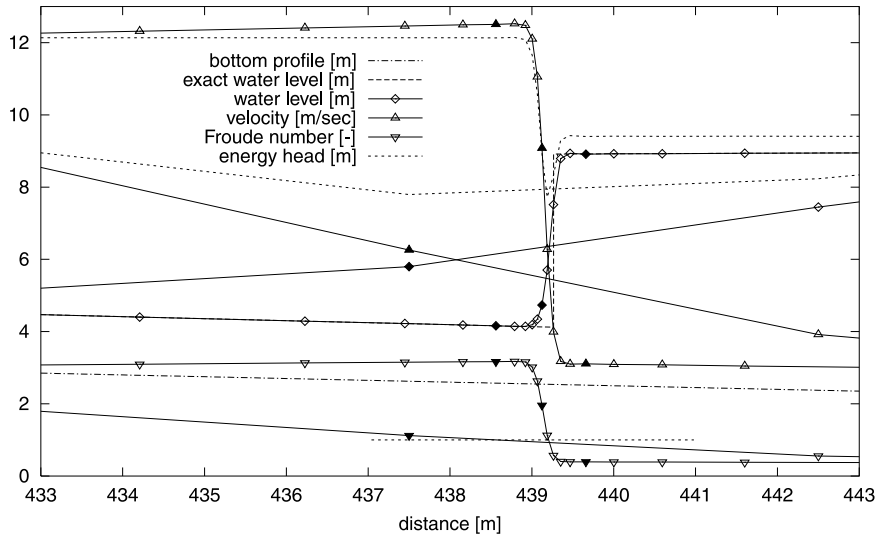


FIGURE 5.6
Comparison between uniform grid result and adaptive grid result at position of hydraulic jump, 100 finite volumes.

should have been maximal. They are not, which means that the interpolation error does not become zero near the boundaries. This can only be due to a small but noticeable wiggle, indicating the presence of non-negligible higher-order errors that are most likely caused by the non-smooth edges of the geometry. The same wiggle is also responsible for the comparatively large mass conservation error.

Table 5.1 summarizes the results of a number of adaptive grid experiments varying the number of finite volumes. We used the somewhat severe grid convergence criterion $\max_i |\xi_i' - i| < 0.1$. The number of outer grid iterations required to reach that criterion is given in the second column of the table. Convergence was not obtained for 280 and 400 volumes; the grid convergence error did not get lower than $O(1)$ in these calculations. We also needed to apply underrelaxation for the grid updates in the outer grid iteration loop when more than 100 volumes were used in order to stabilize the solution algorithm. These convergence problems are clearly related to the small but nevertheless important wiggles near the boundaries, as is evidenced by the maximum grid size $\Delta x_{\max}$. One would expect $\Delta x_{\max}$ to be roughly independent of the number of finite volumes, in accordance with the fact that the grid in the inlet section near the left boundary can have maximum stretching because of the very smooth solution. The solution is however less smooth than expected and so quite a number of grid points are drawn to the inlet section. This explains the large reduction of $\Delta x_{\max}$, the moderate gain in accuracy and the convergence problems when the number of volumes is larger than 100.

To confirm the hypothesis that these problems are essentially caused by the non-smooth geometry, we did the same series of tests with a geometry where the edges

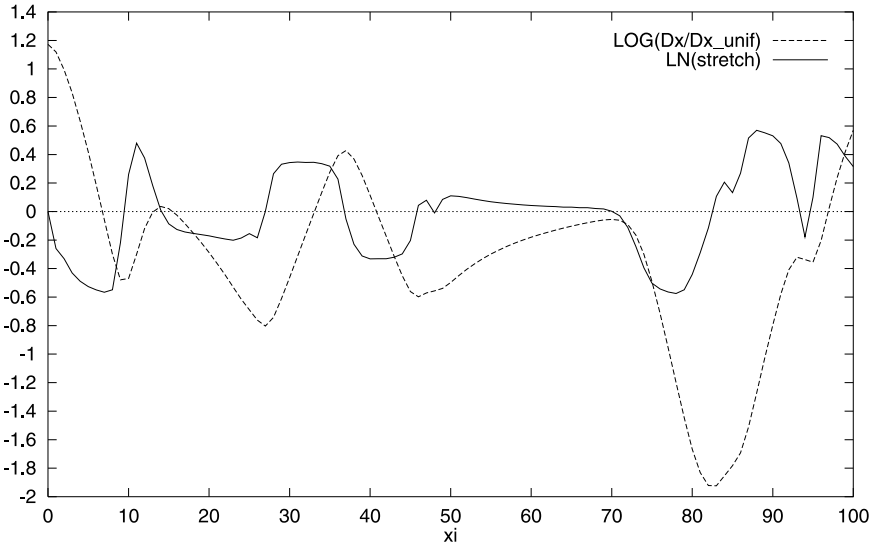


FIGURE 5.7
Computed optimal coordinate transformation for steady-state shallow-water calculation, 100 finite volumes.

Table 5.1 Convergence Behavior of Adaptive Grid Shallow-Water Calculations

# Vols	# Iters	$\Delta x_{\min}$	$\Delta x_{\max}$	$\|\overline{h} - h\|_1$	Order	CPU (sec.)
25	8	5.79E0	109.4	2.59E-1		0.38
35	9	2.22E0	100.9	1.81E-1	1.06	0.44
50	11	7.29E-1	92.7	9.61E-2	1.78	0.77
70	11	2.32E-1	78.4	4.07E-2	2.55	1.10
100	13	5.96E-2	74.4	1.34E-2	3.31	1.48
140	25	1.81E-2	29.7	5.15E-3	2.68	3.90
200	30	6.71E-3	11.3	2.27E-3	2.30	7.09
280	50	3.44E-3	7.3	1.01E-3	2.40	24.22
400	50	2.49E-3	3.8	5.43E-4	1.74	40.15

were rounded over a length of 10 m. This is, however, only a partial solution to the problem since pointwise geometry approximation (5.15) is not able to "see" the curved shape of the smoothed edges in between the grid points. As a consequence, each time the grid is adapted and the points move to a different position, the flow solver sees a slightly different geometry and hence calculates a slightly different solution. This is obviously not favorable to the grid convergence process and explains why the results were only marginally better.

A series of tests with $\alpha = 6$, in an attempt to compensate for the non-smooth geometry by increasing the artificial viscosity smoothing, was more successful. This

time we also obtained grid convergence when 280 volumes were used, and lower errors for 200 finite volumes and more, despite the fact that the grid was less refined near the hydraulic jump (the location of minimum grid size $\Delta x_{\min}$). From this we conclude that the adaptive grid modeling of the discontinuous jump is not posing a problem. Since the hydraulic jump is at a position where the geometry is smooth, this is consistent with our interpretation. The solution error was, however, larger when less than 200 volumes were used, because with $\alpha = 6$ the grid stretching is limited to 51%.

The last column in Table 5.1 gives the CPU time of the calculations that were all executed on a 400-MHz Pentium notebook computer. To assess the efficiency of the adaptive grid method they should be compared with the CPU time of the same calculation on a uniform grid. Steady-state shallow-water calculations on a uniform grid are, however, very fast, mainly because of the effort that we have spent in optimizing the iterative flow solver (see Section 5.4.2). For example, a fully converged steady-state calculation for the same problem on a uniform grid of 400 finite volumes takes only 0.82 CPU sec. The L_1 error in the water level, $\|\overline{h} - h\|_1$, turns out to be 2.06E-2 for this calculation. This result is more accurate and obtained faster than the result of the adaptive grid calculation with 70 finite volumes (see Table 5.1). On the other hand, the adaptive grid convergence criterion that we applied was rather severe. There is also still room for improvement of the adaptive grid solver (see, e.g., Section 5.4.2). We especially expect a significant gain in accuracy and efficiency once geometry smoothing is implemented.

5.7.2 Unsteady Application

We present an unsteady adaptive grid application that, strictly speaking, is not even a genuine unsteady application. It is a fairly simple and yet extremely complicated test for moving adaptive grid methods, and clearly shows both the advantages and shortcomings of our moving adaptive grid approach.

The test is simple: starting from the steady-state solution shown in Figure 5.4, calculate the steady-state solution on the adapted grid of Figure 5.5 by solving the flow equations both in time and space using a time step of 1 sec. Although a steady-state problem in physical space, it is *not* a steady-state problem in computational space where we solve (5.11) and (5.12).

We will first solve this problem in the standard way, applying the equidistribution principle in space and ignoring the effect that this has on the error in computational time. This is realized by solving per outer grid iteration Equation (5.48) with $c_\gamma = 0$. As can be seen in Figure 5.8, the effect is dramatic. Virtually all grid points are drawn toward the hydraulic jump region in the very first time step. As a consequence, many grid points move to a position with a totally different water depth and velocity. This leads to very large variations in computational time and, since time derivatives and space derivatives are coupled, also to very large perturbations of the solution in space. This applies in particular to grid points at and in the vicinity of the bar.

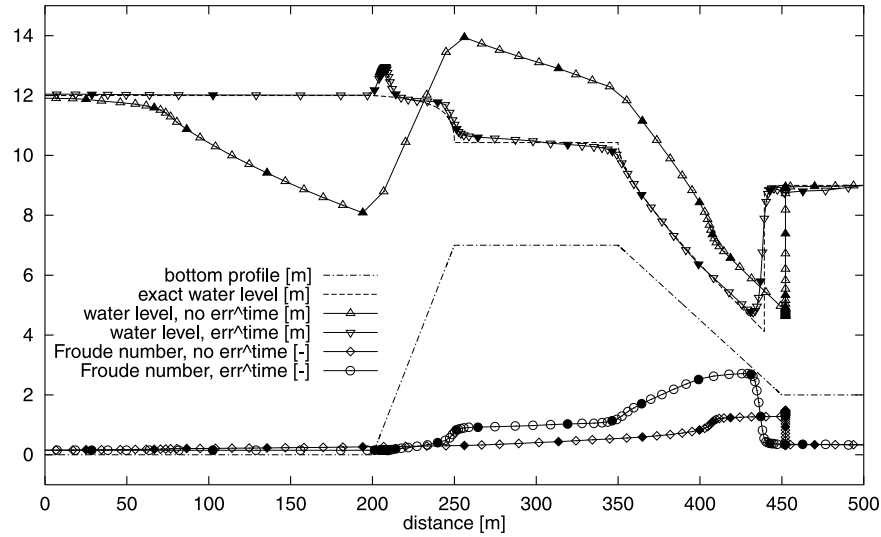


FIGURE 5.8
Unsteady shallow-water solution at the first time level on an adapted grid with and without compensation of time discretization error, 100 finite volumes.

The reason for this erroneous behavior is the non-uniform bottom profile in combination with the fact that the adaptive grid algorithm did not take into account the effect of the grid point redistribution on the error in time. Still, even a uniform bottom may be difficult to handle if the time discretization error is not taken into account. Similar solution perturbations as shown here (although probably not that strong) may occur for example if a sudden change at one of the boundaries forces the grid points to move through the domain in the direction of that boundary, crossing a region where due to the presence of waves there may be a significant variation in water depth and velocity (cf. Section 5.6).

The erroneous motion of grid points can be suppressed by smoothing the grid velocity [5, 30, 31]. However, this slows down the entire grid adaptation process and can be expected to have in general a negative effect on the quality of the solution [22]. This can be compensated for by using a small time step but then the advantages of grid adaptation become unclear. A calculation on a uniform grid using a large number of points may actually be more efficient and more accurate.

A moving adaptive grid algorithm that minimizes the combined space-time numerical modeling error should perform better. The right-hand side of (5.48) should be able to detect large errors in time and avoid excessive grid point displacements whenever this would lead to large errors in time. On the other hand, large grid adaptations in regions where it has little effect on the accuracy in time are still possible.

The result obtained in the first time step using (5.48) with $c_\gamma = 6$ (a typical value, see Section 5.5.3) and with $c_\tau = 10$ in (5.49) is shown in Figure 5.8 as well. The result is much better, but still not satisfactory. One notices the large concentration of

grid points at the base of the bar which is due to the fact that the grid points are again drawn toward the hydraulic jump. The difference with the previous result without time error compensation is that this time the error minimization in time prevents the grid points from crossing the bar.

The overshoot in the water level at the base is probably due to the non-smoothness of the geometry. The results obtained in subsequent time steps show that this overshoot initiates a small wave moving essentially upstream and decaying rapidly. However, grid points start following that wave and are then not available to improve the accuracy elsewhere. We have observed that the solution error $\|\overline{h} - h\|_1$ did decrease from the very first time step, although very slowly (less than a factor 2 in 20 time steps) due to the perturbations introduced in the first time step.

No full convergence has been obtained for the adapted grids of this section, although we did not spend much effort in finding suitable values for the different convergence parameters. We found that rather useless because of the fundamental shortcoming already mentioned in Section 5.6: the lack of adaptive flexibility when the grid is continuous in time. The results presented here reveal that this is a major drawback in shallow-water applications where grid points may have to move continuously across non-uniform parts of a channel geometry in order to get a high resolution in certain dynamically changing regions. This will always lead to relatively large errors in time and will always require some compensation mechanism. In our method the compensation is included automatically, based on the error in time and only active when necessary. Nevertheless, this compensation (and any such compensation mechanism) will inevitably limit the grid speed and hence the potential gain in accuracy. There may then be no advantage in using a moving adaptive grid technique.

The solution to this problem has already been suggested in Section 5.6: consider a grid that is discontinuous in time and minimize the error per time step (5.45) also with respect to $\overline{x}^{n-1}(\xi)$. The derivation of the optimal grid equations is straightforward and leads to an interesting result that can be summarized as follows. Average grid size $x_i^{n-\frac{1}{2}} - x_{i-1}^{n-\frac{1}{2}}$ will be optimized for minimal error in space, while grid displacement $x_i^n - x_i^{n-1}$ will be optimized for minimal error in time. This virtually independent minimization of the numerical modeling error in space and time indicates that a discontinuously moving adaptive grid method may prove to be very efficient.

5.8 Conclusions

We have presented the development of a moving adaptive grid method that aims at minimizing the numerical modeling error (the part of the numerical solution error generated locally as a result of solving the model equations numerically), rather than the numerical solution error. Besides being virtually impossible to realize for problems of practical interest, the usefulness of the latter is limited because of the presence of physical modeling errors and data errors whose effect should also be taken into account.

The idea behind the present approach is first to ensure that physics-based artificial smoothing terms form the dominant numerical modeling error to allow a direct comparison with physical modeling terms; and second to minimize the effect of the artificial smoothing terms by means of grid adaptation. We have shown that this is feasible in 1-D, although the presented numerical algorithm is still incomplete. One element that needs to be added is geometry smoothing, the importance of which is illustrated by the results.

Smoothness is the key element of the proposed method; it ensures that the effect of discretization errors on the numerical solution is small, which is essential for a meaningful error analysis. The use of a compatible scheme is required to be able to analyze that effect, and to obtain a useful approximation of the numerical modeling error in space and in time as a function of the local grid parameters. The combination of artificial smoothing and a compatible discretization has enabled us to develop an error-minimizing moving adaptive grid algorithm, minimizing the effect of both discretization errors and smoothing errors.

The results clearly show the importance of taking the error in time into account when adapting the grid in space. However, grid adaptation can be very inefficient if the grid is forced to move continuously. This applies in particular to the unsteady shallow-water applications with non-uniform bottom in which we are interested. An extension that needs to be considered is therefore the development of a discontinuously moving adaptive grid method.

The complexity of the developed method is large. On the other hand, a high gain in accuracy is possible since the method is capable of using grid points very efficiently. For unsteady calculations this will only be true after the method has been extended with a grid that can move discontinuously in time.

There are indications that the 1-D error analysis of the compatible scheme that we have presented can be extended to several space dimensions, showing the correspondence between the multi-D numerical modeling error and multi-D interpolation errors. Multi-D interpolation errors depend however in a complicated way on the grid size, grid stretching, grid curvature, and grid skewness. Minimizing these errors by means of grid adaptation will be very difficult to realize. On the other hand, elements of such a technique may possibly be combined with a more heuristic adaptive grid approach to arrive at better monitor functions and hence more efficient grid adaptation procedures for multi-D applications of practical interest.

References

[1] M. Arora and P.L. Roe, On postshock oscillations due to shock capturing schemes in unsteady flows, *J. Comput. Phys.,* **130**, (1997), 25–40.

[2] M. Borsboom, Development of an error-minimizing adaptive grid method, *Appl. Num. Math.,* **26**, (1998), 13–21.

[3] M.H. Chaudhry, *Open-Channel Flow,* Prentice-Hall, 1993.

[4] V.T. Chow, *Open-Channel Hydraulics,* McGraw-Hill, 1973.

[5] E.A. Dorfi and L.O'C. Drury, Simple adaptive grids for 1-D initial value problems, *J. Comput. Phys.,* **69**, (1987), 175–195.

[6] R.L. Dougherty, A. Edelman, and J.M. Hyman, Nonnegativity-, monotonicity-, or convexity-preserving cubic and quintic Hermite interpolation, *Math. Comp.,* **52**, (1989), 471–494.

[7] K. Eriksson, D. Estep, P. Hansbo, and C. Johnson, Introduction to adaptive methods for differential equations, *Acta Numerica,* **4**, (1995), 105–158.

[8] K. Eriksson and C. Johnson, Adaptive streamline diffusion finite element methods for stationary convection-diffusion problems, *Math. Comp.,* **60**, (1993), 167–188.

[9] R.M. Furzeland, J.G. Verwer, and P.A. Zegeling, A numerical study of three moving-grid methods for one-dimensional partial differential equations which are based on the method of lines, *J. Comput. Phys.,* **89**, (1990), 349–388.

[10] P. Garcia-Navarro, F. Alcrudo, and J.M. Saviron, 1-D open-channel flow simulation using TVD-McCormack scheme, *J. Hydr. Engrg.,* **118**, (1992), 1359–1372.

[11] T. Geßner, D. Kröner, B. Schupp, and M. Wierse, Finite volume methods for conservation laws and convection-dominated diffusion equations, in: F. Benkhaldoun and R. Vilsmeier, eds., *Finite Volumes for Complex Applications — Problems and Perspectives,* Hermes, Paris, 1996, 61–76.

[12] D.F. Hawken, J.J. Gottlieb, and J.S. Hansen, Review of some adaptive node-movement techniques in finite-element and finite-difference solutions of partial differential equations, *J. Comput. Phys.,* **95**, (1991), 254–302.

[13] C. Hirsch, *Numerical Computation of Internal and External Flows, Volume 2,* Wiley, 1990.

[14] P. Houston, J.A. Mackenzie, E. Süli, and G. Warnecke, A posteriori error analysis for numerical approximations of Friedrichs systems, *Numer. Math.,* **82**, (1999), 433–470.

[15] P. Houston, R. Rannacher, and E. Süli, A posteriori error analysis for stabilised finite element approximations of transport problems, *Tech. Rep. NA-99/04,* Oxford University Computing Laboratory, 1999.

[16] W. Huang and R.D. Russell, Analysis of moving mesh partial differential equations with spatial smoothing, *SIAM J. Numer. Anal.,* **34**, (1997), 1106–1126.

[17] C. Johnson, R. Rannacher, and M. Boman, Numerics and hydrodynamic stability: toward error control in computational fluid dynamics, *SIAM J. Numer. Anal.,* **32**, (1995), 1058–1079.

[18] A.A. Khan and P.M. Steffler, Physically based hydraulic jump model for depth-averaged computations, *J. Hydr. Engrg.,* **122**, (1996), 540–548.

[19] R.J. LeVeque and H.C. Yee, A study of numerical methods for hyperbolic conservation laws with stiff source terms, *J. Comput. Phys.,* **86**, (1990), 187–210.

[20] J.A. Mackenzie, The efficient generation of simple two-dimensional adaptive grids, *SIAM J. Sci. Comput.,* **19**, (1998), 1340–1365.

[21] E.A. Meselhe, F. Sotiropoulos, and F.M. Holly Jr., Numerical simulation of transcritical flow in open channels, *J. Hydr. Engrg.,* **123**, (1997), 774–783.

[22] Y. Qiu and D.M. Sloan, Numerical solution of Fisher's equation using a moving mesh method, *J. Comput. Phys.,* **146**, (1998), 726–746.

[23] T. Sonar and E. Süli, A dual graph-norm refinement indicator for finite volume approximations of the Euler equations, *Numer. Math.,* **78**, (1998), 619–658.

[24] B. van Leer and W.A. Mulder, Relaxation methods for hyperbolic equations, in: F. Angrand, A. Dervieux, J.A. Desideri, and R. Glowinsky, eds., *Numerical Methods for the Euler Equations of Fluid Dynamics,* SIAM, 1985, 312–333.

[25] J.G. Verwer, J.G. Blom, R.M. Furzeland, and P.A. Zegeling, A moving grid method for one-dimensional PDEs based on the method of lines, in: J.E. Flaherty, P.J. Paslow, M.S. Shephard, and J.D. Vasilakis, eds., *Adaptive Methods for Partial Differential Equations,* SIAM, 1989, 160–175.

[26] J. von Neumann and R.D. Richtmyer, A method for the numerical calculation of hydrodynamic shocks, *J. Applied Physics,* **21**, (1950), 232–237.

[27] G.P. Warren, W.K. Anderson, J.L. Thomas, and S.L. Krist, Grid convergence for adaptive methods, in: M.J. Baines and K.W. Morton, eds., *Numerical Methods for Fluid Dynamics 4,* Oxford University Press, 1993, 317–328.

[28] F.M. White, *Viscous Fluid Flow,* McGraw-Hill, 1974.

[29] Y. Xiang, N.R. Thomson, and J.F. Sykes, Fitting a groundwater contaminant transport model by L_1 and L_2 parameter estimators, *Adv. Water Res.,* **15**, (1992), 303–310.

[30] P.A. Zegeling, Moving-grid methods for time-dependent partial differential equations, *CWI-tract No. 94,* Centre for Math. and Comp. Science, Amsterdam, 1993.

[31] P. Zegeling, M. Borsboom, and J. van Kester, Adaptive moving grid solutions of a shallow-water transport model with steep vertical gradients, in: V.N. Burganos, ed., *Proc. 12th Int. Conf. on Comput. Methods in Water Resources, Vol. 2,* Computational Mechanics Publications, 1998, 427–434.

Chapter 6

An Adaptive Method of Lines Approach for Modeling Flow and Transport in Rivers

Gerd Steinebach and Peter Rentrop

6.1 Introduction

Mathematical modeling has become a very important tool in many areas of modern hydrology during the last two decades. In [7] an overview of the application of mathematical models in hydrology is given.

In this chapter, we concentrate on three examples: water-level forecast systems, accidental pollution models, and the computation of flood planes. Examples one and two are based on one space-dimensional partial differential equations (1D PDEs), whereas the last example needs a 2D modeling approach.

Figure 6.1 shows the River Rhine catchment area of 185.000 km^2. In summer, the overall flow behavior in the River Rhine is dominated by the alpine regions due to snow melt. In winter, the peak flows are caused by the main tributaries Neckar, Main, and Moselle.

In the first example, a water-level forecast model for the Middle Rhine downstream from the River Main junction to the German-Dutch border at the Lower Rhine will be considered. The aim of the development of the WAter-level FOrecast System (WAFOS) has been to provide navigation during loading in the North Sea harbor Rotterdam with relevant water-level information. With this information the skipper should be enabled to load his ship to a maximum level and with corresponding security with respect to loading bottlenecks during low water periods [9]. Since 1996, WAFOS is applied daily for this purpose in the Federal Institute of Hydrology (Bundesanstalt für Gewässerkunde, BfG).

On the other hand, during floods early warning systems are very important for the mitigation of possible damages. One basic part of such early warning systems is an operational water-level forecast model. Since 1998 WAFOS has been in use for this purpose as well. The forecast results are disseminated by the flood warning center in Mainz, which is operated by the Federal State Rhineland-Palatinate and the Federal Waterway and Navigation Administration [33].

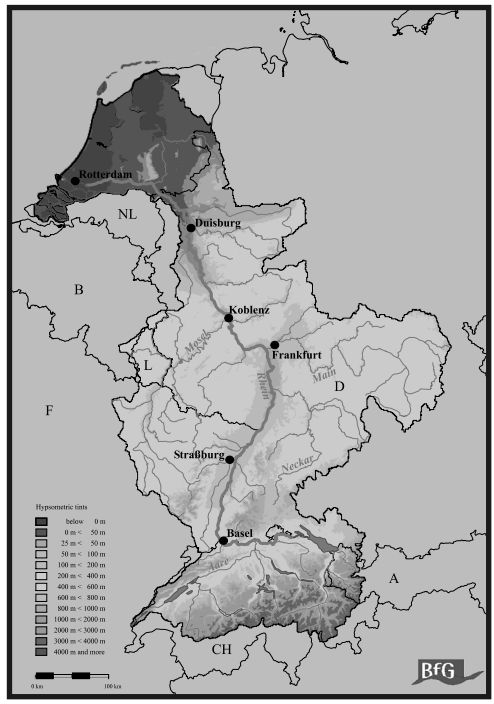


FIGURE 6.1
The River Rhine catchment area.

The model is built up by three sub-models [31]:

1. a hydrodynamic model for the most important river reaches
2. a statistical model for the forecasting of important input gauges of the hydrodynamic model
3. a precipitation-runoff model for relevant sub-catchments

The precipitation-runoff components are presently under development, but are not yet in use operationally [16]. Through a statistical Wiener multi-channel filtering approach [40] the water levels of some important input gauges for the hydrodynamic model are forecast, in order to enlarge the overall forecast time. The hydrodynamic model is based on the hyperbolic Saint–Venant equations. They are described in detail below.

In the second example, so-called alarm models will be considered. In 1986, a disastrous fire at the Sandoz chemical factory near Basel caused a very severe pollution of the River Rhine up to several hundred kilometers downstream. After this accident, the development of an alarm model for the quick and reliable prediction of the transport and dispersion of hazardous substances was demanded [29]. Such alarm models have

also been established in other river basins. In this chapter, the model of the River Elbe is described. It is based on a dead-zone approach, consisting of a parabolic convection-diffusion equation coupled by an additional linear equation, which models the exchange of mass concentration between the main stream and the dead zones [13]. In Figure 6.2 the influence of the different processes transport, diffusion, linear decay, and dead zones is sketched. The related equations are presented in the next section.

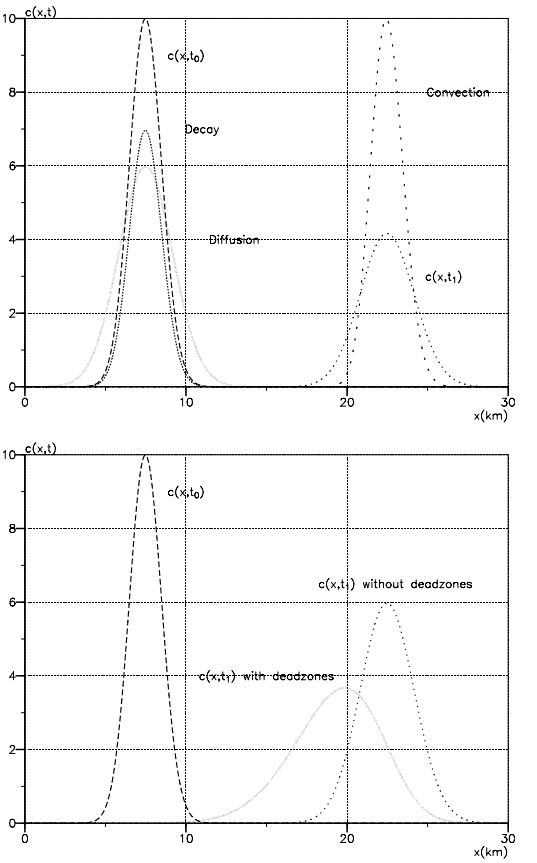


FIGURE 6.2
Influence of different processes on the concentration ($c(x, t_0)$ describes the initial condition and $c(x, t_1)$ is the resulting concentration at time $t_1 > t_0$), (a) convection-diffusion model, (b) dead-zone model.

The final example requires a two space-dimensional modeling. The task is to compute the area, which will be flooded in consequence of a specific flow discharge. Usually, the flow discharge is related to a return period, e.g., a 100-year flood. The flooded area is then shown on a map, in order to derive the flood risk for a building or

a street. In Figure 6.3 such an example is shown for a small reach of the lower River Saar, a tributary of the Moselle.

FIGURE 6.3
Flooded area in the lower River Saar.

6.2 Modeling Flow and Transport in Rivers

River flow modeling in two space dimensions is usually based on the 2D shallow-water equations (2D SWEs). They can be derived from the Navier–Stokes equations by assuming a hydrostatic pressure law. The 2D SWEs in conservative variables read [39]

$$q_t + e(q)_x + g(q)_y = s(q)\ . \tag{6.1}$$

$q = \begin{pmatrix} h \\ uh \\ vh \end{pmatrix}$ is the vector of states with water depth $h(t, x, y)$ and depth averaged velocities $u(t, x, y)$, $v(t, x, y)$ in x (resp. y) direction.

The flux vectors are given by $e(q) = \begin{pmatrix} uh \\ u^2h + \frac{1}{2}gh^2 \\ uvh \end{pmatrix}$ and $g(q) = \begin{pmatrix} vh \\ uvh \\ v^2h + \frac{1}{2}gh^2 \end{pmatrix}$.

The source term $s(q) = \begin{pmatrix} 0 \\ gh\,(S_{0x} - S_{fx}) \\ gh\,(S_{0y} - S_{fy}) \end{pmatrix}$ accounts for bottom friction and bottom slope.

The expression S_{fx} (resp. S_{fy}) is called friction slope. An empirical formula (by Manning–Strickler) reads

$$S_{fx} = \frac{1}{K_S^2 h^{\frac{4}{3}}} \cdot u \cdot \sqrt{u^2 + v^2}, \quad S_{fy} = \frac{1}{K_S^2 h^{\frac{4}{3}}} \cdot v \cdot \sqrt{u^2 + v^2}, \quad K_S \in \mathbb{R}^+ .$$

The constant K_S depends on the soil condition of the riverbed and is called roughness coefficient. It is assumed that the friction slope implicitly accounts for the main effects of turbulence as well.

The bottom slope is given by $S_{0i} = -\partial_i b (i = x, y)$ with the bottom elevation $b(x, y)$. In Figure 6.4 the basic notations are shown.

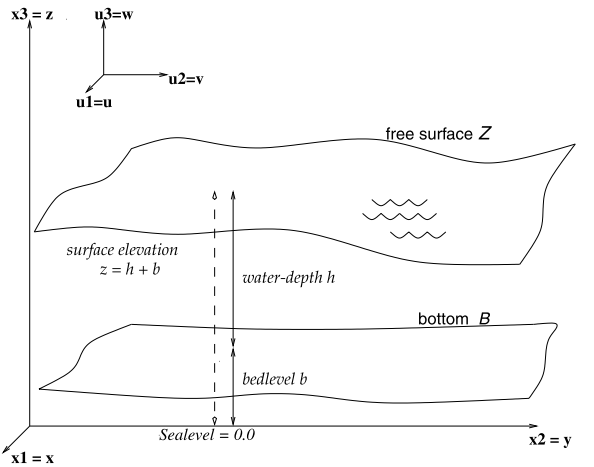


FIGURE 6.4
Basic notations.

The Saint–Venant's equations are the shallow-water equations in one space dimension x directed along the river course. The water surface elevation in this case is given by $z(t, x) = b(x) + h(t, x)$, where $b(x)$ is the bottom level, $S_0 := -b_x$ is the bottom slope, and $h(t, x)$ is the water depth.

For arbitrary cross-sections $A(t, x)$ the equations read [34]:

$$\text{Conservation of mass:} \qquad A_t + (uA)_x = 0 \tag{6.2}$$

$$\text{Conservation of momentum:} \qquad (uA)_t + \left(u^2 A\right)_x = -gA\left(z_x + S_f\right) \tag{6.3}$$

The friction slope is given by

$$S_f = \frac{1}{K_S^2 R^{\frac{4}{3}}} \cdot |u| \cdot u \tag{6.4}$$

where K_S is Strickler's roughness coefficient and the hydraulic radius R is the ratio of the cross-section area and the wetted perimeter. R can be well approximated by $R \approx h$ for wide river cross-sections.

Supplied with data of the cross-section profiles in the form of paired values (z, A), the water-depth $h(t, x)$ can be determined at each location x as a function of the cross-sectional area $A(t, x)$ in some way: $h(t, x) = H(x, A(t, x))$, e.g., by linear interpolation. Then

$$z_x = b_x + \frac{\partial H}{\partial x} + \frac{\partial H}{\partial A}\, A_x$$

and Equation (6.3) may be written as

$$(uA)_t + \left(u^2 A\right)_x + gA\frac{\partial H}{\partial A}\, A_x = -gA\frac{\partial H}{\partial x} + gA\left(S_0 - S_f\right) . \tag{6.5}$$

Equations (6.2) and (6.5) with the variable $Q = uA$ lead to the hyperbolic system

$$\begin{pmatrix} A \\ Q \end{pmatrix}_t + \begin{pmatrix} 0 & 1 \\ gA\frac{\partial H}{\partial A} - \frac{Q}{A}^2 & 2\frac{Q}{A} \end{pmatrix} \begin{pmatrix} A \\ Q \end{pmatrix}_x = \begin{pmatrix} 0 \\ gA(S_0 - S_f - \frac{\partial H}{\partial x}) \end{pmatrix} \tag{6.6}$$

whose eigenvalues are $\lambda_{1,2} = \frac{Q}{A} \pm c$. ($c = \sqrt{gA\frac{\partial H}{\partial A}}$ is the wave speed.)

If both eigenvalues have the same sign, the flow is called supercritical, otherwise it is subcritical (cf. supersonic, subsonic for the Euler equations of gas flow).

A rectangular prismatic channel of constant width B with

$$A(t, x) = B \cdot h(t, x) \quad \Rightarrow \frac{\partial H}{\partial A} = \frac{1}{B} = \text{const.}$$

defines a special case of the Saint–Venant equations:

$$\partial_t \begin{bmatrix} h \\ uh \end{bmatrix} + \partial_x \begin{bmatrix} uh \\ u^2 h + \frac{1}{2} g h^2 \end{bmatrix} = \begin{bmatrix} 0 \\ gh(S_0 - S_f) \end{bmatrix} . \tag{6.7}$$

The transport of soluble substances in natural rivers is usually described in 1D through a simple convection-diffusion-reaction approach:

$$c_t = -uc_x + D_L c_{xx} - K_A c\ , \tag{6.8}$$

with concentration $c(x, t)$, flow velocity u, dispersion coefficient D_L, and linear decay rate K_A.

A more realistic model is obtained if dead zones are taken into account where very low flow velocities occur. Because of concentration exchange with these dead zones, the concentration curves do not remain symmetric in nature. These quasi-2D effects can be modeled by a linear exchange term in the equation for the concentration c and by adding a second equation for the concentration in the dead zone s [13, 25]:

$$c_t = -uc_x + D_L c_{xx} - \frac{A_0}{A} K(c - s) - K_A c\ , \tag{6.9}$$

$$s_t = K(c - s) - K_A s \tag{6.10}$$

with concentration $c(x,t)$ in the main river, concentration $s(x,t)$ in the dead zone, area ratio dead zone/main river $\frac{A_0}{A}$, and exchange rate K.

All equations have to be completed by appropriate boundary and initial conditions.

6.3 Method of Lines Approach

6.3.1 Network Approach

In technical simulation of time-dependent processes, today's industrial software is often based on a network approach. This approach is embedded in the technical computer-aided design (TCAD) environment and allows automatic generation of the mathematical models [12]. In river simulation problems, a similar approach can be applied. The network elements are 1D or 2D models for certain river reaches, coupling elements like weirs, or junctions with tributaries and boundary elements like gauging stations [24]. The single river reaches are modeled by the partial differential equations presented in the upper section. All other elements are represented by algebraic equations. By using the method of lines approach (MOL) for the partial differential equations, a large system of differential algebraic equations (DAEs) is generated [27, 25]. For its time integration, standard DAE-software can be applied.

This process is demonstrated via the following one-dimensional example. All 1D equations may be summarized in

$$q_t = f\,(t, x, q, q_x)\ , \tag{6.11}$$

defined on the time interval $t_0 \le t \le t_1$ and space interval $\alpha \le x \le \beta$. $q = (q_1, \ldots, q_n), q_i := q_i(x,t), i = 1, \ldots, n$ are the unknowns with $q_t = (\frac{\partial q_1}{\partial t}, \ldots, \frac{\partial q_n}{\partial t})^t$, $q_x = (\frac{\partial q_1}{\partial x}, \ldots, \frac{\partial q_n}{\partial x})$ and $f = (f_1, \ldots, f_n)^t$.

The following investigations are based on a semidiscretization in space by finite differences but they are transferable to finite volume discretizations.

Let $q^i(t) = q(X_i, t)$ be defined on the equidistant space-mesh $X_i = \alpha + i\,\Delta x,\ i = 0, \ldots, N$ with meshsize $\Delta x = \frac{\alpha-\beta}{N}$.

Approximation of

$$q_x(X_i, t) = \frac{q^{i+1}(t) - q^{i-1}(t)}{2\Delta x}$$

and substitution into (6.11) yields a linear implicit ODE-system of dimension $n(N+1)$:

$$\begin{aligned} \tilde{A}\frac{dy}{dt} &= \tilde{f}(t, X_0, \ldots, X_N, y)\ , \quad \text{with} \\ y &= \left(q_1^0, \ldots, q_n^0, \ldots, q_1^N, \ldots, q_n^N\right), \quad \tilde{A} = \operatorname{diag}\left(\tilde{A}_0, I, \ldots, I, \tilde{A}_N\right)\ . \end{aligned} \tag{6.12}$$

$\tilde{A}_0$ and $\tilde{A}_N$ depend on the prescribed boundary conditions. In case of the maximum number of $2n$ boundary conditions, one has $\tilde{A}_0 = \tilde{A}_N = 0$. If no boundary condition

has to be satisfied, then $\tilde{A}_0 = \tilde{A}_N = I$, I is the identity matrix. In this case central finite differences must be replaced by forward differences in α and backward differences in β. The function $\tilde{f}$ depends on f, the space discretization, and the boundary conditions. The initial conditions at time t_0 must be consistent with the boundary conditions in α and β [25].

The convergence of the semidiscretized system (6.12) to the exact solution of (6.11) for $N \to \infty$ is presumed.

6.3.2 Space Discretization

To prevent the numerical solution from spurious oscillations, the space discretization must be adapted to the hyberbolic character of the flow equations. Moreover, in a 2D model, complex river geometries must be handled properly. This is possible through conservative finite volume (FV) schemes.

We start with the initial boundary value problem

$$\begin{aligned} q_t + f(q)_x &= s(q)\,, && (6.13)\\ q(0,x) &= q_0(x)\,, && (6.14)\\ q(t,\alpha) &= q_\alpha(t)\,, q(t,\beta) = q_\beta(t)\,, && (6.15) \end{aligned}$$

where the space-interval $[\alpha, \beta]$ is partitioned in N cells $I_1, \ldots, I_N$ through a given set of $N+1$ mesh-points by $\alpha = x_{\frac{1}{2}} < \cdots < x_{N+\frac{1}{2}} = \beta$.

Integration of (6.13) over the control volume $I_j = [x_{j-\frac{1}{2}}, x_{j+\frac{1}{2}}]$ with length $\Delta x_j = x_{j+\frac{1}{2}} - x_{j-\frac{1}{2}}$ yields

$$\begin{aligned} \int_{x_{j-\frac{1}{2}}}^{x_{j+\frac{1}{2}}} \partial_t q(t,x)\,dx &= -\left[f\left(q\left(t, x_{j+\frac{1}{2}}\right)\right) - f\left(q\left(t, x_{j-\frac{1}{2}}\right)\right)\right] \\ &\quad + \int_{x_{j-\frac{1}{2}}}^{x_{j+\frac{1}{2}}} s(q(t,x))\,dx\,. \end{aligned} \tag{6.16}$$

Defining the cell-average

$$Q_j(t) = \frac{1}{\Delta x_j} \int_{x_{j-\frac{1}{2}}}^{x_{j+\frac{1}{2}}} q(t,x)\,dx \tag{6.17}$$

on the middle points (cell centers)

$$x_j = \frac{1}{2}\left(x_{j-\frac{1}{2}} + x_{j+\frac{1}{2}}\right), \quad j = 1, \ldots, N \tag{6.18}$$

Equation (6.16) can be written as

$$\begin{aligned} Q_j'(t) &= -\frac{1}{\Delta x_j}\left[f\left(q\left(t, x_{j+\frac{1}{2}}\right)\right) - f\left(q\left(t, x_{j-\frac{1}{2}}\right)\right)\right] \\ &\quad + \frac{1}{\Delta x_j}\int_{x_{j-\frac{1}{2}}}^{x_{j+\frac{1}{2}}} s(q(t,x))\,dx\,. \end{aligned}$$

The approximation of the right hand side for $j = 1, \ldots, N$ leads to a system of ordinary differential equations (ODEs):

$$Q'_j(t) = \left(-\frac{1}{\Delta x_j} \left[f^*_{j+\frac{1}{2}} - f^*_{j-\frac{1}{2}} \right] + s_j \right) (Q(\cdot; t)) \tag{6.19}$$

The right-hand side usually depends not only on one state Q_j but also on the states of some neighboring cells.

The computation of $f^*_{j+\frac{1}{2}}$, $f^*_{j-\frac{1}{2}}$ requires the solution of local Riemann problems. This is carried out through the flux-difference splitting approach described in [1, 2]. This scheme was derived for the homogeneous flow Equations (6.7) for rectangular cross-sections in one space dimension and (6.1) in two space dimensions. A modification is given in [19, 32], in order to properly take into account the source terms friction and bottom slope.

A suitable discretization of the dead-zone equations (6.9) and (6.10) was presented in [13]. To avoid negative concentrations an ENO scheme was proposed for the discretization of the convective term $-uc_x$ and a standard central finite difference discretization for the diffusion term $D_L c_{xx}$. This ENO approach can be applied to the 1D Saint–Venant Equations (6.6) for arbitrary cross-sections, too. It should be mentioned that the 2nd order ENO and 2nd order flux-difference scheme with minmod limiter [35] lead to nearly identical numerical results, if applied to Equation (6.7).

The idea of this ENO discretization is the reconstruction of the numerical flux function $f^*_{j+\frac{1}{2}}$ by a primitive function. The primitive function can be approximated by polynominal interpolation. This interpolation is calculated via divided differences and the set of points (stencil) included in the interpolation is chosen in order to get a smooth polynominal.

6.3.3 Time Integration

The most attractive feature in MOL applications is the possibility to use high quality and sophisticated integration schemes of high order for the semidiscretized equations. Since the system of differential equations or DAEs generated via the space-discretization process is usually stiff, implicit or semi-implicit time-integration schemes are preferable. Rosenbrock–Wanner (ROW) methods are known to be efficient for moderate accuracy requirements [38]. Due to their semi-implicit structure they allow large time-steps, which are appropriate for the simulation of slowly varying flow problems. A ROW-method with stage-number s for the numerical solution of a linear-implicit index 1 DAE system of the type

$$My' = f(t, y) \ , \ t \in [t_0, t_1] \tag{6.20}$$

with possible singular (n,n)-matrix M and given consistent initial values $y(t_0)$ is defined by

$$y_1 = y_0 + \sum_{i=1}^{s} b_i K_i \tag{6.21}$$

$$\left(M - h\gamma \frac{\partial f}{\partial y}(t_0, y_0)\right) K_i = hf\left(t_0 + \alpha_i h, y_0 + \sum_{j=1}^{i-1} \alpha_{ij} K_j\right) + \gamma_i h^2 \frac{\partial f}{\partial t}(t_0, y_0)$$

$$+ h \frac{\partial f}{\partial y}(t_0, y_0) \sum_{j=1}^{i-1} \gamma_{ij} K_j \tag{6.22}$$

with

$$\alpha_i = \sum_{j=1}^{i-1} \alpha_{ij}, \quad \gamma_i = \sum_{j=1}^{i} \gamma_{ij}, \quad \gamma_{ii} = \gamma \tag{6.23}$$

and the coefficients $\gamma, \alpha_{ij}, \gamma_{ij}, \quad i = 1, \ldots, s, \quad j = 1, \ldots, i-1$ and weights b_i, y_1 being the approximation to the solution at time $t + h$ with $y(t) = y_0$.

Unfortunately severe order-reductions can occur if Runge–Kutta or Rosenbrock methods are applied to semidiscretized PDEs [37, 20, 18]. This can be demonstrated via the following example.

Let

$$u(x, t) = x\phi(t) + (1 - x)\psi(t) \qquad \phi, \psi \in C^1[0, \infty) .$$

be the solution of the parabolic problem

$$u_t = u_{xx} + f \ , x \in [0, 1] \ , t \geq 0$$

with $f(x, t) = x\phi' + (1 - x)\psi'$ and $u(0, t) = \psi(t), \quad u(1, t) = \phi(t), u(x, 0) = x\phi(0) + (1 - x)\psi(0)$.

Semidiscretization on the equidistant mesh $X_i = \frac{i}{N+1}, i = 1, \ldots, N$ with central finite differences leads to

$$U' = AU + B(t) + G'(t) \tag{6.24}$$

with $U = (U_1, \ldots, U_N)^t$, $U_i = u(X_i, t)$, $G = (G_1, \ldots, G_N)^t$, $G_i(t) = X_i\phi(t) + (1 - X_i)\psi(t)$, $B(t) = (N + 1)^2(\psi(t), 0, \ldots, 0, \phi(t))^t$ and matrix A defined by

$$A = -(N + 1)^2 \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}$$

Equation (6.24) is equivalent to

$$U' = A(U - G) + G'$$

and diagonalization of $A = T^{-1}\Lambda T$ yields the decoupled system

$$Y' = \Lambda(Y - \tilde{G}) + \tilde{G}' \text{ with } Y = TU \text{ and } \tilde{G} = TG$$

of Prothero–Robinson type [23]:

$$y' = \lambda(y - g) + g', \quad g(t) \text{ smooth and } Re\lambda \leq \lambda_0 < 0\,. \tag{6.25}$$

This model has the exact solution $y(t) = g(t)$ for $y(0) = g(0)$ and for $y(0) \neq g(0)$ with stiffness $Re\lambda \ll 0$ the solution $y(t)$ attains $g(t)$ very quickly asymptotically.

It is well known that many methods, if applied to the Prothero–Robinson model, suffer from order-reduction. For the well-known ROW method RODAS [11], an order reduction from theoretical order 4 to 1 can be observed [30]. RODAS is an A-stable stiffly accurate embedded ROW method of order 4(3) with stage number $s = 6, \hat{s} = 5$. Scholz [28] derived additional order-conditions for ROW-methods to overcome the order-reduction, and Ostermann and Roche [21] could show that these additional conditions are sufficient to preserve the classical order of convergence on certain classes of semidiscretized linear parabolic PDEs. In [30] a new coefficient set for RODAS was derived in order to avoid order reduction phenomena in the context of MOL. These coefficients are given in Table 6.1.

Table 6.1 Set of Modified Coefficients for RODAS, with $\beta_{ij} = \alpha_{ij} + \gamma_{ij}$

$\gamma = 0.25$	$\alpha_{21} = 0.75$	$\beta_{21} = 0.0$
	$\alpha_{31} = 8.6120400814152190E-2$	$\beta_{31} = -0.049392$
$b_1 = \beta_{61}$	$\alpha_{32} = 0.1238795991858478$	$\beta_{32} = -0.014112$
$b_2 = \beta_{62}$	$\alpha_{41} = 0.7749345355073236$	$\beta_{41} = -0.4820494693877561$
$b_3 = \beta_{63}$	$\alpha_{42} = 0.1492651549508680$	$\beta_{42} = -0.1008795555555556$
$b_4 = \beta_{64}$	$\alpha_{43} = -0.2941996904581916$	$\beta_{43} = 0.9267290249433117$
$b_5 = \beta_{65}$	$\alpha_{51} = 5.308746682646142$	$\beta_{51} = -1.764437648774483$
$b_6 = \gamma$	$\alpha_{52} = 1.330892140037269$	$\beta_{52} = -0.4747565572063027$
	$\alpha_{53} = -5.374137811655562$	$\beta_{53} = 2.369691846915802$
$\hat{b}_1 = \beta_{51}$	$\alpha_{54} = -0.2655010110278497$	$\beta_{54} = 0.6195023590649829$
$\hat{b}_2 = \beta_{52}$	$\alpha_{61} = -1.764437648774483$	$\beta_{61} = -8.0368370789113464E-2$
$\hat{b}_3 = \beta_{53}$	$\alpha_{62} = -0.4747565572063027$	$\beta_{62} = -5.6490613592447572E-2$
$\hat{b}_4 = \beta_{54}$	$\alpha_{63} = 2.369691846915802$	$\beta_{63} = 0.4882856300427991$
$\hat{b}_5 = \gamma$	$\alpha_{64} = 0.6195023590649829$	$\beta_{64} = 0.5057162114816189$
	$\alpha_{65} = 0.25$	$\beta_{65} = -0.1071428571428569$

A disadvantage of ROW-methods, if applied to ODE systems $y' = f(t, y)$ of higher dimension, is the requirement of the exact Jacobian ($\frac{\partial f}{\partial y}$) of the right-hand side at every time-step. Therefore, the computation of the Jacobian and the solution of the linear equation systems are the main computational costs in case of integrating

systems of large dimension. The code RODAS of Hairer and Wanner was modified especially to take into account sparse non-banded Jacobians and to make efficient use of sparse linear algebra software. This altered code, together with the new coefficients, is referred to as RODASP [30].

The **computation of the Jacobian** is in most cases performed by finite differences. In case of a full (n,n)-matrix this can be done by (n + 1) function evaluations of the right-hand side f with suitably chosen `delta`:

```
dy1=f(t,y)
for j=1 to n do begin
  y(j)=y(j)+delta
  dy=f(t,y)
  for i=1 to n do begin
    Jac(i,j)=(dy(i)-dy1(i))/delta
  end
  y(j)=y(j)-delta
end
```

In case of banded Jacobians with bandwidth m, the above algorithm is usually modified so that it needs only m evaluations of f. The idea is to alter $\frac{n}{m}$ components of `y` at once before f is evaluated. These components must be chosen so that two or more nonzero entries in one row `i` never appear for two different components `j`. This idea can be exploited directly for the computation of sparse Jacobians if the sparsity structure is known [4]. Often it is easy to modify the subroutine defining the right-hand side in order to determine the sparsity structure of the Jacobian in a preprocessing step.

The linear algebra routines in RODASP allow the treatment of sparse matrices and the solution of the linear equations with the preconditioned BI-CGSTAB algorithm [36] or with the well-known method MA28 for the solution of unsymmetric sparse systems [6, 5].

Another extension of RODASP was necessary in order to use it in combination with the second-order, flux-difference splitting or ENO space-discretization schemes. Since the positions of the ENO interpolation stencil can change at every evaluation of the right-hand side $f(t, y)$, the numerically computed Jacobian may not reflect $(\frac{\partial f}{\partial y})$ consistently. Therefore, at the beginning of each time-step the stencil positions have to be chosen once and are fixed during all other right-hand side evaluations within this time-step.

This feature is demonstrated by the 1D idealized dam-break problem. A 2000-m long channel is assumed to be rectangular, horizontal, and frictionless. The reservoir water (with depth h^R) and the tail water (with depth h^T) are separated by a dam placed in the middle (x_M) of the channel.

The sudden and complete removal of the dam can be simulated by the homogeneous initial value problem (6.7) with

$$\begin{aligned} h(0,x) &= \begin{cases} h^R & \text{if } x < x_M \\ h^T & \text{if } x > x_M \end{cases} \\ u(0,x) &= 0\,. \end{aligned}$$

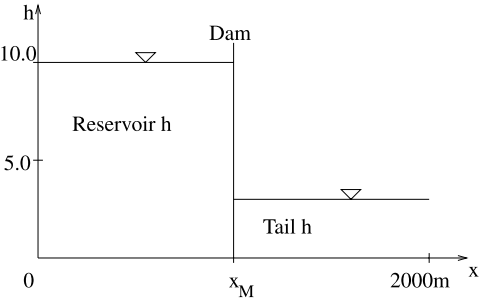


FIGURE 6.5
1D dam-break: setup.

By the theory of simple waves the exact depth $h(t,x)$ is found to be a piecewise smooth function on four varying intervals [34]:

$$h(t,x) = \begin{cases} h_1 = h^R & \text{if} \quad x \le -c_1 t \\ h_2 = \frac{1}{9g}\left(2c_1 - \frac{x - x_M}{t}\right)^2 & \text{if} \quad -c_1 t < x < (u_3 - c_3)t \\ h_3 & \text{if} \quad (u_3 - c_3)t \le x \le \dot{\xi} t \\ h_4 = h^T & \text{if} \quad x > \dot{\xi} t \end{cases}$$

where $c_i = \sqrt{g h_i}$ $(i = 1, \ldots, 4$ and $g = 9.81 \frac{m}{s^2})$.

Still a system of three nonlinear equations has to be solved for the unknowns u_3 (velocity behind the shock), c_3 (wave speed behind the shock), and $\dot{\xi}$ (shock speed of the flood wave). Introducing the abbreviation $\eta = \frac{\xi}{c_4}$ they read:

$$\text{conservation of mass:} \quad \frac{c_3}{c_4} = \sqrt{\frac{1}{2}\left(\sqrt{1 + 8\eta^2} - 1\right)} \tag{6.26}$$

$$\text{conservation of momentum:} \quad \frac{u_3}{c_4} = \eta - \frac{1}{4\eta}\left[1 + \sqrt{1 + 8\eta^2}\right] \tag{6.27}$$

$$u + 2c = \text{ const. in regions 2 and 3:} \quad \frac{u_3}{c_4} + 2\frac{c_3}{c_4} = 2\frac{c_1}{c_4} \tag{6.28}$$

After inserting (6.26) and (6.27) into (6.28) we realize that η, and therefore the whole solution, is implicitly dependent on the ratio of the initial depths, since $\frac{c1}{c4} = \sqrt{\frac{h^R}{h^T}}$.

Table 6.2 summarizes a comparison of the different methods. The simulations stop at $t_{\text{End}} = 60.0\ s$ and the water depth is compared to the analytical solution.

The abbreviations are

> Roe(2): Roe's second order flux-difference splitting scheme with minmod-limiter. It is well known [17] that monotone upwind centered scheme for conservation law (MUSCL) extrapolation with the minimod limiter is identical to a second-order ENO-extrapolation.

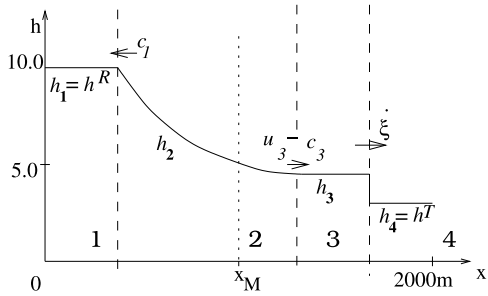


FIGURE 6.6
1D dam-break: progression.

Roe(2)*: Roe(2) with fixed stencils within each time step.

The calculations were performed on an HP-UX B.10.20 A 9000/780 with the relative and absolute error tolerances rtol $= 5 \times 10^{-4}$ for RODASP:

NFCN	: Number of right-hand side evaluations
NSTEP	: Number of computed time steps
NACCEPT	: Number of accepted time steps
	(equals NJAC, the number of Jacobian evaluations)

The L^2-error measures the deviation of the approximated depths h_j from the analytical solution $h(x)$ and is defined as $\sqrt{\frac{1}{N}\sum_{j=1}^{N}\left(h_j - h(x_j)\right)^2}$.

Table 6.2 1D Dam-Break Problem

Methods	Δx	NFCN	NSTEP	NACCEPT	CPU/sec	L^2-error
Roe(2)	20	1254	225	129	6.99	0.12568
	10	2187	390	237	22.98	0.07474
Roe(2)*	20	485	83	70	3.38	0.13015
	10	829	141	124	11.02	0.07845

The modification of the MUSCL-extrapolation scheme reduces NFCN drastically by increasing Δt and has nearly no influence on the accuracy.

It has to be mentioned that explicit schemes are preferable for those highly dynamic simulations with strongly varying solution.

6.4 Adaptive Space Mesh Strategies

In the classical MOL approach, a space mesh is chosen in the beginning and the resulting ODEs or DAEs are integrated in time. If during the integration steep moving

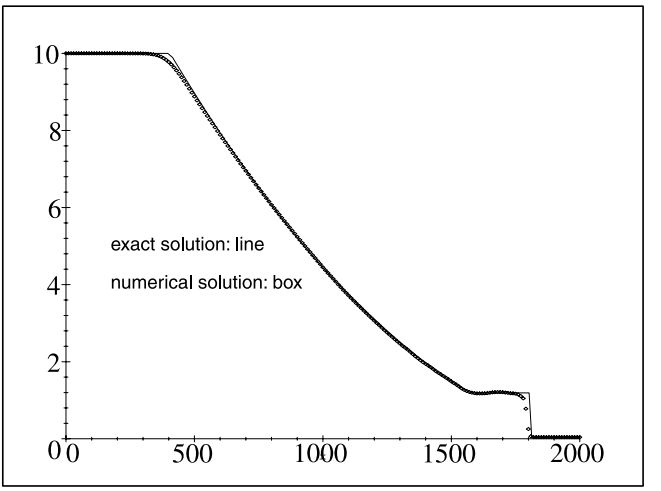


FIGURE 6.7
Dam-break problem with Roe(2)*, grid-size: $\Delta x = 10$.

fronts occur, a very fine mesh for the whole space interval $[\alpha, \beta]$ is necessary. In this case adaptive meshes would be preferable. The mesh is adapted during the integration in such a way that a fine resolution is obtained near the fronts and a coarse one in regions of smooth solution components.

One has to distinguish between static and dynamic remeshing [8, 14]. In dynamic remeshing the space-discretization points are considered to be time dependent and they move with the solution. The disadvantage is the introduction of new unknowns (the meshpoints) and the altered structure of the semidiscretized equations [22].

In static remeshing a new grid is fixed after one or m integration steps depending on the actual solution behavior. The solution has to be interpolated from the old mesh onto the new one and the integration procedure can be continued. Since every static remeshing step introduces a new system of equations with possibly different dimension, the application of one-step methods is preferable.

In the following example, the application of a static remeshing strategy is demonstrated. The advection dominated problem

$$c_t = -uc_x + Dc_{xx} \qquad t \geq 0\ ,\ x \in [\alpha, \beta]$$

with Peclet-number

$$P_e = \frac{u}{D}(\beta - \alpha) = 10^3$$

and typical constants $\alpha = 0, \beta = 30000, u = 1, D = 30$ is treated. The left boundary condition

$$c_\alpha(t) = 10 \exp\left(-0.001 \frac{(t - 7500)^2}{t}\right)$$

represents a wave coming from left into the computational domain $[\alpha, \beta]$. The problem is integrated until $t = 60.000$. At this time the wave has left the interval $[\alpha, \beta]$.

The equidistribution strategy of [15], which is also implemented in the SPRINT-software [3], has been used. By this strategy it is possible to construct a locally bounded mesh $\alpha = X_0 < \cdots < X_N = \beta$ with respect to a bound $K \geq 1$:

$$\frac{1}{K} \leq \frac{X_{i+1} - X_i}{X_i - X_{i-1}} \leq K \ .$$

Moreover, the mesh is sub-equidistant with respect to a mesh-function $m(x)$ and a constant $c > 0$ with

$$Nc \geq \int_a^b m(x)dx \ \text{ and } \ \int_{X_i}^{X_{i+1}} m(x)dx \leq c \quad \text{for } i = 0, \ldots, N-1 \ .$$

The mesh-function was chosen as

$$m(x) = \sqrt{\sigma + c_{xx}^2} \tag{6.29}$$

By the parameter σ, the maximum possible mesh-size can be controlled.

Figure 6.8 shows a typical adaptive grid during simulation. Finite differences have been applied in this example for space discretization. In [26] it has been shown that this type of problem can benefit from adaptive grid strategies. Fixing the CPU-time, the maximum error is halved; or describing an equal error, the number of grid points is smaller for the adaptive strategy. For larger Peclet-numbers the improvements are increasing.

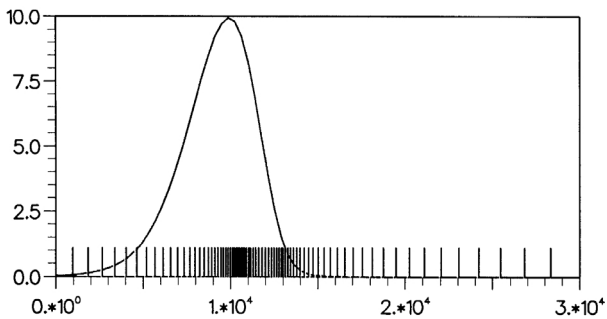


FIGURE 6.8
Adaptive grid.

6.4.1 Extension to 2D Problems

The equidistribution strategy can easily be extended to 2D river flow problems. In a first step, a basic mesh for the 2D domain must be constructed. Figure 6.9 shows such a coarse mesh for a stretch of the lower River Saar. This mesh has been constructed orthogonal to the main flow direction by a linear affine interpolation [39].

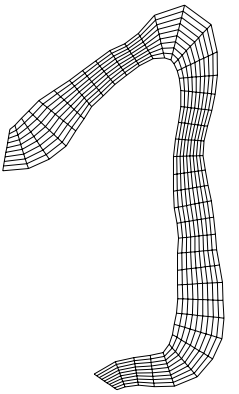


FIGURE 6.9
Basic 2D mesh.

In the next step, one preferred direction along the river course is chosen and parametrizations along and across this direction are defined (cf. Figure 6.10).

Parametrizations along the river

$$\begin{aligned} s &\longmapsto (x_m(s), y_m(s))^t, \quad s \in [0,1] \\ \Longrightarrow s &\longmapsto (x_l(s), y_l(s))^t, \quad s \longmapsto (x_r(s), y_r(s))^t \end{aligned}$$

and for all $s \in [0,1]$ across the river:

$$t \longmapsto (x(s,t), y(s,t))^t, \quad t \in [0,1]$$

$$\begin{pmatrix} x(s,t) \\ y(s,t) \end{pmatrix} = \begin{pmatrix} x_l(s) \\ y_l(s) \end{pmatrix} + t \begin{pmatrix} x_r(s) - x_l(s) \\ y_r(s) - y_l(s) \end{pmatrix}$$

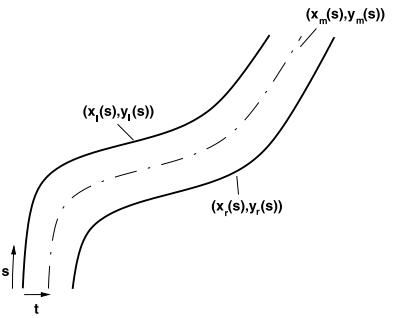


FIGURE 6.10
Parametrization of space coordinates.

Now, a mesh function $m(s)$ along the river course can be defined by

$$m(s) = \int_0^1 c(x(s,t), y(s,t))\, dt$$

with a certain function $c(x, y)$, e.g., $c(x, y) = \sqrt{u^2 + v^2}$ (norm of velocities).

This mesh function is equidistributed and new mesh points s_i, $i = 1, \ldots, n$ in s direction can be calculated.

In the next step, functions $m_i(t)$ are defined for all s_i, $i = 2, \ldots, n-1$:

$$m_i(t) = \int_{s_{i-1}}^{s_{i+1}} c(x(s,t), y(s,t))\, ds$$

The equidistribution of $m_i(t)$ leads to new mesh points t_{ij}, $j = 1, \ldots, m$ across the river.

If necessary, a more smooth behavior of the mesh points and less distorted angles can be obtained through the modified mesh functions

$$m_i(t) = \int_{s_{i-k}}^{s_{i+k}} c(x(s,t), y(s,t))\, ds, \quad k > 1$$

Figure 6.11 shows the new resulting mesh. This mesh has been adapted to the bottom elevation in t direction with the choice $c(x, y) = b_{\max} - b(x, y)$, $b_{\max}$ being the maximum bottom elevation in the corresponding cross-section. In Figure 6.11 the course of the river bed within the flood plane can be seen.

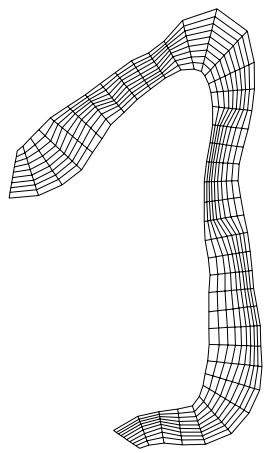


FIGURE 6.11
Adaptive mesh.

Finally, an interpolation of the solution from the old onto the new mesh has to be performed. Remember that in the finite volume approach the unknowns Q_j are mean values on the mesh cells Ω_j: $Q_j = \frac{1}{|\Omega_j|} \int_{\Omega_j} q\, dA$.

The interpolated solution can be calculated via

$$Q_{j_{\text{new}}} = \frac{1}{|\Omega_{j_{\text{new}}}|} \sum_i |\Omega_i \cap \Omega_{j_{\text{new}}}| Q_i$$

This approach has been tested for the Molenkamp problem (2D convection):

$$c_t = uc_x + vc_y, \quad x \in [-1, 1], \quad y \in [-1, 1]$$

with $u = -2\pi y$, $v = 2\pi x$.

An analytical solution is

$$c(x, y, t) = 0.01^{4r^2}, \quad \text{with} \quad r = \sqrt{(x + \frac{1}{2}\cos 2\pi t)^2 + \left(y + \frac{1}{2}\sin 2\pi t\right)^2}$$

Initial condition and boundary conditions are chosen according to this solution.

Table 6.3 compares the fixed and adaptive solutions after one revolution ($t_{\text{end}} = 1$) on different meshes, ϵ being the maximum absolute error. For the time integration in this non-stiff example the explicit scheme DOPRI5 [10] has been applied. After every 5 to 20 time steps a remeshing has been done. It can be concluded that the adaptive approach leads to an improved accuracy, but CPU time is increased extremly due to the 2D interpolation. For a final maximum absolute error ϵ, the computation times are nearly equivalent, see the results for $\epsilon = 0.4$ in row one and $\epsilon = 0.39$ in row two. Thus, this proposed adaptive method does not really pay off in 2D problems. Nevertheless, it is very useful for mesh construction. The initial adaptive 31×31 mesh is shown in Figure 6.12.

Table 6.3 Numerical Results for Molenkamp Problem

		31×31	41×41	51×51	61×61
Fixed	ϵ	0.62	0.47	0.40	0.29
	CPU	19	45	87	153
Adaptive	ϵ	0.39	0.25	0.19	
	CPU	101	233	488	

6.5 Applications

The aim of the model WAFOS is the forecasting of water levels at important gauges along the river during low water for navigation and during floods for flood warning. Usually, a daily 48-h forecast run is performed on the basis of measured water levels up to 7.00 AM. During floods, the model is operated up to three times a day. In this case the results up to a forecast time of 24 h are disseminated for 18 main gauges on River Rhine downstream of Karlsruhe/Maxau. Figure 6.13 shows an example of forecast results for gauging station Koblenz during a medium flood in 1999.

Since the gauging station Koblenz is located 800 m upstream of the junction with the Moselle, severe backwater effects occur. Therefore, a coupled hydrodynamic

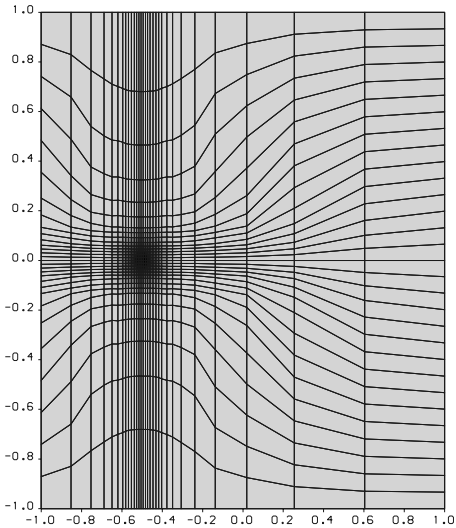


FIGURE 6.12
Initial adaptive 31 × 31 mesh for Molenkamp problem.

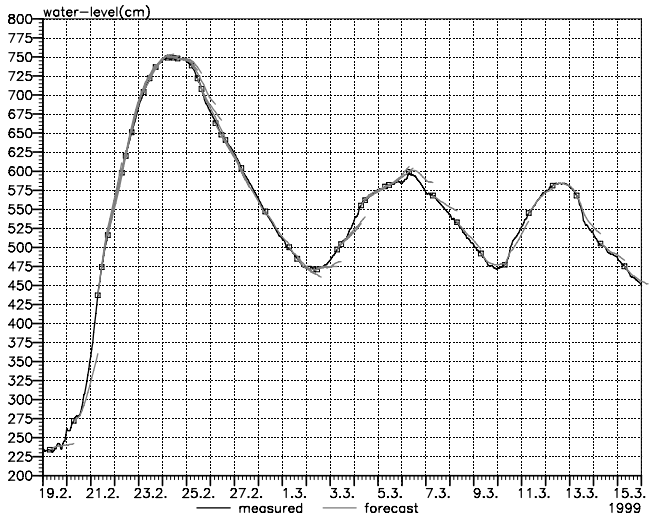


FIGURE 6.13
Forecast results for gauging station Koblenz/Rhine.

model of rivers Rhine and Moselle must be used. This model includes 700 km river reach. Space discretization results in a system of 7216 DAEs. A forecast run including simulation of the last 2 days requires less than a minute on a Pentium III, 500 MHz.

The alarm model for River Elbe includes an approximately 600-km river reach in Germany. ENO schemes are used for space discretization and RODASP for the solution in time. In order to calibrate the model, tracer experiments have been carried out. A harmless tracer was injected into the river and the tracer concentrations were measured along the river. Figure 6.14 shows an example of measured and computed concentrations at several stations.

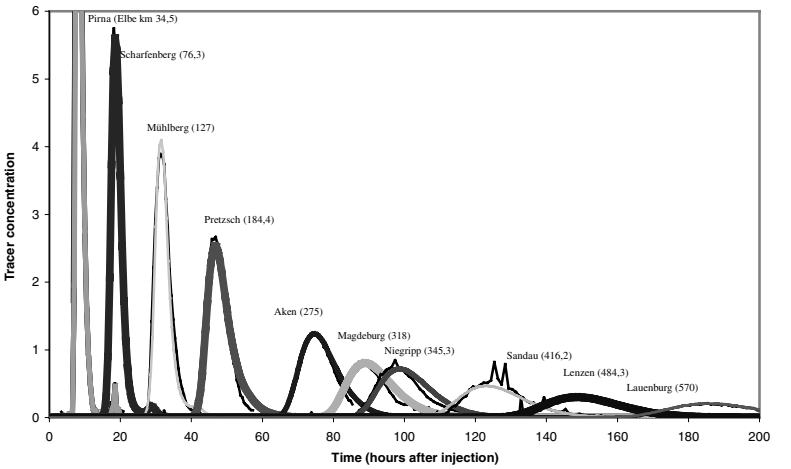


FIGURE 6.14
Measured and simulated tracer concentrations along the River Elbe.

The final application is a 2D model of the lower River Saar. To obtain a refined resolution near the river bed without increasing the number of mesh cells, the adaptive strategies were applied for mesh construction. Figure 6.15 shows the bottom elevation for an adaptive 296 $\times$ 72 mesh resolution. An example of simulation results of the flooded area during the flood in 1993 have been shown in Figure 6.3. The computations were carried out with the flux-difference splitting Roe scheme for the 2D shallow water equations and RODASP for time integration.

6.6 Conclusion

It has been demonstrated that the MOL approach can be applied successfully in practical river flow and transport problems. Adaptive meshes are useful if steep fronts occur and give information for 2D mesh construction. The proposed Rosenbrock-

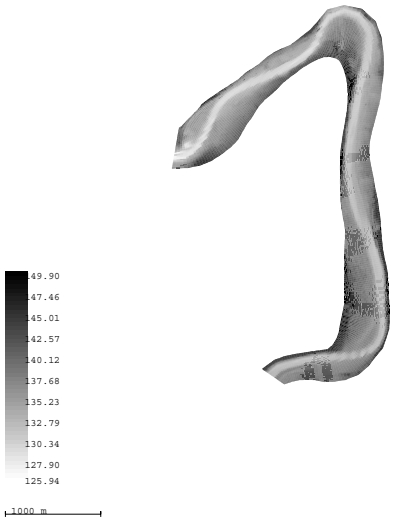


FIGURE 6.15
Bottom elevations (adaptive 296 × 72 mesh).

Wanner method for the numerical solution of the semi-discretized PDEs is a robust and reliable scheme and well suited for a daily use simulation software.

Acknowledgment

The authors would like to thank Michael Hilden, Adrian Q.T. Ngo, and Silke Rademacher for helpful contributions and cooperation and Dr. Klaus Wilke for his support.

References

[1] F. Alcrudo, P. Garcia-Navarro, and J.-M. Saviron, Flux-difference splitting for 1D open channel flow equations, *Int. J. f. Num. Meth. Fluids,* **14**, (1992), 1009–1018.

[2] F. Alcrudo and P. Garcia-Navarro, A high-resolution Godunov-type scheme in finite volumes for the 2D shallow-water equations, *Int. J. f. Num. Meth. Fluids,* **16**, (1993), 489–505.

[3] M. Berzins and R.M. Furzeland, A user's manual for sprint — A versatile software package for solving systems of algebraic ordinary and partial differential equations, *Part 1 (TNER.85.058), 2 (TNER.86.050) and 3 (TNER.88.034),* Thornton Research Centre, Shell Maatschappij, 1985, 1986, 1989.

[4] A.R. Curtis, M.J.D. Powell, and J.K. Reid, On the estimation of sparse jacobian matrices, *J. Inst. Maths. Applics.,* **13**, (1974), 117–119.

[5] I.S. Duff, MA28 — a set of FORTRAN subroutines for sparse unsymmetric linear equations, *AERE R8730, HMSO,* London, 1977.

[6] I.S. Duff, A.M. Erisman, and J.K. Reid, *Direct Methods for Sparse Matrices,* Clarendon Press, Oxford, 1986.

[7] V. von Dalwigk and G. Steinebach, *BfG-Mitteilungen Nr.19,* Mathematische Modelle in der Gewässerkunde — Stand und Perspektiven, Koblenz, 1999.

[8] J.E. Flaherty, et al., Adaptive methods for partial differential equations, *SIAM Publications,* 1989.

[9] W. Fröhlich, M. Heinz, G. Steinebach, and K. Wilke, Water level forecasts for navigation on the river Elbe and river Rhine — A contribution to an intelligent waterway, Proc. 29th PIANC International Navigation Congress The Hague 1998, Section I, subject 3, 55-60, *International Navigation Association,* 1998.

[10] E. Hairer, S.P. Nørsett, and G. Wanner, Solving ordinary differential equations I, *Springer Series in Computational Mathematics,* Berlin, Heidelberg, 1993.

[11] E. Hairer and G. Wanner, Solving ordinary differential equations II, *Springer Series in Computational Mathematics,* Berlin, Heidelberg, 1996.

[12] M. Hoschek, P. Rentrop, and Y. Wagner, Network approach and differential-algebraic systems in technical applications, *Surv. Math. Ind.,* **9**, (1999), 49–76.

[13] M. Hilden and G. Steinebach, ENO-discretizations in MOL-applications: some examples in river hydraulics, *Applied Numerical Mathematics,* **28**, (1998), 293–308.

[14] J.M. Hyman, Moving mesh methods for partial differential equations, in *Mathematics Applied to Science,* J. Goldstein, S. Rosencrans, and G. Sod, eds., Academic Press, (1988), 129–154.

[15] J. Kautsky and N.K. Nichols, Equidistributing meshes with constraints, *SIAM J. Sci. Stat. Comput.,* **1**, (1980), 499–511.

[16] P. Krahe, Water-level and discharge forecasts in the river Moselle basin on the basis of measured and forecasted meteorological data, *Proc. of the Workshop*

on Flood Forecasting, Czech Hydrometeorological Institute, 108–117 Prague, 1998.

[17] R.J. LeVeque, Numerical methods for conservation laws, *Lectures in Mathematics,* Birkhäuser, Zürich, 1992.

[18] Ch. Lubich and A. Ostermann, Runge–Kutta methods for parabolic equations and convolution quadrature, *Math. Comp.,* **60**, (1992), 105–131.

[19] Q.T. Ngo, Numerical simulation of river flow problems based on a finite volume model, Diploma thesis, Univ. Kaiserslautern, 1999.

[20] A. Ostermann and M. Roche, Runge–Kutta methods for partial differential equations and fractional orders of convergence, *Math. Comp.,* **59**, (1992), 403–420.

[21] A. Ostermann and M. Roche, Rosenbrock methods for partial differential equations and fractional orders of convergence, *SIAM J. Numer. Anal.,* **30**, (1993), 1084–1098.

[22] L.R. Petzold, Observations on an adaptive moving grid method for one-dimensional systems of partial differential equations, *Applied Numer. Math.,* **3**, (1987), 347–360.

[23] A. Prothero and A. Robinson, The stability and accuracy of one-step methods, *Math. Comp.,* **28**, (1974), 145–162.

[24] P. Rentrop, M. Hilden, and G. Steinebach, Wissenschaftliches Rechnen, *Der Ingeniuer in der Wasser- und Schifffahrtsverwaltung,* **19**, (1999), 19–23.

[25] P. Rentrop and G. Steinebach, Model and numerical techniques for the alarm system of river Rhine, *Surveys Math. Industry,* **6**, (1997), 245–265.

[26] P. Rentrop and G. Steinebach, A method of lines approach for river alarm systems, *ECMI Progress in Industrial Mathematics at ECMI'96,* Brons, M., Bendsoe, M.P., Sorensen, M.P., eds., 12–19, Teubner Stuttgart, 1997.

[27] W.E. Schiesser, *The Numerical Methods of Lines,* Academic Press, San Diego, CA, 1991.

[28] S. Scholz, Order barriers for the B-convergence of ROW methods, *Computing,* **41**, (1989), 219–235.

[29] M. Spreafico and A. van Mazijk, Alarmmodell Rhein, Ein Modell für die operationelle Vorhersage des Transportes von Schadstoffen im Rhein, *KHR-Bericht Nr. I-12,* Lelystad, 1993.

[30] G. Steinebach, Order-reduction of ROW-methods for DAEs and method of lines applications, *Preprint-Nr. 1741, FB Mathematik, TH Darmstadt,* 1995.

[31] G. Steinebach, Using hydrodynamic models in forecast systems for large rivers, *Proc. Advances in Hydro-Science and -Engineering, Vol. 3 incl. CD-ROM,* Holz, K.P., Bechteler, W., Wang, S.S.Y., Kawahara, M., eds., Cottbus, 1998.

[32] G. Steinebach and A.Q.T. Ngo, A method of lines flux-difference splitting finite volume approach for 1D and 2D river flow problems, to appear in *Godunov Methods: Theory and Applications,* E.F. Toro, ed., Kluwer Academic/Plenum Publishers, 2001.

[33] G. Steinebach and K. Wilke, Flood forecasting and warning on the River Rhine, Water and Environmental Management, *J. CIWEM,* **14**, (2000), 39–44.

[34] J.J. Stoker, *Water Waves, the Mathematical Theory with Applications,* Interscience Publishers, New York, 1957.

[35] E.F. Toro, *Riemann Solvers and Numerical Methods for Fluid Dynamics,* Springer, Berlin, Heidelberg, 1999.

[36] H.A. van der Vorst, Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems, *SIAM J. Sci. Stat. Comp.,* **13**, (1992), 631–644.

[37] J.G. Verwer, Convergence and order reduction of diagonally implicit Runge–Kutta schemes in the method of lines, in *Griffiths, Watson: Numerical Analysis, Pitman Research Notes in Mathematics*, 220–237, 1986.

[38] J.G. Verwer, W.H. Hundsdorfer, and J.G. Blom, Numerical time integration for air pollution models, *Modeling, Analysis and Simulation Report MAS-R9825,* 58 p., CWI Amsterdam, 1998.

[39] C.B. Vreugdenhil, Numerical methods for shallow-water flow, Kluwer Acad. Pub., Dordrecht, 1994.

[40] K. Wilke, Mehrkanalfiltermodell (MKF), in *Beschreibung hydrologischer Vorhersagemodelle im Rheineinzugsgebiet, Bericht I-7 der KHR,* Lelystad, 71–85, 1988.

Chapter 7

An Adaptive Mesh Algorithm for Free Surface Flows in General Geometries

Mark Sussman[1]

7.1 Introduction

In this chapter we present an adaptive method for computing incompressible free surface flows in general geometries. An example of two flows that we consider are (1) flows in a 3D jetting device (Figure 7.11) and (2) ship waves (Figure 7.9). Our computations are done on an adaptive grid as described by Berger and Colella [6] and Almgren et al. [1]. The free surface separating the gas and liquid is modeled using "embedded boundary" techniques; specifically, a coupled level set and volume of fluid method is used [50]. Our method for modeling the free surface allows for the arbitrary merge and break-up of fluid mass while maintaining excellent mass conservation. An "embedded boundary" (a.k.a. Cartesian grid [20]) method is also used to represent irregular geometries (e.g., ship hull or jetting housing). In the process of describing our methods for modeling the free surface and geometry, we also present a new (easy) way for enforcing the contact angle boundary condition at points where the free surface meets the geometry.

7.1.1 Overview: Adaptive Gridding

For the problems we consider, dynamic adaptive grid refinement is important. The error is largest in regions near the free surface. A finer mesh is needed at the free surface more than elsewhere. There are quite a few numerical techniques for implementing adaptivity. In the finite element framework, the reader is referred to the following works [47, 37, 19, 9, 35, 63, 30, 36]. In the finite difference framework

[1]Work supported in part by NSF # DMS 97-06847, DOE (MICS) program contract DE-AC03-76SF00098, DOE (MICS) program contract DE-FG03-95ER25271, and an ASCI grant from the Los Alamos National Laboratory.

(i.e., uniform rectangular mesh), the reader is referred to the following dynamic adaptive grid methods [6, 1, 48, 62, 34]. Other methods that allow one to add grid resolution where needed are so-called Overset-grid methods [13, 18].

In our work, we adopt the finite-difference-based adaptive grid techniques described in [6] and extended to incompressible flows in [1]. The idea behind these methods is that the basic numerical methodology used for a single rectangular mesh should be unchanged when generalizing to a collection of rectangular meshes with differing resolutions. The only additional logic added to the base numerical algorithm is to be able to handle boundary conditions at coarse-fine grid interfaces or fine-fine grid interfaces. Another advantage to the adaptive grid techniques that we adopt is that these techniques are naturally parallelizable. Each rectangular grid can be assigned a different processor.

7.1.2 Overview: Free Surface Model

There are two classes of free surface algorithms commonly used for incompressible two-phase flow problems: (1) body-fitted or Lagrangian techniques and (2) embedded boundary techniques.

In body-fitted/Lagrangian techniques [11, 10, 29, 61, 60], the computational grid is aligned with the free surface at all times. These methods are generally more accurate than their embedded boundary counterparts and also more efficient. Unfortunately, these methods will break down when the free surface develops a change in topology unless special measures are taken [51]. Also, there is a regridding issue as the free-surface deforms.

In embedded boundary techniques [43, 52, 59, 12, 46, 23, 44, 58, 17, 16, 45, 27, 31, 26, 25] the free surface is allowed to cut through the computational grid. The computational grid remains fixed while the free surface deforms arbitrarily. These methods are typically more robust and easier to program than their body-fitted/Lagrangian counterparts. On the other hand, one generally cannot achieve higher than first-order accuracy using embedded boundary techniques for the free surface.

In our work we adopt the "coupled level set volume of fluid" method described in [50]. This method falls in the category of an "embedded boundary" technique. The free surface cuts through the computational grid. The free surface is represented "implicitly" by two field variables: (1) the level set function $\phi(\mathbf{x}, t)$, positive in liquid and negative in gas, (2) the volume of fluid function $F(\mathbf{x}, t)$, 1 in liquid, 0 in gas, and $0 < F < 1$ in partially filled computational elements.

Remarks:

1. Front tracking approaches [59, 58] are generally more accurate approaches for representing the embedded free surface than level set [52] or volume-of-fluid [12] methods; but front tracking methods are complicated to implement for 3D problems with multiple changes in topology (e.g., wave sloshing, droplet break-up).

2. In our work we solve for the flow in both the liquid and gas. Some methods (e.g., [16]) solve for the liquid only and assume pressure is constant in the gas.

Methods that solve in the liquid only are generally more efficient, especially if the flow is comprised of a relatively small portion of liquid.

7.1.3 Overview: Modeling Flows in General Geometries

As with algorithms for free surfaces, algorithms for flows in general geometries fall into two categories: (1) body-fitted and (2) embedded boundary.

In body-fitted techniques (structured, unstructured, mapped grids) [55, 14, 7, 32, 38, 5, 41, 56], the computational grid is aligned with the geometry. These methods are generally more accurate than their embedded boundary counterparts and also more efficient. A drawback with body-fitted techniques for flows in general geometries is that one has to generate an appropriate grid and design a numerical method which operates on non-uniform/mapped grids.

In embedded boundary techniques (a.k.a. Cartesian grid methods) [42, 24, 33, 57, 2], the general geometry cuts through the computational grid. This allows one to use algorithms designed for fixed rectangular grids with little modification.

In our work, the irregular boundary (e.g., ship hull or jetting device housing) is represented as the zero level set of a *second* level set function ψ along with the corresponding area fractions A and volume fractions V. ψ is positive in the active flow region and negative elsewhere. $V = 1$ for computational elements fully contained within the active flow region and $V = 0$ for computational elements fully outside the active flow region. The representation of irregular boundaries via area fractions and volume fractions has been used previously in the following work for incompressible flows [2, 57].

7.2 Governing Equations

We assume that both the liquid and the gas are incompressible, immiscible fluids. The equations for both the liquid and the gas have the form

$$\begin{aligned} \boldsymbol{U}_t + \nabla \cdot (\boldsymbol{U}\boldsymbol{U}) &= -\frac{\nabla p}{\rho} + \frac{\mu \Delta \boldsymbol{U}}{\rho} - \mathbf{G}. \\ \nabla \cdot \boldsymbol{U} &= 0 \end{aligned} \tag{7.1}$$

The quantities ρ and μ in (7.1) represent the values of liquid or gas depending on what fluid one is in. The free surface boundary conditions are as follows:

$$\begin{aligned} \boldsymbol{U}^g &= \boldsymbol{U}^l \\ \left(2\mu^l D^l - 2\mu^g D^g\right) \cdot \mathbf{n} &= \left(p^l - p^g + \gamma\kappa\right)\mathbf{n}\,. \end{aligned} \tag{7.2}$$

$\mathbf{n}$ is the outward normal drawn from the gas into the liquid. κ is the local mean curvature of the free surface. D is the rate of deformation tension,

$$D = \frac{1}{2}\left(\nabla \boldsymbol{U} + \nabla \boldsymbol{U}^T\right) .$$

We shall enforce the no-slip boundary conditions at solid walls,

$$\boldsymbol{U} = 0 . \tag{7.3}$$

Also, at solid walls, we enforce a contact angle boundary condition,

$$\mathbf{n} \cdot \mathbf{n}_{\text{wall}} = \cos(\theta) , \tag{7.4}$$

where θ is a user-defined contact angle and $\mathbf{n}_{\text{wall}}$ is the outward normal drawn from the active flow region into the geometry region.

The explicit enforcement of the free surface boundary condition (7.2) can be complicated in 3D; especially for interfaces that can merge or break. Instead of solving in the gas and liquid separately, and then coupling the solutions at the free surface, we solve the following equations for both the gas and the liquid:

$$\boldsymbol{U}_t = -\nabla \cdot (\boldsymbol{U}\boldsymbol{U}) - \frac{\nabla p}{\rho(\phi)} + \frac{\nabla \cdot 2\mu(\phi) D}{\rho(\phi)} - \frac{\gamma \kappa(\phi) \nabla H(\phi)}{\rho(\phi)} - \mathbf{G} . \tag{7.5}$$

$$\begin{aligned} \nabla \cdot \boldsymbol{U} &= 0 \\ \phi_t + \boldsymbol{U} \cdot \nabla \phi &= 0 \\ \rho(\phi) &\equiv \rho_l H(\phi) + \rho_g (1 - H(\phi)) \\ \mu(\phi) &\equiv \mu_l H(\phi) + \mu_g (1 - H(\phi)) \\ H(\phi) &= \begin{cases} 1 & \phi > 0 \\ 0 & \text{otherwise} \end{cases} \\ \kappa(\phi) &\equiv \nabla \cdot \frac{\nabla \phi}{|\nabla \phi|} \end{aligned} \tag{7.6}$$

The level set function ϕ is defined to be positive in the liquid and negative in the gas. The motion of the free surface is determined from the level set equation (7.6). The level set equation tells us that ϕ remains constant on particle paths. In other words, if the zero level set of ϕ coincides with the free surface, then solutions at a later time will also have the zero level set of ϕ coinciding with the free surface. It has been shown by Chang et al. [15] that weak solutions of (7.5) satisfy the free surface boundary conditions (7.2). We never have to explicitly enforce the free surface boundary conditions. They are implicitly enforced through the use of the Heaviside function $H(\phi)$.

7.2.1 Projection Method

In order to solve (7.5), we use a variable density projection method [4] which is a generalization of the constant density projection method presented by [3]. First, one

can rewrite (7.5) in the following form:

$$\vec{W}_d + \nabla p/\rho = \vec{W} \tag{7.7}$$

where W_d represents $\boldsymbol{U}_t$ and $\vec{W}$ represents,

$$\vec{W} = -\nabla \cdot (\boldsymbol{U}\boldsymbol{U}) + \frac{\nabla \cdot 2\mu(\phi)D}{\rho(\phi)} - \frac{\gamma\kappa\nabla H(\phi)}{\rho(\phi)} - \mathbf{G}.$$

After taking the divergence of both sides of (7.7), we use the continuity equation in order to set $\nabla \cdot \vec{W}_d = 0$, thus resulting in the following equation for the pressure field:

$$\nabla \cdot \frac{\nabla p}{\rho} = \nabla \cdot \vec{W} \; . \tag{7.8}$$

In order to impose a no-outflow condition at solid walls, one has the following Neumann boundary condition on p:

$$\frac{\nabla p}{\rho} \cdot \mathbf{n}_{\text{wall}} = \vec{W} \cdot \mathbf{n}_{\text{wall}}.$$

Once the pressure field p is determined from (7.8), one can then update $\vec{W}_d$ as

$$\vec{W}_d = \vec{W} - \nabla p/\rho \; .$$

We shall denote the projection operator as:

$$\vec{W}_d = P_\rho(\vec{W}) \; .$$

The resulting equations to be solved now, when written in terms of the projection operator, are

$$\begin{aligned} \boldsymbol{U}_t &= P_\rho\left(-\nabla \cdot (\boldsymbol{U}\boldsymbol{U}) + \frac{\nabla \cdot 2\mu(\phi)D}{\rho(\phi)} - \frac{\gamma\kappa\nabla H(\phi)}{\rho(\phi)} - \mathbf{G}\right) \\ \phi_t + \boldsymbol{U} \cdot \nabla\phi &= 0 \; . \end{aligned} \tag{7.9}$$

7.3 Discretization

We discretize (7.9) on a fixed rectangular grid. The free surface and geometry are embedded within the grid. The free surface is represented as the zero level set of a smooth function ϕ. The geometry is represented as the zero level set of a smooth function ψ. Important to our numerical scheme is the volume fraction F of liquid in each computational element. For each computational element, Ω_{ij}, the volume fraction of liquid is defined as:

$$F_{ij} \equiv \frac{1}{|\Omega_{ij}|} \int_{\Omega_{ij}} H(\phi) d\mathbf{x} \; .$$

Also important to our numerical scheme is the geometry volume fraction V of Ω_{ij}, and the geometry area fraction A of $\Gamma_{i+1/2,j}$,

$$\begin{aligned} V_{ij} &\equiv \frac{1}{|\Omega_{ij}|}\int_{\Omega_{ij}} H(\psi)d\mathbf{x}\,. \\ A_{i+1/2,j} &\equiv \frac{1}{|\Gamma_{i+1/2,j}|}\int_{\Gamma_{i+1/2,j}} H(\psi)d\mathbf{x}\,. \end{aligned}$$

$\Gamma_{i+1/2,j}$ represents the left face of a computational element; similar definitions apply to $\Gamma_{i-1/2,j}$, $\Gamma_{i,j+1/2}$, $\Gamma_{i,j-1/2}$.

The state variables $\mathbf{u}_{ij}$, ϕ_{ij}, and F_{ij} are stored at the center of each computational grid cell. The pressure $p_{i+1/2,j+1/2}$ is stored at the cell corners (nodes).

A simple first-order discretization is as follows:

1. Given ϕ^n, F^n, $\boldsymbol{U}^n$

2a. Set $\boldsymbol{U}^n = 0$ in computational cells where $V_{ij} = 0$, i.e., velocity satisfies no-slip conditions on geometry walls and velocity is identically zero within the geometry. This is a first-order boundary condition; for an example of higher-order Cartesian Grid discretizations, see [33, 20].

2b. Extend ϕ^n into regions where $V_{ij} < 1$. The extension procedure "implicitly" enforces the contact angle boundary condition

$$\mathbf{n}\cdot\mathbf{n}_{\text{wall}} = \cos(\theta)\,.$$

The extension procedure is described in Section 7.5.2.

3. Form

$$\boldsymbol{V}^n = -[\nabla(\boldsymbol{U}\boldsymbol{U})]^n + \frac{\nabla\cdot(2\mu(\phi^n)D^n)}{\rho(\phi^n)} - \frac{\gamma\kappa(\phi^n)\nabla H(\phi^n)}{\rho(\phi^n)} - \mathbf{G}$$

4. Update the position of the free surface using the "coupled level set volume-of-fluid" (CLS) method (see Section 7.4),

$$\begin{aligned} \phi^{n+1} &= \phi^n - \Delta t[\nabla\cdot(\mathbf{u}^{MAC}\phi)]^n \\ F^{n+1} &= F^n - \Delta t[\nabla\cdot(\mathbf{u}^{MAC}F)]^n \end{aligned}$$

Remark: modifications to the (CLS) method for general geometries are provided in Section 7.5.3.

5. Update the velocity (pressure solve)

$$\boldsymbol{U}^{n+1} = \boldsymbol{U}^n + \Delta t\, P_{\rho(\phi^n)}(\boldsymbol{V}^n) \tag{7.10}$$

6. Reinitialize ϕ^{n+1} using current values for ϕ^{n+1} and F^{n+1} (maintain ϕ^{n+1} as the signed distance from the zero level set of ϕ^{n+1}).

Remarks:

- The nonlinear term $[\nabla(\boldsymbol{UU})]^n$ is discretized using a second-order, slope-limited predictor corrector method described in [48].
- For the sake of readability, we describe the first order in time method above. In practice we employ the second-order "Crank-Nicolson" time discretization described by Bell et al. [3] and also implemented in [48].
- The step that takes the most time is the projection step (7.10). In the projection step, we solve the following discretized equation for p,

$$\nabla \cdot \frac{1}{\rho(\phi^n)} \nabla p = \nabla \cdot \boldsymbol{V}^n \,, \tag{7.11}$$

  subject to the boundary conditions

$$\frac{\nabla p}{\rho(\phi^n)} \cdot \mathbf{n}_{\text{wall}} = \boldsymbol{V}^n \cdot \mathbf{n}_{\text{wall}} \,.$$

  Details of how we enforce the no-flow condition at solid walls are given in Section 7.5.1.2 below.

  In order to solve the resulting linear system, we use the multigrid preconditioned conjugate gradient method [54].

- We use time-step constraints due to the CFL condition, viscous terms, and surface-tension terms. For jetting problems, it is the surface-tension, time-step constraint which is most restrictive.

7.3.1 Thickness of the Interface

In the discretization of (7.5) we replace $H(\phi)$ with $H_\epsilon(\phi)$ where $H_\epsilon(\phi)$ is defined as

$$H_\epsilon(\phi) = \begin{cases} 0 & \phi < -\epsilon \\ \frac{1}{2}\left[1 + \frac{\phi}{\epsilon} + \frac{1}{\pi}\sin(\pi\frac{\phi}{\epsilon})\right] & |\phi| \le \epsilon \\ 1 & \phi > \epsilon \end{cases}$$

For most of our computations, $\epsilon = 3\Delta x$. For a few 3D problems (see remark in Section 7.7.1.1) we set $\epsilon = 4\Delta x$. Without smoothing (i.e., $\epsilon = 0$), our method yields oscillatory results, probably due to the fact that with zero thickness the tangential velocity jumps sharply across the free surface (high Reynolds number flows).

Because we give the interface a thickness ϵ, we find it necessary to maintain the level set function ϕ as a signed distance function. Otherwise, one would not have a uniform thickness. Over time, the nonzero level sets can stack up in some regions and spread apart in others. In [52], a comparison is given between computations with and without reinitialization for a rising steady-gas-bubble problem; it was shown that reinitialization is needed in order to preserve the steady solution.

As a remark, in work by Sussman and Smereka [51] the level set method was compared to the boundary integral method for bubble and drop problems. The two methods compared very well despite the fact that the level-set method gives the interface a thickness ϵ whereas the boundary-integral method considers the interface as sharp.

7.4 Coupled Level Set Volume of Fluid Advection Algorithm

In this section, we describe the 2D coupled level set and volume of fluid (CLS) algorithm for representing the free surface. For more details, e.g., axisymmetric and 3D implementations, see [50]. In the CLS algorithm, the position of the interface is updated through the level-set equation and volume-of-fluid equation,

$$\begin{aligned} \phi_t + \nabla \cdot \left(\boldsymbol{U}^{MAC} \phi \right) &= 0 \\ F_t + \nabla \cdot \left(\boldsymbol{U}^{MAC} F \right) &= 0 \,. \end{aligned}$$

In order to implement the CLS algorithm, we are given a discretely divergence-free velocity field $\mathbf{u}^{MAC}$ defined on the cell faces (MAC grid),

$$\frac{u_{i+\frac{1}{2},j} - u_{i-\frac{1}{2},j}}{\Delta x} + \frac{v_{i,j+\frac{1}{2}} - v_{i,j-\frac{1}{2}}}{\Delta y} = 0 \,. \tag{7.12}$$

Given ϕ_{ij}^n, F_{ij}^n, and $\boldsymbol{U}^{MAC}$, we use a "coupled" second-order, conservative-operator split advection scheme in order to find ϕ_{ij}^{n+1} and F_{ij}^{n+1}. The 2D operator split algorithm for a general scalar s follows as

$$\tilde{s}_{ij} = \frac{s_{ij}^n + \frac{\Delta t}{\Delta x}\left(G_{i-\frac{1}{2},j} - G_{i+\frac{1}{2},j} \right)}{1 - \frac{\Delta t}{\Delta x}\left(u_{i+\frac{1}{2},j} - u_{i-\frac{1}{2},j} \right)} \tag{7.13}$$

$$s_{ij}^{n+1} = \tilde{s}_{ij} + \frac{\Delta t}{\Delta y}\left[\left(\tilde{G}_{i,j-\frac{1}{2}} - \tilde{G}_{i,j+\frac{1}{2}} \right) + \tilde{s}_{ij}\left(v_{i,j+\frac{1}{2}} - v_{i,j-\frac{1}{2}} \right) \right], \tag{7.14}$$

where $G_{i+\frac{1}{2},j} = s_{i+\frac{1}{2},j} u_{i+\frac{1}{2},j}$ denotes the flux of s across the right edge of the (i, j)th cell and $\tilde{G}_{i,j+\frac{1}{2}} = \tilde{s}_{i,j+\frac{1}{2}} v_{i,j+\frac{1}{2}}$ denotes the flux across the top edge of the (i, j)th cell. The operations (7.13) and (7.14) represent the case when one has the "x-sweep" followed by the "y-sweep." After every time step the order is reversed; "y-sweep" (done implicitly) followed by the "x-sweep" (done explicitly).

The scalar flux $s_{i+\frac{1}{2},j}$ is computed differently depending on whether s represents the level-set function ϕ or the volume fraction F.

For the case when s represents the level-set function ϕ, we have the following representation for $s_{i+\frac{1}{2},j}$ ($u_{i+\frac{1}{2},j} > 0$):

$$s_{i+\frac{1}{2},j} = s^n_{ij} + \frac{\Delta x}{2}(D_x s)^n_{ij} + \frac{\Delta t}{2}\left(-u_{i+\frac{1}{2},j}\,(D_x s)^n_{ij}\right)$$

where

$$(D_x s)^n_{ij} \equiv \frac{s^n_{i+1,j} - s^n_{i-1,j}}{\Delta x}\ .$$

The above discretization is motivated by the second-order, predictor-corrector method described in [3] and the references therein.

For the case when s represents the volume fraction F we have the following representation for $s_{i+\frac{1}{2},j}$ ($u_{i+\frac{1}{2},j} > 0$):

$$s_{i+\frac{1}{2},j} = \frac{\int_\Omega H(\phi^{n,R}_{ij}(x,y))d\Omega}{u_{i+\frac{1}{2},j}\Delta t\Delta y} \tag{7.15}$$

where

$$\Omega \equiv \left\{(x,y)|x_{i+\frac{1}{2}} - u_{i+\frac{1}{2},j}\Delta t \le x \le x_{i+\frac{1}{2}} \text{ and } y_{j-\frac{1}{2}} \le y \le y_{j+\frac{1}{2}}\right\}$$

The integral in (7.15) is evaluated by finding the volume cut out of the region of integration by the line represented by the zero level set of $\phi^{n,R}_{ij}$.

The term $\phi^{n,R}_{ij}(x,y)$ found in (7.15) represents the linear reconstruction of the interface in cell (i,j). In other words, $\phi^{n,R}_{ij}(x,y)$ has the form

$$\phi^{n,R}_{ij}(x,y) = a_{ij}(x - x_i) + b_{ij}(y - y_j) + c_{ij}\ . \tag{7.16}$$

A simple choice for the coefficients a_{ij} and b_{ij} is as follows:

$$a_{ij} = \frac{1}{2\Delta x}(\phi_{i+1,j} - \phi_{i-1,j}) \tag{7.17}$$

$$b_{ij} = \frac{1}{2\Delta y}(\phi_{i,j+1} - \phi_{i,j-1})\ . \tag{7.18}$$

The intercept c_{ij} is determined so that the line represented by the zero level set of (7.16) cuts out the same volume in cell (i,j) as specified by F^n_{ij}. In other words, the following equation is solved for c_{ij}:

$$\frac{\int_\Omega H(a_{ij}(x - x_i) + b_{ij}(y - y_j) + c_{ij})d\Omega}{\Delta x\Delta y} = F^n_{ij}$$

where

$$\Omega \equiv \left\{(x,y)|x_{i-\frac{1}{2}} \le x \le x_{i+\frac{1}{2}} \text{ and } y_{j-\frac{1}{2}} \le y \le y_{j+\frac{1}{2}}\right\}\ .$$

After ϕ^{n+1} and F^{n+1} have been updated according to (7.13) and (7.14) we "couple" the level-set function to the volume fractions as a part of the level-set reinitialization step. The level-set reinitialization step replaces the current value of ϕ^{n+1} with the exact distance to the VOF reconstructed interface. At the same time, the VOF reconstructed interface uses the current value of ϕ^{n+1} to determine the slopes of the piecewise linear reconstructed interface.

Remarks:

- The distance is only needed in a tube of K cells wide $K = \epsilon/\Delta x + 2$, therefore, we can use "brute force" techniques for finding the exact distance. See [50] for details.
- During the reinitialization step we truncate the volume fractions to be 0 or 1 if $|\phi| > \Delta x$. Although we truncate the volume fractions, we still observe that mass is conserved to within a fraction of a percent for our test problems.

7.5 Discretization in General Geometries

The discretization of the following items need additional explanation in general geometries:

1. Projection step
2. Surface tension (contact angle boundary conditions)
3. CLS advection

7.5.1 Projection Step in General Geometries

7.5.1.1 MAC Project

In order to construct the advective "MAC" velocities (7.12) located at cell face centroids (see [1] for further details of the "MAC" projection step), a "MAC" projection step is needed.

In the MAC projection step, we solve the following discretized equation for p:

$$\nabla \cdot \frac{1}{\rho(\phi^n)} \nabla p = \nabla \cdot \boldsymbol{V}^n \,, \tag{7.19}$$

subject to the boundary conditions

$$\frac{\nabla p}{\rho(\phi^n)} \cdot \mathbf{n}_{\text{wall}} = \boldsymbol{V}^n \cdot \mathbf{n}_{\text{wall}} \,. \tag{7.20}$$

In order to discretely enforce the boundary conditions (7.20) at the geometry surface, we use a finite-volume approach for discretizing (7.19).

Given an irregular computational element Ω_{ij} (see Figure 7.1), we have

$$\int_{\Omega_{ij}} \nabla \cdot \mathbf{U} dV = \int_{\partial\Omega_{ij}} \mathbf{U} \cdot \mathbf{n}_{\text{wall}}\, dA\ .$$

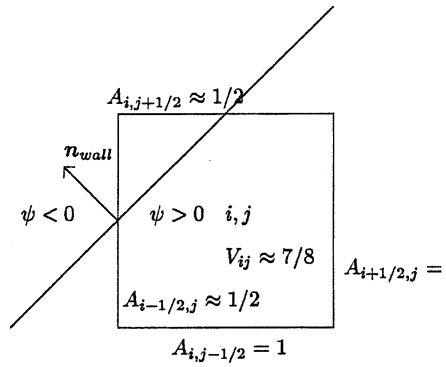


FIGURE 7.1
Diagram of computational element (i, j) that is cut by the embedded geometry.

The divergence theorem motivates the following second-order approximation of the divergence $\nabla \cdot \mathbf{U}$ at the centroid of Ω_{ij}:

$$\nabla \cdot \mathbf{U} \approx \frac{1}{|\Omega_{ij}|} \int_{\partial\Omega_{ij}} \mathbf{U} \cdot \mathbf{n}_{\text{wall}}\, dA\ . \tag{7.21}$$

In terms of geometry volume fractions V_{ij} and area fractions $A_{i+1/2,j}$, (7.21) becomes,

$$\begin{aligned} \nabla \cdot \mathbf{U} \quad \approx \quad & \frac{1}{V_{ij}\Delta x \Delta y}\Big[\left(A_{i+1/2,j}\Delta y\right) u_{i+1/2,j} - \left(A_{i-1/2,j}\Delta y\right) u_{i-1/2,j} \\ & + \left(A_{i,j+1/2}\Delta x\right) v_{i,j+1/2} - \left(A_{i,j-1/2}\Delta x\right) v_{i,j-1/2} \\ & - L_{ij}^{\text{wall}} \mathbf{U}_{ij}^{\text{wall}} \cdot \mathbf{n}_{\text{wall}} \Big]\ . \end{aligned} \tag{7.22}$$

For a zero flux boundary condition at the wall, the last term in (7.22), $L_{ij}^{\text{wall}} \mathbf{U}_{ij}^{\text{wall}} \cdot \mathbf{n}_{\text{wall}}$, is zero.

The finite-volume approach, when applied to the divergence operator in (7.19) becomes:

$$\begin{aligned} \nabla \cdot \frac{1}{\rho(\phi^n)} \nabla p \quad \approx \quad & \frac{1}{V_{ij}\Delta x \Delta y}\Big[A_{i+1/2,j}\Delta y (p_x/\rho)_{i+1/2,j} - A_{i-1/2,j}\Delta y (p_x/\rho)_{i-1/2,j} \\ & + A_{i,j+1/2}\Delta x (p_y/\rho)_{i,j+1/2} - A_{i,j-1/2}\Delta x (p_y/\rho)_{i,j-1/2} \\ & - L_{ij}^{\text{wall}} (\nabla p/\rho)_{ij}^{\text{wall}} \cdot \mathbf{n}_{\text{wall}} \Big]\ . \end{aligned}$$

and

$$\nabla \cdot \boldsymbol{V}^n \approx \frac{1}{V_{ij}\Delta x \Delta y}\Big[\left(A_{i+1/2,j}\Delta y\right) u_{i+1/2,j} - \left(A_{i-1/2,j}\Delta y\right) u_{i-1/2,j}$$
$$+ \left(A_{i,j+1/2}\Delta x\right) v_{i,j+1/2} - \left(A_{i,j-1/2}\Delta x\right) v_{i,j-1/2} - L_{ij}^{\text{wall}} \boldsymbol{V}_{ij}^{n,wall} \cdot \mathbf{n}_{\text{wall}} \Big] .$$

Due to the no-flow condition (7.20), the terms $L_{ij}^{\text{wall}}(\nabla p/\rho)_{ij}^{\text{wall}} \cdot \mathbf{n}_{\text{wall}}$ and $L_{ij}^{\text{wall}} \boldsymbol{V}_{ij}^{n,wall} \cdot \mathbf{n}_{\text{wall}}$ cancel each other. The resulting discretization for p is:

$$\begin{aligned} &A_{i+1/2,j}\Delta y (p_x/\rho)_{i+1/2,j} - A_{i-1/2,j}\Delta y (p_x/\rho)_{i-1/2,j} \\ &\quad + A_{i,j+1/2}\Delta x (p_y/\rho)_{i,j+1/2} - A_{i,j-1/2}\Delta x (p_y/\rho)_{i,j-1/2} \\ &= (A_{i+1/2,j}\Delta y) u_{i+1/2,j} - (A_{i-1/2,j}\Delta y) u_{i-1/2,j} \\ &\quad + (A_{i,j+1/2}\Delta x) v_{i,j+1/2} - (A_{i,j-1/2}\Delta x) v_{i,j-1/2} \end{aligned}$$

where, for example, $(p_x)_{i+1/2,j}$ is discretized as

$$\frac{p_{i+1,j} - p_{i,j}}{\Delta x} .$$

7.5.1.2 Nodal Projection

The "nodal" projection step solves (7.11) for $p_{i+1/2,j+1/2}$ subject to the following boundary conditions at the embedded boundary:

$$\frac{\nabla p}{\rho(\phi^n)} \cdot \mathbf{n}_{\text{wall}} = \boldsymbol{V}^n \cdot \mathbf{n}_{\text{wall}}. \tag{7.23}$$

The following modification of (7.11) implicitly enforces (7.23),

$$\nabla \cdot \frac{1}{\rho(\phi^n)} H(\psi) \nabla p = \nabla \cdot H(\psi) \boldsymbol{V}^n , \tag{7.24}$$

where H is the Heaviside function. In other words, weak solutions of (7.24) automatically satisfy (7.23). We solve (7.24) in a fixed rectangular domain that contains the embedded geometry (the zero level set of ψ).

In order to discretize (7.24), we modify the standard discretization of the following pressure equation:

$$\nabla \cdot \frac{1}{\rho(\phi^n)} \nabla p = \nabla \cdot \boldsymbol{V}^n ,$$

by replacing $\frac{1}{\rho(\phi_{ij}^n)}$ with $\frac{V_{ij}}{\rho(\phi_{ij}^n)}$ and by replacing $\boldsymbol{V}_{ij}^n$ with $V_{ij}\boldsymbol{V}_{ij}^n$.

Remark:
Our discretization of (7.24) is only first-order accurate at cells that have a partial geometry volume fraction $0 < V_{ij} < 1$. For possible higher order discretizations, we refer the interested reader to [33, 20].

7.5.2 Contact-Angle Boundary Condition in General Geometries

The contact-angle boundary condition at solid walls is given by (7.4). In terms of ϕ and ψ, (7.4) becoc

$$\frac{\nabla\phi}{|\nabla\phi|}\cdot\frac{-\nabla\psi}{|\nabla\psi|}=\cos(\theta)\ .$$

In Figure 7.2, we show a diagram of how the contact angle θ is defined in terms of how the free surface intersects the geometry surface.

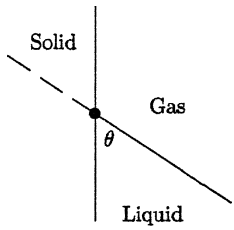


FIGURE 7.2
Diagram of gas/liquid interface meeting at the solid. The dashed line represents the imaginary interface created through the level-set extension procedure.

The "extension" equation has the form of an advection equation:

$$\phi_\tau+\mathbf{u}^{\text{extend}}\cdot\nabla\phi=0 \qquad\qquad \psi<0 \tag{7.25}$$

In regions where $\psi\geq 0$, ϕ is left unchanged.

For a 90° contact angle, we have

$$\mathbf{u}^{\text{extend}}=-\frac{\nabla\psi}{|\nabla\psi|}\ .$$

In other words, information propagates normal to the geometry surface.

For contact angles different from 90°, the following procedure is taken to find $\mathbf{u}^{\text{extend}}$:

$$\begin{aligned}
\mathbf{n} &\equiv \frac{\nabla\phi}{|\nabla\phi|}\\
\mathbf{n}_{\text{wall}} &\equiv -\frac{\nabla\psi}{|\nabla\psi|}\\
\mathbf{n}_1 &\equiv -\frac{\mathbf{n}\times\mathbf{n}_{\text{wall}}}{|\mathbf{n}\times\mathbf{n}_{\text{wall}}|}\\
\mathbf{n}_2 &\equiv -\frac{\mathbf{n}_1\times\mathbf{n}_{\text{wall}}}{|\mathbf{n}_1\times\mathbf{n}_{\text{wall}}|}\\
c &\equiv \mathbf{n}\cdot\mathbf{n}_2
\end{aligned}$$

$$\mathbf{u}^{\text{extend}}=\begin{cases}
\frac{\mathbf{n}_{\text{wall}}-\cot(\pi-\theta)\mathbf{n}_2}{|\mathbf{n}_{\text{wall}}-\cot(\pi-\theta)\mathbf{n}_2|} & \text{if } c<0\\
\frac{\mathbf{n}_{\text{wall}}+\cot(\pi-\theta)\mathbf{n}_2}{|\mathbf{n}_{\text{wall}}+\cot(\pi-\theta)\mathbf{n}_2|} & \text{if } c>0\\
\mathbf{n}_{\text{wall}} & \text{if } c=0
\end{cases}$$

Remarks:

- In 3D, the contact line (CL) is the 2D curve which represents the intersection of the free surface with the geometry surface. The vector $\mathbf{n}_2$ is orthogonal to the CL and lies in the tangent plane of the geometry surface.

- Since both ϕ and ψ are defined within a narrow band of the zero level set of ϕ, we can also define $\mathbf{u}^{\text{extend}}$ within a narrow band of the free surface.

- We use a first-order upwind procedure for solving (7.25). The direction of upwinding is determined from the extension velocity $\mathbf{u}^{\text{extend}}$. We solve (7.25) for $\tau = 0 \ldots \epsilon$.

- For viscous flows, there is a conflict between the no-slip condition (7.3) and the idea of a moving contact line. See [28, 40, 21] and the references therein for a discussion of this issue. We have performed numerical studies for axisymmetric oil spreading in water under ice [53] with good agreement with experiments. In the future, we wish to experiment with appropriate slip-boundary conditions near the contact line.

7.5.3 CLS Advection in General Geometries

For computational elements which contain only air and/or water, i.e., $V_{ij} = 1$, the CLS advection algorithm as described in Section 7.4 remains unchanged. For computational elements in which $0 < V_{ij} < 1$, we use the extension procedure described in Section 7.5.2 in order to initialize the level-set function ϕ and the volume fraction F in partial elements.

Remarks:

1. Since we only discretize the CLS advection step in full cells, we avoid stringent CFL conditions that exist in very small partial cells.

2. The discretization of the CLS advection step is not conservative. See [42, 20] for conservative "finite-volume-based" alternatives.

7.6 Adaptive Mesh Refinement

We describe the extension of the single-grid algorithm (Section 7.3) to an adaptive hierarchy of nested rectangular grids. For general references on adaptive mesh refinement (AMR) we refer the reader to [8, 6, 39, 1]. The ideal of AMR is that the solution procedure for a fixed uniform computational grid should remain unchanged. Adaptivity is achieved by dynamically overlaying successively finer grids in order to increase resolution of the free surface. The main modification to the single-grid

algorithm would be to supply boundary conditions at points where coarse grids and fine grids meet.

In Figure 7.3 we show an example of the grid structure used in AMR. The grid hierarchy is composed of different levels of refinement ranging from coarsest, $\ell = 0$, to finest, $\ell = \ell_{\max}$. The coarsest level, $\ell = 0$, covers the whole computational domain while successively higher levels, $\ell + 1$, lie on top of the level underneath them, level ℓ.

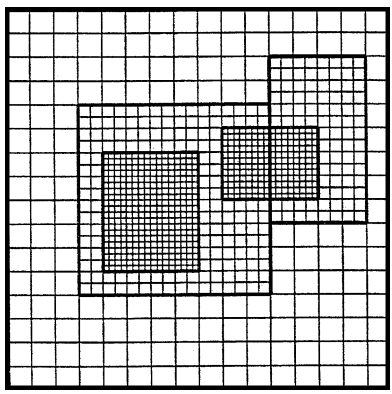


FIGURE 7.3
Diagram of grid structure used in adaptive mesh refinement (AMR). In this example, there are 3 levels. Level 0 has one 16 × 16 grid. Level 1 has two grids: a 16 × 16 grid and a 8 × 14 grid. Level 2 also has two grids: a 16 × 20 grid and a 16 × 12 grid. The refinement ratio between levels in this example is 2.

7.6.1 Time-Stepping Procedure for Adaptive Mesh Refinement

We use a "no-sub-cycling" time stepping procedure. In other words, the time step used on the finest level is the same as on all other levels. The details of our implementation can be found in [48]. An outline of our adaptive algorithm is as follows:

1. Given ϕ^n, F^n, $\boldsymbol{U}^n$ on coarse and fine levels.

2a. For coarse and fine levels, set $\mathbf{u}^n = 0$ in computational cells where $V_{ij} = 0$.

2b. Extend ϕ^n into regions where $V_{ij} = 0$. Repeat on all AMR levels (coarse and fine grids).

3. Repeat on coarse and then finer level(s):

$$\boldsymbol{V}^n = -[\nabla(\boldsymbol{U}\boldsymbol{U})]^n + \frac{\nabla \cdot 2\mu(\phi^n D^n)}{\rho(\phi^n)} - \frac{\gamma\kappa(\phi^n)\nabla H(\phi^n)}{\rho(\phi^n)} - \mathbf{G}\,.$$

4. Synchronize coarse and fine data for $\boldsymbol{V}^n$; i.e., average down fine grid data onto underlying coarse grids.

5. Update position of free surface on coarse and then finer level(s),

$$\begin{aligned} \phi^{n+1} &= \phi^n - \Delta t[\nabla \cdot (\mathbf{u}^{MAC}\phi)]^n \\ F^{n+1} &= F^n - \Delta t[\nabla \cdot (\mathbf{u}^{MAC}F]^n \end{aligned}$$

6. Update the velocity (pressure solve) on coarse and then finer level(s),

$$\boldsymbol{U}^{n+1} = \boldsymbol{U}^n + \Delta t\, P_{\rho(\phi^n)}(\boldsymbol{V}^n)\ . \tag{7.26}$$

7. Synchronize coarse and fine data for ϕ^{n+1}, F^{n+1}, and $\boldsymbol{U}^{n+1}$.

8. Reinitialize ϕ^{n+1}.

7.7 Results and Conclusions

7.7.1 Axisymmetric Jetting Convergence Study

We present a numerical convergence study for a 2D axisymmetric micro-scale jetting problem. In Figure 7.4 we display results where the effective fine-grid resolution is 64×1024. The diameter of the nozzle is 32 μ and the length of the nozzle is 70 μ. In Figure 7.5 we display the pressure profile that is applied at the base of the nozzle. This particular pressure impulse models the effect of a piezo-electric jetting device. In Table 7.1 we display the relative errors between computations in which we successively add finer resolved adaptive grids.

We remark that for this particular problem, "embedded boundary" methods seem ideal for modeling the flow. The ink-jet breaks up into two or more pieces which can be very difficult to model using a front tracking or boundary fitted grid approach for the free surface.

Table 7.1 Convergence Study Using the CLS Algorithm for the Axisymmetric Jetting of Ink

Grid	E(interface)
16×256	N/A
32×512	2.6E4
64×1024	1.7E4
$t = 70\ \mu$s, $\mu_l/\mu_g = 64$, $\rho_l/\rho_g = 816$.	

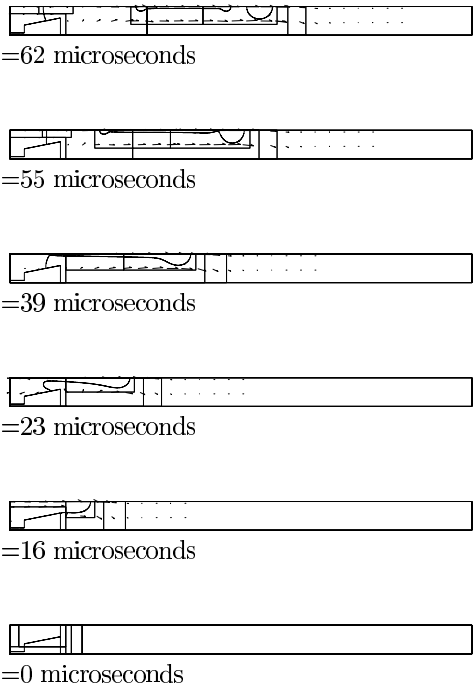


FIGURE 7.4
Axisymmetric jetting of ink. $\rho_w/\rho_a = 816$, $\mu_w/\mu_a = 64$. **Effective fine grid resolution is 64 $\times$ 1024.**

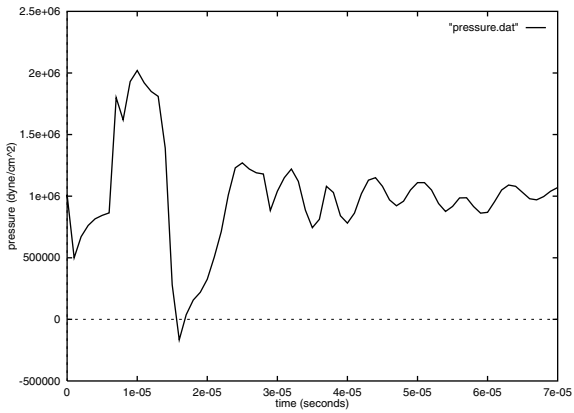


FIGURE 7.5
Pressure vs. time applied to base of nozzle for modeling piezo-electric device.

7.7.1.1 Validation of Contact Angle; Relaxation of Meniscus to Static Shape

In this section we initialize a horizontal meniscus within a 2D axisymmetric cylindrical nozzle. The parameters used are similar to the conditions that exist in an ink-jet nozzle. The initial length of the meniscus is 36 μ. Gas is on top and liquid is on the bottom. See Figure 7.6 for a diagram of initial conditions. The thick curved line in Figure 7.6 represents the expected final solution. The meniscus will relax to the

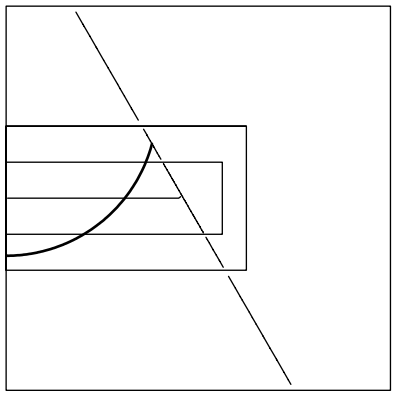


FIGURE 7.6
Initial free surface. Contact-angle boundary condition set at $\Theta = 45°$. Thick curved line represents expected static solution. Effective fine grid resolution 128 $\times$ 128.

shape that minimizes surface energy. If we assume zero gravity, then the static shape will be the part of a sphere that intersects the nozzle at the appropriate contact angle. In Figure 7.7 we compare our computed static shape with the expected shape when the contact angle is set at $\Theta = 45°$, the surface tension coefficient is 40 dyne/cm, and the viscosity of the liquid is 0.05 g/(cms). In Figure 7.8 we plot the kinetic energy vs. time as the meniscus relaxes to its static shape.

As a remark, we have performed the same test above using the 3D version of our code with similar results; although, for the 3D test, the interfacial thickness for the free surface has to be set at $\epsilon = 4\Delta x$ instead of $\epsilon = 3\Delta x$.

7.7.2 3D Ship Waves

In Figure 7.9, we show a volume rendering of adaptive computations of flow past a model Navy DDG 5415 ship. In Figure 7.10, we show the x-z slice of the ship flow. The Froude number for this problem is $F_r^2 = U^2/(gL) = 0.41$. We specify periodic boundary conditions in the x-direction and no-outflow boundary conditions in the y-direction and at the lower z-direction. The dimensionless length of the ship is 1 unit and the dimensions of our tank (in terms of dimensionless parameters) is $2 \times 0.5 \times 0.5$. At moderate to high speed, the turbulent flow along the hull of a ship and behind the stern is characterized by complex physical processes which involve breaking waves,

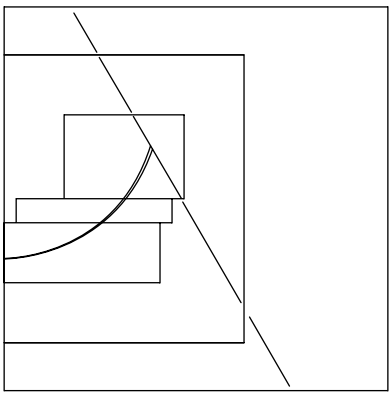


FIGURE 7.7
Free surface profile after initial meniscus is allowed to relax to steady state. Contact-angle boundary condition is set at $\Theta = 45°$. Thick lines represent expected static solution. Effective fine grid resolution is 128×128.

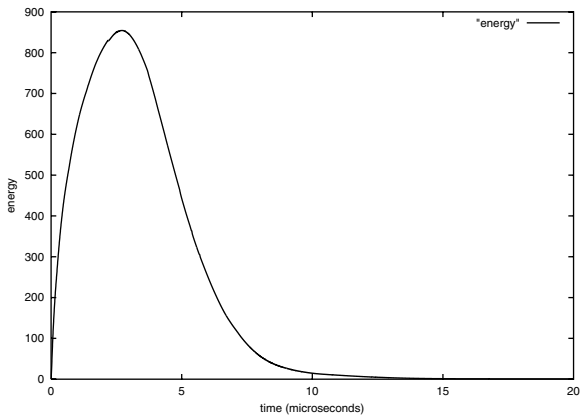


FIGURE 7.8
Kinetic energy vs. time for the relaxation of a meniscus to its final static shape. Effective fine grid resolution is 128×128.

air entrainment, free-surface turbulence, and the formation of spray [22]. Traditional numerical approaches to these problems, which use boundary-fitted grids, are difficult and time consuming to implement. Also, as waves steepen, boundary-fitted grids will break down unless ad hoc treatments are implemented to prevent the waves from getting too steep. At the very least, a bridge is required between potential-flow methods, which model limited physics, and more complex boundary-fitted grid

methods, which incorporate more physics, albeit with great effort and with limitations on the wave steepness. Cartesian-grid (embedded boundary) methods are a natural choice because they allow more complex physics than potential-flow methods and, unlike boundary-fitted methods, Cartesian-grid methods require minimal effort with no limitation on the wave steepness. Although Cartesian-grid methods are presently incapable of resolving the hull boundary layer, Cartesian-grid methods can model wave-breaking, free-surface turbulence, air entrainment, spray-sheet formation, and complex interactions between the ship hull and the free surface, such as transom-stern flows and tumblehome bows [49].

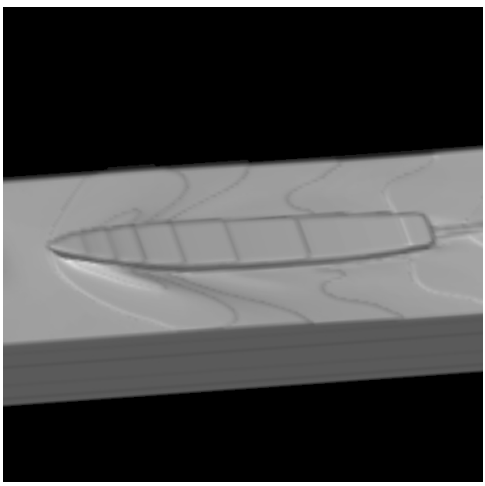


FIGURE 7.9
Flow past a model Navy DDG 5415 ship. Effective fine grid resolution is 256 $\times$ 64 $\times$ 64.

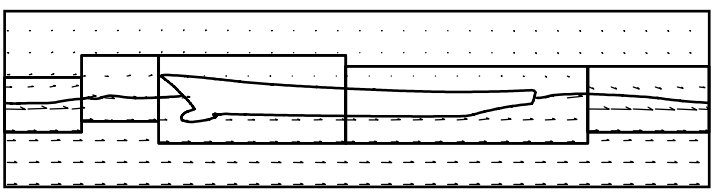
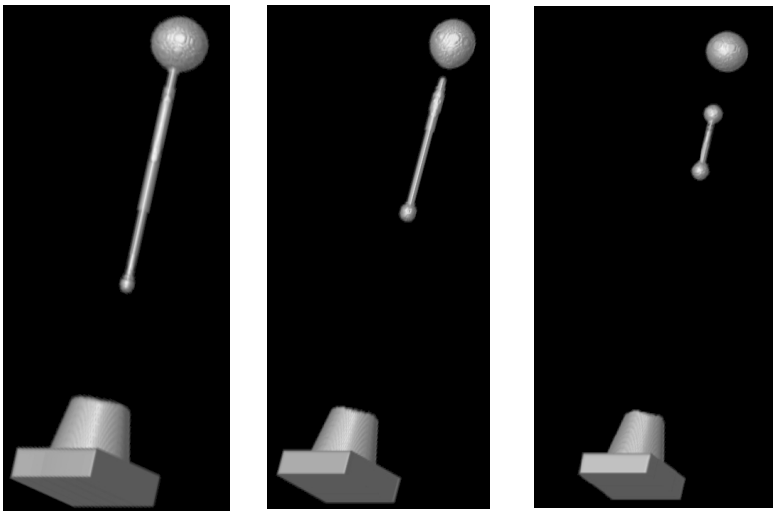


FIGURE 7.10
x-z slice of Flow past a model Navy DDG 5415 ship. Effective fine grid resolution is 256 $\times$ 64 $\times$ 64.



t=52 microseconds. t=69 microseconds. t=81 microseconds.

FIGURE 7.11
3D computation of jetting of ink. Solid parts are liquid. $\rho_w/\rho_a = 816$**,** $\mu_w/\mu_a = 64$**. Effective fine grid resolution is 32 × 32 × 256.**

References

[1] A.S. Almgren, J.B. Bell, P. Colella, L.H. Howell, and M. Welcome, A conservative adaptive projection method for the variable density incompressible Navier–Stokes equations, *J. Comput. Phys.,* **142**, 1–46, 1998.

[2] A.S. Almgren, J.B. Bell, P. Colella, and T. Marthaler, A Cartesian grid projection method for the incompressible euler equations in complex geometries, *SIAM J. Sci. Comput.,* **18**(5), 1289–1309, 1997.

[3] J.B. Bell, P. Colella, and H.M. Glaz, A second-order projection method for the incompressible Navier–Stokes equations, *J. Comput. Phys.,* **85**, 257–283, December 1989.

[4] J.B. Bell and D.L. Marcus, A second-order projection method for variable-density flows, *J. Comput. Phys.,* **101**, 334–348, 1992.

[5] J.B. Bell, J.M. Solomon, and W.G. Szymczak, A second-order projection method for the incompressible navier stokes equations on quadrilateral grids, in *9th AIAA Computational Fluids Dynamics Conference,* Buffalo, June 14–16, 1989.

[6] M.J. Berger and P. Colella, Local adaptive mesh refinement for shock hydrodynamics, *J. Comput. Phys.,* **82**, 64–84, 1989.

[7] M.J. Berger and A. Jameson, Automatic adaptive grid refinement for the Euler equations, *AIAA J.,* **23**, 561–568, 1985.

[8] M.J. Berger and J. Oliger, Adaptive mesh refinement for hyperbolic partial differential equations, *J. Comput. Phys.,* **53**, 484–512, 1984.

[9] M. Bern and D. Epstein, Mesh generation and optimal triangulation, Technical Report CSL-92-1, Xerox PARC, 1992.

[10] J.P. Best, The formation of toroidal bubbles upon the collapse of transient cavities, *J. Fluid Mech.,* **251**, 79–107, 1993.

[11] J.M. Boulton–Stone and Blake, Gas bubbles bursting at a free surface, *J. Fluid Mech.,* **254**, 437–466, 1993.

[12] J.U. Brackbill, D.B. Kothe, and C. Zemach, A continuum method for modeling surface tension, *J. Comput. Phys.,* **100**, 335–353, 1992.

[13] D.L. Brown, A finite volume method for solving the Navier–Stokes equations on composite overlapping grids, in *3rd International Conference on Hyperbolic Problems,* Uppsala, Sweden, 1990, LA-UR-90-3551.

[14] D.A. Caughey and A. Jameson, *AIAA J.,* **18**(11), 1281, 1980.

[15] Y.C. Chang, T.Y. Hou, B. Merriman, and S. Osher, Eulerian capturing methods based on a level set formulation for incompressible fluid interfaces, *J. Comput. Phys.,* **124**, 449–464, 1996.

[16] S. Chen, D. Johnson, and P. Raad, Velocity boundary conditions for the simulation of free surface fluid flow, *J. Comput. Phys.,* **116**, 262–276, 1995.

[17] S. Chen, D. Johnson, P. Raad, and D. Fadda, The surface marker and micro cell method, *International Journal for Numerical Methods in Fluids,* **25**, 749–778, 1997.

[18] G. Chessire and W.D. Henshaw, Composite overlapping meshes for the solution of partial differential equations, *J. Comput. Phys.,* **90**, 1–64, 1990.

[19] L.P. Chew, Guaranteed-quality triangular meshes. Technical Report TR 89-983, Cornell University Department of Computer Science, 1989.

[20] P. Colella, D.T. Graves, D. Modiano, E.G. Puckett, and M. Sussman, An embedded boundary/volume of fluid method for free surface flows in irregular geometries. In *Proceedings of the 3rd ASME/JSME Joint Fluids Engineering Conference,* number FEDSM99-7108, San Francisco, CA, 1999.

[21] R.G. Cox, The dynamics of the spreading of liquids on a solid surface. part 1. viscous flow, *J. Fluid Mech.,* **168**, 169–194, 1986.

[22] D.G. Dommermuth, G.E. Innis, T. Luth, E.A. Novikov, E. Schlageter, and J.C. Talcott, Numerical simulation of bow waves, in *Proceedings of the Twenty-Second Symposium on Naval Hydro.,* pages 508–521, Washington, D.C., 1998.

[23] R. Fedkiw, T. Aslam, and S. Xu, The ghost fluid method for deflagration and detonation discontinuities, *J. Comput. Phys.,* **154**, 393–427, 1999.

[24] D. Goldstein, R. Handler, and L. Sirovich, Modeling a no-slip flow boundary with an external force field, *J. Comput. Phys.,* **105**, 354–366, 1993.

[25] D. Gueyffier, J. Li, A. Nadim, R. Scardovelli, and S. Zaleski, Volume of fluid interface tracking with smoothed surface stress methods for three-dimensional flows, *J. Comput. Phys.,* **152**, 423–456, 1999.

[26] B.T. Helenbrook, L. Martinelli, and C.K. Law, A numerical method for solving incompressible flow problems with a surface of discontinuity, *J. Comput. Phys.,* **148**, 366–396, 1999.

[27] C.W. Hirt, *Flow-3D Users Manual,* 1988. Flow Sciences, Inc.

[28] L.M. Hocking and A.D. Rivers, The spreading of a drop by capillary action, *J. Fluid Mech.,* **121**, 425–442, 1982.

[29] T.Y. Hou, J.S. Lowengrub, and M.J. Shelley, Removing the stiffness from interfacial flows with surface tension, *J. Comput. Phys.,* **114**, 312, 1994.

[30] W. Huang, Y. Ren, and R. Russell, Moving mesh methods based on moving mesh partial differential equations, *J. Comput. Phys.,* **113**, 2, 1994.

[31] D. Jacqmin, An energy approach to the continuum surface method, Number AIAA-96-0858, Reno, NV, 1996. Presented at the 34th Aerospace Sciences Meeting.

[32] A. Jameson, T.J. Baker, and N.P. Weatherill, Number AIAA Paper 86-0103, Reno, NV, 1986. AIAA 25th Aerospace Meeting.

[33] H. Johansen and P. Colella, A Cartesian grid embedded boundary method for Poisson's equation on irregular domains, *J. Comput. Phys.,* **147**(1), 60–85, 1998.

[34] Y. Kurihara, G. Tripoli, and M. Bender, Design of a movable nested-mesh primitive equation model, *Monthly Weather Review,* **107**, 239–249, 1979.

[35] F. Liu, S. Ji, and G. Liao, An adaptive grid method and its application to steady Euler flow calculations, *SIAM J. Sci. Comput.,* **20**(3), 811–825, 1998.

[36] R. Lohner, K. Morgan, and O.C. Zienkiewicz, Adaptive grid refinement for the compressible Euler equations, in I. Babuska and O.C. Zienkiewicz, eds., *Accuracy Estimates and Adaptivity for Finite Elements,* page 281, Wiley, New York, 1986.

[37] R.L. Lohner and J.D. Baum, Numerical simulation of shock interaction with complex geometry structures using a new h-refinement scheme on unstructured grids. Technical report, 28th AIAA Aerospace Sciences Meeting, 1990.

[38] D.J. Mavriplis, *AIAA J.,* **28**(2), 213, 1990.

[39] M.L. Minion, A projection method for locally refined grids, *J. Comput. Phys.,* 127, 1996.

[40] C.G. Ngan and E.B. Dussan V., On the dynamics of liquid spreading on solid surfaces, *J. Fluid Mech.,* **209**, 191–226, 1989.

[41] S. Osher and S. Chakravarthy, Upwind schemes and boundary conditions with applications to Euler equations in general geometries, *J. Comput. Phys.,* **50**(3), 447–481, 1982.

[42] R.B. Pember, J.B. Bell, P. Colella, W.Y. Crutchfield, and M.L. Welcome, An adaptive Cartesian grid method for unsteady compressible flow in irregular regions, *J. Comput. Phys.,* **120**, 278–304, 1995.

[43] C.S. Peskin, Numerical analysis of blood flow in the heart, *J. Comput. Phys.,* **25**, 220–252, 1977.

[44] E.G. Puckett, A.S. Almgren, J.B. Bell, D.L. Marcus, and W.G. Rider, A high-order projection method for tracking fluid interfaces in variable density incompressible flows, *J. Comput. Phys.,* **130**, 269–282, 1997.

[45] P. Raad, S. Chen, and D. Johnson, The introduction of micro cells to treat pressure in free surface fluid flow problems, *J. Fluids Eng.,* **117**, 683–690, 1995.

[46] W.J. Rider, D.B. Kothe, S.J. Mosso, J.H. Cerutti, and J.I, Hochstein, Accurate solution algorithms for incompressible multiphase flows, *AIAA paper 95-0699,* October 30 1994.

[47] M.S. Shephard, Parallel automated adaptive procedures for unstructured meshes, Technical Report R-907, AGARD, 1995.

[48] M. Sussman, A. Almgren, J. Bell, P. Colella, L. Howell, and M. Welcome, An adaptive level set approach for incompressible two-phase flows, *J. Comput. Phys.,* **148**, 81–124, 1999.

[49] M. Sussman and D.G. Dommermuth, The numerical simulation of ship waves using Cartesian grid methods, In *Proceedings of the Twenty-Third Symposium on Naval Hydrodynamics,* Val-De-Reuil, France, September 2000, to appear.

[50] M. Sussman and E.G. Puckett, A coupled level set and volume of fluid method for computing 3D and axisymmetric incompressible two-phase flows, *J. Comp. Phys.,* **162**, 301–337, 2000.

[51] M. Sussman and P. Smereka, Axisymmetric free boundary problems, *J. Fluid Mech.,* **341**, 269–294, 1997.

[52] M. Sussman, P. Smereka, and S.J. Osher, A level set approach for computing solutions to incompressible two-phase flow, *J. Comput. Phys.,* **114**, 146–159, 1994.

[53] M. Sussman and S. Uto, Computing oil spreading underneath a sheet of ice. Technical Report CAM Report 98-32, University of California, Los Angeles, July 1998.

[54] O. Tatebe, The multigrid preconditioned conjugate gradient method, in *6th Copper Mountain Conference on Multigrid Methods,* Copper Mountain, CO, April 4–9 1993.

[55] J.F. Thompson, Z.U.A. Warsi, and C.W. Mastin, *J. Comput. Phys.,* 47, 1982.

[56] T.M. Tsai and M. Miksis, Dynamics of a drop in a constricted capillary tube, *J. Fluid Mech.,* **274**, 197–217, 1994.

[57] H.S. Udaykumar, H.C Kan, W. Shyy, and R. Tran-Son-Tay, Multiphase dynamics in arbitrary geometries on fixed Cartesian grids, *J. Comput. Phys.,* **137**(2), 366–405, 1997.

[58] H.S. Udaykumar, M.M. Rao, and W. Shyy, Elafint — a mixed Eulerian-Lagrangian method for fluid flows with complex and moving boundaries, *Int. J. Numer. Meths. Fluids.,* **22**(8), 691–704, 1996.

[59] S.O. Unverdi and G. Tryggvason, A front-tracking method for viscous, incompressible, multi-fluid flows, *J. Comput. Phys.,* **100**, 25–37, 1992.

[60] J.M. Waldvogel and D. Poulikakos, Solidification phenomena in picoliter size solder droplet deposition on a composite substrate, *Int. J. Heat Mass Transfer,* **40**, 295–309, 1997.

[61] J.M. Waldvogel, D. Poulikakos, D.B. Wallace, and R. Marusak, Transport phenomena in picoliter size solder droplet dispension, *Transactions of the ASME, J. Heat Transfer,* **118**, 148–156, 1996.

[62] D. De Zeeuw and K.G. Powell, An adaptively-refined Cartesian mesh solver for the euler equations, *J. Comput. Phys.,* **104**, 55, 1992.

[63] P.A. Zegeling, *Moving Grid Methods,* Utrecht University Press, 1992.

Chapter 8

The Solution of Steady PDEs on Adjustable Meshes in Multidimensions Using Local Descent Methods

M.J. Baines

8.1 Introduction

The method of lines (MOL) is a technique for solving partial differential equations (PDEs) in which the parameters of a space discretization of the PDE are advanced in time through the solution of an ordinary differential equation (ODE) system, normally using a software package. The method can be applied to time-dependent or steady PDES; in the latter case via convergence in pseudotime. Iterative procedures are in any case necessary for steady nonlinear PDEs. The MOL has reached a high degree of sophistication, as evidenced elsewhere in this volume, and has produced impressive results.

The purpose of this chapter is to discuss the introduction of mesh movement for steady PDEs in multidimensions, using mesh locations as additional parameters. In this way we may seek an optimal mesh at the same time as finding a converged solution on that mesh. When the mesh locations are included in the parameters of the space discretization, an extended system of ordinary differential equations (ODEs) or differential algebraic equations (DAEs) is normally obtained which includes both mesh and solution parameters in a coupled way. Integration of these equations may then be carried out using the MOL, although there is a wide variety of approaches.

Adaptation via mesh movement is known as r-refinement. The underlying idea is that the numerical solution of PDEs, particularly those that have rough solutions, should use all available resources and the mesh is one of these resources. In r-refinement the solution is adaptively improved by mesh relocation, normally using a fixed amount of resource. In its simplest form the number of nodes remains unchanged and, provided there is no change in connectivity, there is a fixed data structure.

There are, however, a number of special difficulties with algorithms which involve mesh movement. First, there is generally no information within the problem about

how the mesh should be moved and so prescription of the movement is very much in the hands of the algorithm designer. Although there is sometimes an obvious choice for the mesh velocity, for example in Lagrangian fluid codes where the mesh is moved with the velocity of the fluid, there is in general no physically identifiable choice. Second, many PDEs are derived from the application of a physical principle, for example conservation of mass, in a fixed frame of reference. If the frame of reference moves, the PDE must be modified and may not retain the physical properties on which it is based. Third, mesh movement algorithms are essentially nonlinear and exhibit a high degree of complexity.

Another major difficulty is the possibility of mesh tangling. In one dimension this simply means node overtaking. In two dimensions, to take an example, a moving triangulation in which a node of a triangle crosses an opposite side may lead to the breakdown of a method, either because of singularity at the point of crossing or because of the inability of the method to function on an invalid triangulation. Therefore, constraints are often built into a method to avoid tangling.

There are two main techniques for the movement of nodes. The most well-known technique in this area is that of equidistribution, but we shall be mostly concerned with techniques that use optimization, since they are valid in multidimensions. The two techniques overlap in some formulations. Equidistribution is a one-dimensional concept although there have recently been some significant advances in generalizing the idea to two dimensions. Links with an approximate form of multidimensional equidistribution are described in the penultimate section of the chapter. We shall consider two kinds of functional to be minimized, normally associated with two different types of PDE. The first is a class of variational principles which generate PDEs of Euler–Lagrange type, which of their nature are of second order. The second is the L_2 norm of the residual associated with a discretization of the PDE, which can be used for first-order equations and systems. Both finite-element and finite-volume discretizations will be discussed.

The existence of a functional allows at least two different approaches to generate solutions (and meshes). In the more standard approach the full (augmented) ODE system of normal equations can be solved by the MOL, which we shall refer to as the global approach. In the other approach we use a descent method on the functional, which can be implemented in a local manner (node by node) sweeping through the mesh, which we shall refer to as the local approach.

An early moving-mesh method was the moving finite element (MFE) method [1, 5, 4], which uses piecewise linear finite elements and generates the augmented ODE system from minimization of the L_2 norm of the residual of the PDE over the time derivatives of both the nodal positions and the solution parameters. The method is a Galerkin method in which the test functions span the space of the Lagrangian time derivatives. The method is truly multidimensional and has had some notable successes, particularly for parabolic problems. However, it also showed up one of the difficulties in using piecewise linear approximation in a moving node context, namely an indeterminacy when the solution and the mesh are simultaneously trying to represent a linear manifold. This problem occurs when the local curvature of the

solution manifold vanishes. For this reason (as well as the difficulty of mesh tangling), regularizing terms were added to the L_2 norm in the MFE method with a number of adjustable parameters. However, the effectiveness of the method was found to depend crucially on the manipulation of these parameters and the method has not found favor with practitioners.

Nevertheless, for a class of *steady* problems derived from variational principles, the MFE method gives an optimal solution and the time-dependent form may be used as an iterative method to drive solutions to steady state (see Section 8.2).

We commence this chapter by considering the role of the MFE method in the context of optimization. Although originally formulated as an L_2 minimization, the method is not an optimization method in the usual sense but simply an extended weak form of the PDE. On the other hand, for steady equations of variational type, it has been shown in [2] that the weak forms correspond to the optimization of a minimization principle in a discrete space.

The MFE philosophy incorporates a global MOL approach to the solution of the normal equations. An example of this approach for the steady MFE method is given in [2]. However, if a functional is available, there is the alternative of sweeping through the mesh using local descent methods. This is the central theme of this chapter. Such an approach to optimization using minimization principles is given in [3] and is described in Section 8.3. A possible finite volume formulation is also proposed.

Section 8.4 is devoted to least-squares minimization, of particular relevance to first-order equations and systems. The least squares MFE (LSMFE) method [11] and a corresponding finite-volume method [7] are described, which use global and local approaches to the solution procedure, respectively. For the important case of conservation laws, the finite-volume procedure may be extended to systems [15], and a description of this technique forms Section 8.5.

Finally, there is a section on the links with equidistribution (in one dimension) and approximate equidistribution (in higher dimensions), and a summary section.

8.2 Moving Finite Elements

The moving finite element (MFE) method [1, 5, 4] for the time-dependent PDE

$$u_t = Lu \ , \tag{8.1}$$

where u is a function of $\underline{x}$ and t, and L is a space operator, is a semi-discrete moving-mesh, finite-element method in which the node locations are allowed to depend on time. It is based on two weak forms of the PDE which can be derived from the minimization of the L_2 norm of the residual over the time derivatives of the parameters.

The approximate solution U is an explicit function of the $\underline{X}_j(t)$ (the nodal positions) of the form

$$U = \sum_j U_j \psi_j(\underline{x}) \tag{8.2}$$

where U_j are coefficients and the $\psi_j(\underline{x})$ are piecewise linear-basis functions. Using the result

$$\frac{\partial U}{\partial \underline{X}_j} = (-\nabla U)\,\psi_j \tag{8.3}$$

(see, e.g., [5]) the derivative of U with respect to t becomes

$$\begin{aligned} U_t = \frac{\partial U}{\partial t}|_{\text{moving}\underline{X}} &= \frac{\partial U}{\partial t}|_{\text{fixed}\underline{X}} + \sum_j \frac{\partial U}{\partial \underline{X}_j}.\frac{d\underline{X}_j}{dt} \\ &= \frac{d\hat{U}}{} dt + \sum_j (-\nabla U)\,\psi_j.\frac{d\underline{X}_j}{dt} \\ &= \dot{U} - \nabla U.\,\dot{\underline{X}} \end{aligned} \tag{8.4}$$

where the independent $\hat{U}$ and $\underline{X}$ functions have time derivatives

$$\dot{U} = \frac{d\hat{U}}{dt} = \sum_j \frac{dU_j}{dt}\psi_j\,, \qquad \dot{\underline{X}} = \frac{d\underline{X}}{dt} = \sum_j \frac{d\underline{X}_j}{dt}\psi_j \tag{8.5}$$

which are taken to be continuous functions, corresponding to the evolution of a continuous piecewise linear approximation.

From (8.1) and (8.4) minimization of the square of the L_2 residual $\|U_t - LU\|_{L_2}^2$ over the coefficients $\dot{U}_j, \dot{\underline{X}}_j$ then takes the form

$$\min_{\dot{U}_j, \dot{X}_j} \left\| \dot{U} - \nabla U.\,\dot{\underline{X}} - LU \right\|_{L_2}^2 \tag{8.6}$$

and, using (8.5), gives the MFE or extended Galerkin equations

$$\left\langle \psi_j, \dot{U} - \nabla U.\,\dot{\underline{X}} - LU \right\rangle = 0 \tag{8.7}$$

$$\left\langle (-\nabla U)\,\psi_j, \dot{U} - \nabla U.\,\dot{\underline{X}} - LU \right\rangle = 0 \tag{8.8}$$

Substituting for $\dot{U}$ and $\dot{\underline{X}}$ from (8.5) gives a nonlinear system of ODEs for U_j and $\underline{X}_j$ containing an extended MFE mass matrix. The system may be solved globally for the unknowns U_j and $\underline{X}_j$ by a stiff ODE package, as in the MOL.

The basic method has intrinsic singularities, however. If the gradients ∇U have components whose values are equal in adjacent elements (dubbed "parallelism"

in [1]), the system of equations (8.7)/(8.8) becomes singular and must be regularized in some way. If the area of an element vanishes (i.e., a triangle becomes degenerate) the system again becomes singular and special action is required. In Miller's MFE method, penalty functions are added to the L_2 norm of the residual in (8.6) (see [1, 4, 5]).

Although a full understanding of the MFE method is incomplete, in the steady limit the resulting mesh has a significant optimal property. We now consider this limit.

8.2.1 MFE in the Steady-State Limit

In many cases the MFE method may be used to generate weak forms for the approximate solution of the *steady* PDE

$$Lu = 0 \tag{8.9}$$

by driving the MFE solutions to convergence in pseudotime, although not always. For scalar first-order PDEs, the MFE method is known to move the nodes with characteristic speeds [5] which do not generally settle down to a steady state.

From (8.6) the MFE method in the steady case implements the minimization

$$\min_{\dot{U}_j, \dot{\underline{X}}_j} \|LU\|_{L_2}^2 \tag{8.10}$$

and the steady-state solution satisfies the weak forms

$$\left\langle \begin{pmatrix} 1 \\ -\nabla U \end{pmatrix} \psi_j, LU \right\rangle = 0 \,. \tag{8.11}$$

Although $\dot{U}$ and $\dot{\underline{X}}$ no longer appear in LU, the minimization is over the span of their time derivatives] which is the space spanned by the functions $\{\psi_j, (-\nabla U)\,\psi_j\}$.

In order to describe the optimal property of the steady MFE method, we recall the origin of PDEs of Euler–Lagrange type.

8.2.2 Minimization Principles and Weak Forms

A standard result in classical analysis is that minimization of the functional

$$J(F) = \int F(u, \nabla u) d\underline{x} \tag{8.12}$$

over a suitable class of functions yields the PDE

$$Lu = -\frac{\partial F}{\partial u} + \nabla . \frac{\partial F}{\partial \nabla u} = 0 \,. \tag{8.13}$$

By a similar argument, minimization of the functional (8.12) over the finite dimensional space spanned by the $\{\psi_j(\underline{x})\}$ [i.e., over approximations of the form (8.2)] on a fixed mesh yields the weak form

$$\left\langle \psi_j, \frac{\partial F}{\partial U} \right\rangle + \left\langle \nabla \psi_j, \frac{\partial F}{\partial \nabla U} \right\rangle = 0 \,. \tag{8.14}$$

8.2.3 An Optimal Property of the Steady MFE Equations

It has been shown in [2] that minimization of the functional (8.12) over functions in the MFE approximation space, spanned by $\psi_j(\underline{x})$, $-\nabla U)\psi_j(\underline{x})$, yields the steady MFE equations (8.11). Hence, the steady MFE equations provide an optimal U and $\underline{X}$ for variations of the functional (8.12) within this space.

These weak forms are

$$\frac{\partial}{\partial U_j}\int F(U,\nabla U)d\underline{x} = \left\langle \psi_j, \frac{\partial F}{\partial U}\right\rangle + \left\langle \nabla\psi_j, \frac{\partial F}{\partial \nabla U}\right\rangle = 0 \tag{8.15}$$

as in (8.14) and

$$\frac{\partial}{\partial \underline{X}_j}\int F(U,\nabla U)d\underline{x} = \left\langle \psi_j, \frac{\partial F}{\partial \underline{x}}\right\rangle + \left\langle \nabla\psi_j, \left(F - \nabla U\cdot\frac{\partial F}{\partial \nabla U}\right)\right\rangle = 0 \tag{8.16}$$

where the identity

$$\nabla\cdot\left(F - \nabla U\cdot\frac{\partial F}{\partial \nabla U}\right) = \frac{\partial F}{\partial \underline{x}} + \frac{\partial F}{\partial U}\nabla U - \left(\nabla\cdot\frac{\partial F}{\partial \nabla U}\right)\nabla U \tag{8.17}$$

has been used to derive a form of (8.16) which is formally suitable for piecewise linear approximation. In carrying out the integration by parts to arrive at (8.16) we have used the fact that the continuous piecewise linear finite element basis function ψ_j vanishes on the boundary of the patch. The result in [2] is that (8.15) and (8.16) are identical to the weak forms (8.11).

It may be possible to use the time-dependent MFE method as an iterative procedure to generate locally optimal meshes in the steady state. This approach has been used in [2] to generate optimal solutions with variable nodes to a number of examples of PDEs of Euler–Lagrange type. A partially regularized form of the MFE method is used in order to avoid singular behavior and a global MOL solver employed to extract the solution. For further details see [2]. However, since the iteration need not be time accurate, the MFE mass matrix may, if desired, be replaced by any positive definite matrix.

We now come to the central theme of this chapter, which is to consider the role of descent methods in generating solutions of problems of this type using a local approach.

8.3 A Local Approach to Variational Principles

Since minimization principles provide a functional to monitor and reduce, it is possible to take advantage of standard optimization procedures in generating local minima. For example, procedures based on descent methods give the freedom to use a local approach to iteration, which significantly reduces the complexity of problems involving mesh movement. First we recall the nature of descent methods.

8.3.1 Descent Methods

Descent methods are based upon the property that the first variation of a functional J with respect to a vector variable $\underline{Y}$,

$$\delta J = \frac{\partial J}{\partial \underline{Y}} \delta \underline{Y} = \underline{g}^T \delta \underline{Y} \tag{8.18}$$

say, is negative when

$$\delta \underline{Y} = -\tau \underline{g} = -\tau \frac{\partial J}{\partial \underline{Y}} \tag{8.19}$$

for a sufficiently small positive relaxation parameter τ, and therefore reduces J. Choice of τ is normally governed by a line search or a local quadratic model.

The left-hand side of (8.19) may be preconditioned by any positive definite matrix. The Hessian gives the full Newton approach but may be approximated in various ways.

In the present context of r-adaptivity, a local approach is advantageous which consists of updating the unknowns one node at a time (scalar $\underline{Y}$), using only local information. Moreover, U_j and $\underline{X}_j$ may be updated sequentially, which permits close control of the mesh movement. The updates may be carried out in a block (Jacobi iteration) or sequential (Gauss–Seidel) manner. Descent methods of this type have been used by Tourigny and Baines [6] and Tourigny and Hulsemann [3] in the L_2 case and by Roe [7] and Baines and Leary [8] in the discrete case.

First we mention a local approach to L_2 best fits with adjustable nodes.

8.3.2 A Local Approach to Best Fits

A minimization based on a local approach was used in [9] to generate algorithms to determine best *discontinuous* piecewise constant and piecewise linear L_2 fits to a given function in one or two dimensions, with adjustable nodes. The convergence of the one-dimensional algorithm was subsequently investigated in [6] and the method shown to reduce the L_2 norm of the residual error monotonically. The two-dimensional algorithm was modified in [6] and a procedure for improving the connectivity introduced. Convergence of the method was also considered in [6] for successively (globally) refined meshes.

A special feature of these algorithms is their local nature, the nodal and solution updates being carried out one node at a time within sweeps through the mesh. This approach not only reduces the complexity of the problem but also allows for mesh tangling to be avoided relatively easily using a limiter (see, e.g., [19]). Moreover, owing to the existence of a functional to minimize, edge swapping and node removal are readily incorporated.

We now turn to the approximation of PDEs and consider minimization principles using the local approach.

8.3.3 Direct Optimization Using Minimization Principles

An early attempt to include mesh adaptation into a minimization principle was due to Delfour et al. [10], who sought a finite-element solution with free nodes for a variational formulation of an elliptic PDE but found significant problems with the complexity of the equations and with mesh tangling.

More recently, an iterative algorithm with variable nodes for the finite-element solution of minimization problems has been described in [3] using the local approach. The criterion is that the mesh should be such that a variational "energy functional" evaluated at the finite-element approximation is reduced. Each node is treated separately in sweeping through the mesh. The nodal positions are updated by a steepest descent procedure, during which a sequence of *local* finite-element problems is solved, each involving very few degrees of freedom. The order of sweeping through the mesh is based on the size of the local residuals. The method is applied to a variety of minimization principles in two dimensions.

We describe the essence of the method here. A finite-element approximation U is sought to optimize a convex energy functional of the form (8.12) in a subspace V_h such that

$$J(U) = \min_{V \in V_h(\Delta)} J(V) \tag{8.20}$$

where V_h is the set of piecewise linear functions defined on a triangulation Δ which is allowed to deform. The minimization is conceived in terms of solving a sequence of local problems on patches of triangles $\{T_j\}$ surrounding node j (see Figure 8.1) and sweeping through the mesh. Each local approximation $\hat{U}$ is computed using the

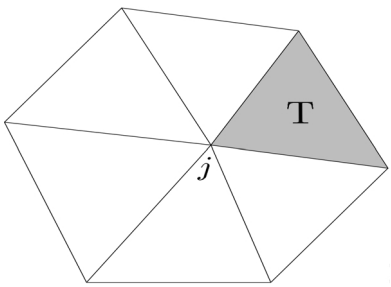


FIGURE 8.1
A local patch of elements surrounding node j.

normal equations

$$\int_{\{T_j\}} \left(\frac{\partial F}{\partial \hat{U}}(\hat{U}, \nabla \hat{U}) \psi_j + \frac{\partial F}{\partial \hat{\nabla} U}(\hat{U}, \nabla \hat{U}) \cdot \nabla \psi_j \right) d\underline{x} = 0 \tag{8.21}$$

$\forall \psi_j \in V_h$ [cf. (8.14)]. In a local patch such as that in Figure 8.1 the computation of $\hat{U}$ at node j on a fixed mesh involves the determination of only a single unknown, which can be carried out cheaply using only local information on the patch.

New local nodal positions $\hat{\underline{X}} = \underline{X}_j^{\text{new}}$ are sought within the iteration, with the corresponding solution $\hat{U} = U^{\text{new}}$, such that $J(U)$ is reduced. A steepest-descent method is used. Thus, if $\underline{X}_j$ is an interior node, a new mesh location is sought along the line given by

$$\underline{X}_j^{\text{new}} = \underline{X}_j - \tau \frac{\partial J}{\partial \underline{X}_j} , \tag{8.22}$$

where the relaxation factor τ is chosen by a line search. Although this requires the solution of a sequence of finite-element problems of the form (8.21), these are small, local problems.

The sequence of nodal updates is carried out in a Gauss-Seidel manner. Edge swapping is interleaved with the grid-movement algorithm, an edge being swapped if it leads to a lower energy. *Global* mesh refinement is also included in the algorithm, refinement taking place after the algorithm for the current number of nodes had converged. That is, starting from a coarse mesh the algorithm is used to optimize the mesh; this optimal mesh is then uniformly refined to provide the starting mesh for the next refinement level. The possible occurrence of degenerate triangles is overcome by the use of a node-deletion algorithm. Convergence to the "global" solution relies on the sweeps through the mesh.

Convergence rates in a test problem on Laplace's equation involving a re-entrant corner showed that, whereas convergence on a uniform mesh was sub-optimal, the approximation on the adapted meshes constructed in this way converged at the optimal rate. Figure 8.2 shows two of the meshes obtained using the algorithm with successive globally refined meshes. There is an exact solution of this problem of the form $r^{2/7} \sin(\frac{2}{7})\theta$ (for full details see [3]).

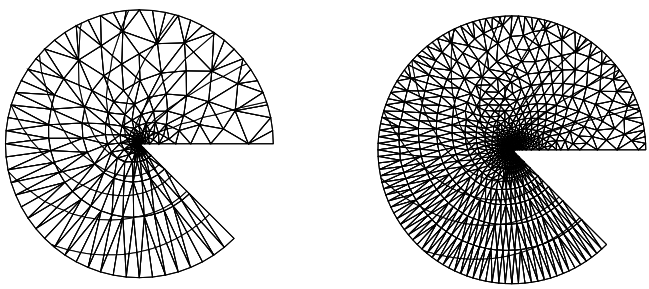


FIGURE 8.2
Two meshes for the re-entrant corner problem.

8.3.4 A Discrete Variational Principle

A discrete form of (8.12) (obtained by quadrature) is

$$J_d(F) = \sum_T S_T \overline{F(U, \nabla U)}_T \tag{8.23}$$

where the suffix T runs over all the triangles of the mesh, S_T is the area of triangle T, and the overbar denotes the average value of the argument over the vertices of the triangle T.

Differentiation of (8.23) with respect to U_j gives (see [22])

$$\frac{\partial J_d}{\partial U_j} = \sum_{T_J} \left[\frac{1}{3} S_T \left(\frac{\partial F}{\partial U} \right)_j + \frac{\partial \overline{F}}{\partial (\nabla U)_T} \cdot \underline{n}_j \right] \tag{8.24}$$

where $\underline{n}_j$ is the inward normal to the side opposite node j scaled by the length of that side. Setting this gradient to zero gives the finite volume weak form corresponding to the finite-element weak form (8.14).

Differentiation with respect to $\underline{X}_j$ gives (see [17])

$$\begin{aligned} \frac{\partial I_d}{\partial \underline{X}_j} = \sum_{T_j} \Bigg[& \frac{1}{3} S_T \left(\frac{\partial \overline{F}}{\partial \underline{x}} \right)_T + \left(-\frac{\partial \overline{F}}{\partial U_y}, \frac{\partial \overline{F}}{\partial U_x} \right)_j \Delta U_j \\ & + \sum_{T_j} \left(\frac{\partial \overline{F}}{\partial \nabla U} . \nabla U - F \right)_T \underline{n}_j \Bigg] . \end{aligned} \tag{8.25}$$

Setting this gradient to zero gives the companion weak form to (8.24), corresponding to the second finite-element weak form (8.16).

We may approximate $(\nabla U)_T$ by the gradient of the linear interpolation between the corner values of U in the triangle T, given by (see [13])

$$(\nabla U)_T = \left(\frac{-\sum' U \Delta Y}{\sum' X \Delta Y}, \frac{\sum' U \Delta X}{-\sum' Y \Delta X} \right) = \left(\frac{\sum' Y \Delta U}{\sum' X \Delta Y}, \frac{-\sum' X \Delta U}{\sum' Y \Delta X} \right) \tag{8.26}$$

where the sums $\sum'$ run over the vertices of the triangle T and ΔX, ΔY, ΔU denote the increments in the values of X, Y, U taken counterclockwise across the side of T *opposite* the corner concerned (see Figure 8.1). This is the same as the piecewise linear approximation used in the finite-element case. In the same notation as above, the area of the triangle T is

$$S_T = \frac{1}{2} \sum{}' X \Delta Y = -\frac{1}{2} \sum{}' Y \Delta X \, . \tag{8.27}$$

The expressions (8.24) and (8.25) with ∇U given by (8.26) provide gradients for a steepest-descent method for minimizing (8.23). Examples of this approach will be given in the next section.

We turn now to least-squares methods, which extend the applicability of the techniques already described to first-order equations and systems.

8.4 Least-Squares Methods

The steady MFE and local minimization techniques are valid and useful if there exists a minimization principle for the PDE, but are not available in other cases where no such principle exists, in particular for first-order equations and systems. (Euler–Lagrange equations, by their nature, are of at least second order.) However, the same minimization techniques can also be applied to least-squares methods, where the "energy functional" is the square of the norm of the residual. We describe two such methods, one of MFE type exploiting the optimal property of the steady MFE method described in Section 8.2.3 and the other arising from a finite-volume approach, similar to that in Section 8.3.4.

In this section, we shall assume that L is a first-order space operator, depending on $\underline{x}$, u, and ∇u only.

8.4.1 Least-Squares Moving Finite Elements

Although the L_2 norm (8.6) was minimized in formulating the MFE method described in Section 8.2, this minimization is only carried out over the velocities $\dot{U}_j$ and $\dot{\underline{X}}_j$ and is thus not a true minimization at the fully discrete level. Variations in U_j and $\underline{X}_j$ are treated as independent of those in $\dot{U}_j$ and $\dot{\underline{X}}_j$ and are ignored. It can therefore be seen from (8.6) that the method is simply a linear least-squares problem, generating the weak forms (8.7) and (8.8). It is more useful to say that the minimization is carried out in the space spanned by the basis functions $\{\psi_j, (-\nabla U)\psi_j\}$.

By contrast, a full minimization of the L_2 norm in (8.10) may be carried out in the steady case over the nodal coordinates $\underline{X}_j$ and the coefficients U_j. This is the approach of the recent least squares moving finite element (LSMFE) method [11]. This is a nonlinear least-squares problem, so only a local minimum can be expected.

Consider then the minimization of (8.10) over these parameters, which leads to the two weak forms

$$\left\langle LU, \left(\frac{\partial}{\partial U_j} LU\right)\right\rangle = \left\langle LU, (-\nabla U)\,\frac{\partial}{\partial U_j}\,(LU)\right\rangle + \oint \frac{1}{2}\,(LU)^2\,\psi_j \underline{\hat{n}}\,ds = 0 \tag{8.28}$$

($\underline{\hat{n}}$ is the unit normal) which may be written

$$\left\langle \frac{\partial\,(LU)^2}{\partial U}, \psi_j \right\rangle + \left\langle \frac{\partial\,(LU)^2}{\partial \nabla U}, \nabla\psi_j \right\rangle = 0 \tag{8.29}$$

and

$$\left\langle \frac{\partial\,(LU)^2}{\partial \underline{x}}, \psi_j \right\rangle + \left\langle \left((LU)^2 - (\nabla U)\,.\frac{\partial\,(LU)^2}{\partial \nabla U} \right), \nabla\psi_j \right\rangle = 0\,, \tag{8.30}$$

where the identity (8.17) has been used with $F = \frac{1}{2}(LU)^2$.

Referring back to (8.15) and (8.16) we see that Equations (8.29) and (8.30) are the *steady* MFE equations for the PDE

$$0 = u_t = -\frac{\partial\,(Lu)^2}{\partial u} + \nabla \cdot \frac{\partial\,(Lu)^2}{\partial \nabla u} \tag{8.31}$$

which corresponds to the Euler–Lagrange equation for the minimization of the least squares functional $\|Lu\|_{L_2}^2$. For the LSMFE method the optimal property holds just as for the variational method in Section 8.3.

It is natural to solve the nonlinear system of Equations (8.29) and (8.30) using the MFE time-stepping method, but any other convenient iteration can be used. In [11] the mass matrix of the MFE method is replaced by a Laplacian-regularization matrix.

8.4.2 Properties of the LSMFE Method

The LSMFE method has the following properties:

- The weak forms (8.29) and (8.30) arising from these variations correspond to Equations (8.15) and (8.16) with F given by $\frac{1}{2}\,(LU)^2$ and, therefore, have the optimal property.
- In the LSMFE tests carried out in [11] on scalar first-order steady equations the nodes no longer move with characteristic speeds but instead move to regions of high curvature. This is to be expected because the least-squares procedure embeds the original first-order equation in the *second-order* equation (8.31), and it is already known that, for Laplace's equation in one dimension, the final positions of the nodes in the MFE steady limit asymptotically equidistribute a $\frac{2}{5}$ power of the second derivative of $|u|$ [12].
- A third property is only stated here. In the particular case where LU takes the form of a divergence of a continuous function, a modification of the result discussed in Section 8.6.3 below shows that, *asymptotically*, minimization of $\|LU\|_{L_2}^2$ is equivalent to an equidistribution of LU over each element in the particular sense described there. For example, in the case where

  $$Lu = \nabla \cdot (\underline{a}u) \tag{8.32}$$

  with constant $\underline{a}$, and u is approximated by the continuous piecewise linear function U, the LSMFE method asymptotically equidistributes the piecewise constant residual $LU = \nabla \cdot (\underline{a}U)$ in each element. Within a coupled iteration scheme this ensures that convergence proceeds in a relatively uniform manner.

An illustration of the results of the method for a 2D circular advection problem is given in Figure 8.4, the mesh being very similar to that given by the least-squares, finite-volume method described there.

We now consider minimization of least squares functionals with moving nodes in which the approximations are of finite-volume type.

8.4.3 Minimization of Discrete Norms

The use of *discrete* norms with area weighting may be regarded as a simple quadrature of the L_2 norm used previously.

Let $\overline{LU}_T$ be the average residual in triangle T and the norm be the weighted sum over triangles of the average residual, viz.

$$\|LU\|_d^2 = \sum_T S_T \overline{LU}_T^2 \tag{8.33}$$

(cf. quadrature of the square of the L_2 norm of LU). Here the suffix T runs over all the triangles of the region, S_T is the area of triangle T given by (8.27), and $\overline{LU}_T$ is the average value of the residual LU over the vertices of T.

This norm coincides with the L_2 norm if LU is constant on each triangle. For then

$$\begin{aligned}\|LU\|_{L_2}^2 &= \int_\Omega LU^2 d\underline{x} = \sum_T \int_T LU^2 d\underline{x} = \sum_T \overline{LU}_T^2 \int_T d\underline{x} \\ &= \sum_T S_T \overline{LU}_T^2 = \|LU\|_d^2\end{aligned} \tag{8.34}$$

If the area weighting in (8.33) is omitted, this link is lost.

The form (8.33) may be rewritten as a sum over nodes j, namely

$$\|LU\|_d^2 = \frac{1}{3} \sum_j \sum_{T_j} S_T \overline{LU}_T^2 \tag{8.35}$$

where T_j runs over the patch of triangles abutting node j (cf. Figure 8.1).

When triangles become degenerate, however, the gradient is unbounded and the norm of the gradient is also unbounded. But by squaring the weight S_T in the norm in (8.33) S_T^2 a second norm

$$|\|LU\||_d^2 = \sum_T S_T^2 \overline{LU}_T^2 \tag{8.36}$$

is obtained which is always well defined, even on degenerate triangles.

Two moving-mesh methods based on discrete least-squares minimization will be described here, that of Roe [7] for scalar PDEs which is applicable to problems with continuous solutions, and that of Baines et al. [15] for systems of conservation laws which includes an algorithm that aligns triangle edges with shocks when flows containing such discontinuities are modeled.

8.4.4 Least-Squares Finite Volumes

In [7] Roe included the nodal positions in the minimization of the discrete least-squares norm (8.33) or (8.35) of the residual of a first-order PDE. Roe uses the idea of a *fluctuation* instead of the residual [13], which we now define.

The fluctuation is defined as the integral of the residual in a triangle T

$$\phi_T = \int_T LU d\underline{x} \,, \tag{8.37}$$

or in its finite-volume form, using quadrature,

$$\phi_T = S_T \overline{LU} \,. \tag{8.38}$$

The discrete least-squares norm of the residual, from (8.33) and (8.35), is therefore given by

$$\|LU\|_d^2 = \sum_T \frac{\phi_T^2}{S_T} = \frac{1}{3} \sum_j \sum_{T_j} \frac{\phi_T^2}{S_T} \,. \tag{8.39}$$

With the alternative weight S_T^2 in (8.36), the discrete least-squares norm of the residual is simply the l_2 norm of the fluctuation,

$$|\|LU\||_d^2 = \|\phi\|_{l_2}^2 = \sum_T \phi_T^2 = \frac{1}{3} \sum_j \sum_{T_j} \phi_T^2 \,. \tag{8.40}$$

By including mesh variables in the least squares minimization of (8.39) Roe in [7] alleviated the counting problems with the use of a fixed grid where, even though the norm of the residuals over a patch may vanish, the element residuals do not, leading to an unsatisfactory solution (see [8]). When nodal positions are included in the minimization process, the number of degrees of freedom is increased and at convergence the *element* fluctuations are driven close to zero and a much improved solution is obtained.

A steepest descent method was used in [7] in which local updates of the solution and the mesh were made with a safe value of the relaxation parameter τ. The convergence of the algorithm is extremely slow but can be improved by using a more sophisticated line search [14]. However, what enormously improves the convergence rate is an updating mechanism which does not come from a full least squares descent method but which takes updates only from the upwind direction [8].

As an illustration consider the scalar two-dimensional advection equation

$$\underline{a}(\underline{x}).\nabla u = 0 \,. \tag{8.41}$$

Then the fluctuation may be written

$$\phi_e = -\frac{1}{2} \sum_{ei=1}^{3} (a_{ei} U_{ei} \Delta Y_{ei} - b_{ei} U_{ei} \Delta X_{ei}) \tag{8.42}$$

where $\underline{a} = (a, b) = (a(X_{ei}, Y_{ei}), b(X_{ei}, Y_{ei}))$.

The steady-state residual is

$$\underline{a}(\underline{x}).\nabla U \tag{8.43}$$

and, from (8.38),

$$\phi_T = \frac{(\underline{\overline{a}}.\nabla U)^2}{S_T} \,. \tag{8.44}$$

Then the derivatives of (8.39) with respect to U_j and $\underline{X}_j$ reduce to

$$\left\langle \underline{\overline{a}}.\nabla U, \underline{\overline{a}}.\frac{\partial\,(\nabla U)}{\partial U_j} \right\rangle_d = \left\langle \underline{\overline{a}}.\nabla U, \frac{\partial\left(\underline{\overline{a}}.\nabla U\right)}{\partial \underline{X}_j} \right\rangle_d + \frac{1}{2}\sum_{\{T_j\}} \overline{\left(\underline{a}.\nabla U_T\right)}^2 \frac{\partial S_T}{\partial \underline{X}_j} = 0 \,, \tag{8.45}$$

subject to boundary conditions, where from (8.26) and (8.27)

$$\frac{\partial\,(\nabla U)_{T_j}}{\partial U_j} = \frac{1}{2S_{T_j}}\underline{n}_j \tag{8.46}$$

$$\frac{\partial\left(\underline{\overline{a}}.\nabla U\right)_{T_j}}{\partial \underline{X}_j} = \Delta U_j \begin{pmatrix} -\overline{b} \\ \overline{a} \end{pmatrix} - \frac{1}{2\left(S_{T_j}\right)} (\nabla U)_{T_j} \,. \tag{8.47}$$

Recall that $\underline{n}_j$ is the inward normal to the side of the triangle opposite node j scaled by the length of that side and ΔU_j is the increment in U across that side, taken counterclockwise (see Figure 8.1).

Equation (8.45) may therefore be written as:

$$\sum_{T_j} \left(\underline{\overline{a}}.\nabla U\right)_T \left(\underline{\overline{a}}.\underline{n}\right)_T = 0 \tag{8.48}$$

and

$$\sum_{T_j} \left\{ \left(\underline{\overline{a}}.\nabla U\right)_T \Delta U_j \begin{pmatrix} -b \\ a \end{pmatrix} - \frac{1}{2}\left(\underline{a}.\nabla U\right)_T^2 \underline{n} \right\} = 0 \,. \tag{8.49}$$

We observe that (8.48) is identical to (8.29) when $LU = \underline{a}.\nabla U$, noting that ∇U is constant and $\nabla\phi = S_T^{-1}\underline{n}$. However, (8.49) does not reduce to (8.30) even when $\underline{a}$ is constant.

We now show an example taken from [7].

8.4.5 Example

Let $\underline{a}(\underline{x}) = (y, -x)$ in a rectangle $-1 \le x \le 1, 0 \le y \le 1$. Then the solution of (8.41) is a semicircular annulus swept out by the initial data, here chosen to be

$$U = \begin{cases} 1 & -0.6 \le x \le -0.5 \\ 0 & \text{otherwise} \,. \end{cases} \tag{8.50}$$

Results are shown in Figures 8.3 and 8.4 for a fixed and moving mesh, respectively, taken from [15].

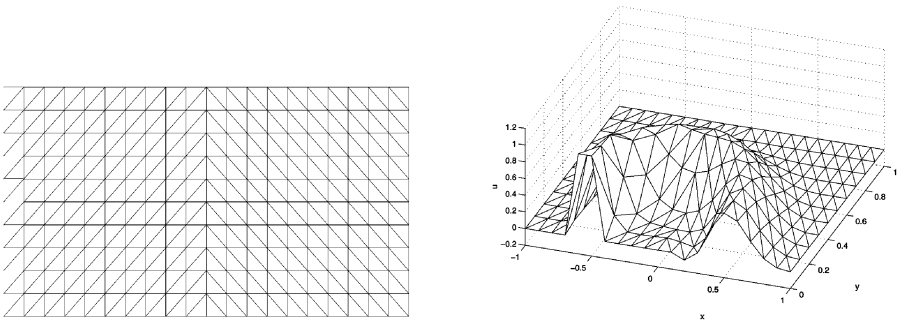


FIGURE 8.3
Initial mesh and solution for the circular advection problem.

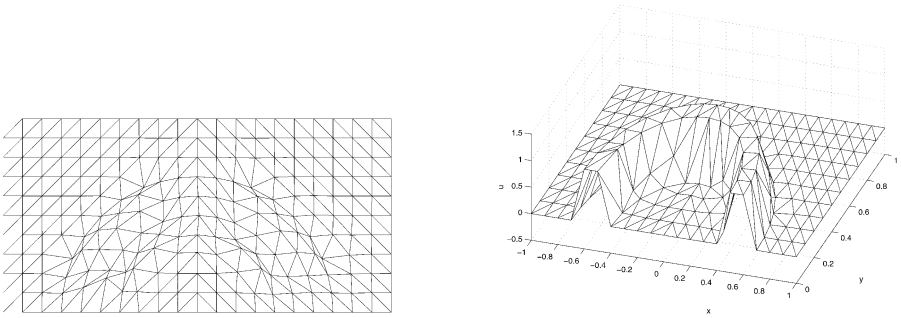
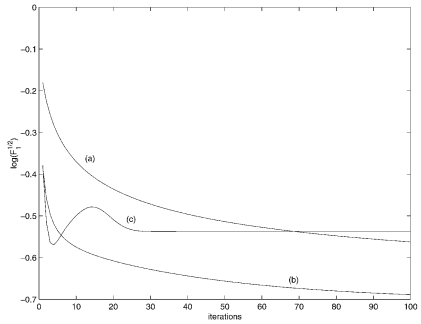


FIGURE 8.4
Final mesh and solution for the circular advection problem.

As expected, the solution on a fixed mesh is poor. However, when the mesh takes part in the minimization the norm is driven down to machine accuracy. The redistribution effected by the least squares minimization forces global conservation and equidistributes ϕ among the triangles (see Section 8.6.3) leading to more uniform convergence. Cell edges have approximately aligned with characteristics in regions of non-zero ϕ, allowing a highly accurate solution to be obtained. Essentially the same final mesh is obtained by the LSMFE method of Section 8.3.

The left-hand graph in Figure 8.5 shows the convergence of the solution updating procedure on the fixed mesh using (a) steepest descent with a global relaxation factor $\tau = 0.5$, (b) optimal local updates using a quadratic model, (c) optimal local updates over downwind cells only. Convergence is much improved in (b) and (c). Even though (c) is not monotonic it converges very quickly, albeit to a higher value of the functional, due to the nature of the procedure.

The convergence rates obtained when the nodes are allowed to move are shown in the right-hand graph in Figure 8.5. The iteration is started from the converged solution on the fixed mesh and uses (a) steepest descent with the global relaxation



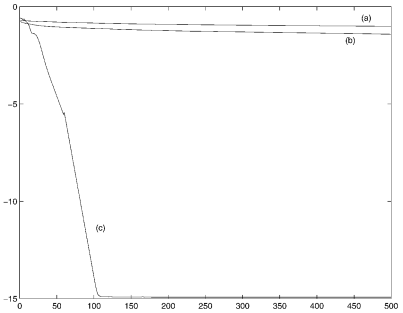


FIGURE 8.5
Comparison of convergence histories.

factors $\tau = 0.5$ for the solution and $\tau = 0.01$ for the meshpoints, (b) a line search using a Newton iteration, (c) a line search with updates over downwind cells only.

A small amount of mesh smoothing was included in (b) and (c). In particular, (b) became stuck in a local minimum if more iterations are used. Node locking was a problem with the full least squares approach. Node removal or steepest descent updates may be used to alleviate this problem ([6, 3]) but when tried in [14] still took over 1000 iterations and so were not competitive when compared to the upwinding approach, which yielded the best result.

Discrete least squares solutions of the Stokes Problem have been considered in [21]. Here the two different discrete norms (8.39) and (8.40) were compared, one with the area weighting and one without, but little difference was seen in the results.

8.5 Conservation Laws by Least Squares

Finally we describe an advance in locating shocks in the solution of systems of nonlinear hyperbolic equations. In [15] the least squares minimization technique was used for systems, combining shock capturing techniques with shock fitting.

A general system of conservation laws is of the form

$$div\underline{\mathbf{f}}(\mathbf{u}) = 0 = \underline{A}(\mathbf{u}).\underline{\nabla}\mathbf{u} \tag{8.51}$$

where $\underline{A}$ is a vector of the Jacobian matrices $(A, B)^T$. The integral form is

$$\oint_{\Gamma} \underline{\mathbf{f}}(\mathbf{u}).\underline{\hat{n}}d\Gamma = 0 \tag{8.52}$$

where $\underline{\hat{n}}$ is the inward facing unit normal.

It is assumed that $\underline{\mathbf{f}}$ is approximated by a piecewise linear function $\underline{\mathbf{F}}$. Then the fluctuation in triangle T is defined to be

$$\mathbf{\Phi}_T = -\int_T div \underline{\mathbf{F}} d\underline{x} = \oint_{\partial T} \underline{\mathbf{F}}.\underline{\hat{n}}\, ds \tag{8.53}$$

where ∂T is the perimeter of T. The average residual is also defined as

$$\overline{\mathbf{R}}_T = \frac{1}{S_T} \oint_{\partial T} \underline{\mathbf{F}}.\underline{\hat{n}}\, ds = \frac{\mathbf{\Phi}_T}{S_T} \tag{8.54}$$

where S_T is the area of triangle T.

Since $\underline{\mathbf{F}}$ is assumed to be linear in the triangle a trapezium rule quadrature can be used to write the fluctuation in triangle e, from (8.53), as

$$\mathbf{\Phi}_e = \frac{1}{2}\left\{\left(\underline{\mathbf{F}}_{e1} + \underline{\mathbf{F}}_{e2}\right).\underline{n}_{e3} + \left(\underline{\mathbf{F}}_{e2} + \underline{\mathbf{F}}_{e3}\right).\underline{n}_{e1} + \left(\underline{\mathbf{F}}_{e3} + \underline{\mathbf{F}}_{e1}\right).\underline{n}_{e2}\right\}\ , \tag{8.55}$$

where $\underline{n}_{ei}$ $(i = 1, 2, 3)$ is the inward unit normal to the ith edge of triangle e (opposite the vertex ei), as shown in Figure 8.6, multiplied by the length of that edge.

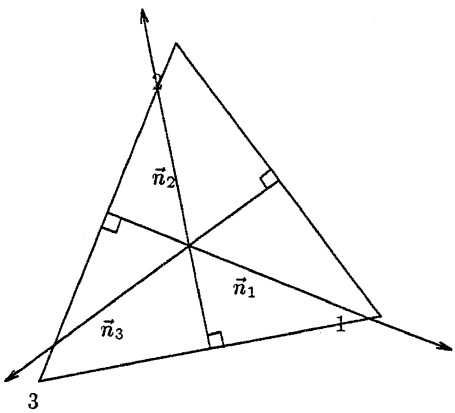


FIGURE 8.6
A general triangular cell e.

It is easy to verify that, for any triangle,

$$\underline{n}_{e1} + \underline{n}_{e2} + \underline{n}_{e3} = 0\ , \tag{8.56}$$

so the fluctuation (8.55) may be written as

$$\mathbf{\Phi}_e = -\frac{1}{2}\left\{\underline{\mathbf{F}}_{e1}.\underline{n}_{e1} + \underline{\mathbf{F}}_{e2}.\underline{n}_{e2} + \underline{\mathbf{F}}_{e3}.\underline{n}_{e3}\right\} \tag{8.57}$$

or, since $\underline{n}_{ei} = (\Delta Y_{ei}, -\Delta X_{ei})$, as

$$\mathbf{\Phi}_e = -\frac{1}{2}\sum_{ei=1}^{3}\left(\mathbf{F}_{ei}\Delta Y_{ei} - \mathbf{G}_{ei}\Delta X_{ei}\right) \tag{8.58}$$

[cf. (8.42), where $\underline{\mathbf{F}} = (\mathbf{F}, \mathbf{G})$ and $(\Delta_{e1}X, \Delta_{e1}Y) = (X_{e2} - X_{e3}, Y_{e2} - Y_{e3})$ denotes the difference in X taken across the side opposite node $e1$ in a counterclockwise sense (with similar definitions for $(\Delta_{e2}X, \Delta_{e2}Y)$ and $(\Delta_{e3}X, \Delta_{e3}Y)$] (see Figure 8.6). A useful dual form of the fluctuation is obtained by rewriting (8.58) as

$$\mathbf{\Phi}_e = \frac{1}{2} \sum_{ei=1}^{3} (Y_{ei} \Delta \mathbf{F}_{ei} - X_{ei} \Delta \mathbf{G}_{ei}) \ . \tag{8.59}$$

We aim to set the fluctuations $\mathbf{\Phi}_e$ to zero in order to minimize a vector form of (8.35).

Two special systems of interest are the Shallow Water equation system and the Euler equations of gasdynamics. Details of these systems are given in [14].

8.5.1 Use of Degenerate Triangles

In the presence of shocks least squares methods give inaccurate solutions which are unacceptable. In [15] a way of combating this problem is shown which is to divide the region into two domains and introduce degenerate triangles at the interface. The least squares method with moving nodes is then used to adjust the position of the discontinuity, as in shock fitting methods.

An initial approximate solution to the equations can be found by any standard method. In [15], a multidimensional upwinding shock capturing scheme is used. An initial discontinuous solution is then constructed by introducing degenerate (vertical) triangles in the regions identified as shocks, using a shock identification technique. This step is carried out manually in [15] although degenerate triangles can be added automatically using techniques that exist in the shock fitting literature (see, for example, [16]). The corners of the degenerate triangles are designated as shocked nodes and these form an internal boundary, on either side of which the least squares method is applied in the two smooth regions where it is known to perform well. The position of the discontinuity is then improved by minimizing a shock functional which is derived from (8.40).

Consider the interface shown in Figure 8.7. The fluctuations $\mathbf{\Phi}_{d1}$ and $\mathbf{\Phi}_{d2}$ in adjacent degenerate triangles d_1 and d_2 on the edges $i = (i_L, j_L)$ and $j = (i_R, j_R)$ are, from (8.57),

$$\mathbf{\Phi}_{d_1} = -\frac{1}{2} \left[\underline{\mathbf{F}}_i \right] . \underline{n}_{i_L} \qquad \mathbf{\Phi}_{d_2} = -\frac{1}{2} \left[\underline{\mathbf{F}}_j \right] . \underline{n}_{j_R} \tag{8.60}$$

respectively, where the square bracket denotes the jump across the discontinuity.

Then

$$\mathbf{\Phi}_{d_1}^T \mathbf{\Phi}_{d_1} + \mathbf{\Phi}_{d_2}^T \mathbf{\Phi}_{d_2} = \frac{1}{4} \left\{ \left([\underline{\mathbf{F}}_i] . \underline{n}_{i_L} \right)^T \left([\underline{\mathbf{F}}_i] . \underline{n}_{i_L} \right) + \left([\underline{\mathbf{F}}_j] . \underline{n}_{j_L} \right)^T \left([\underline{\mathbf{F}}_j] . \underline{n}_{j_L} \right) \right\} \tag{8.61}$$

and the functional

$$\sum_{T \in \Omega_D} \mathbf{\Phi}_T^T \mathbf{\Phi}_T \tag{8.62}$$

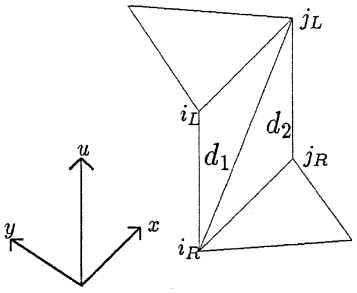


FIGURE 8.7
Degenerate triangles d_1, d_2.

[cf. (8.40)] is minimized to improve the position of the shock, where Ω_D is the set of degenerate triangles. This norm is always bounded, even at shocks where **U** is discontinuous. On the other hand, the average residual, given by (8.39), is not bounded at shocks.

A recent result of Nishikawa et al. [28] somewhat surprisingly suggests that the capability of fluctuation splitting methods to capture characteristics or shocks depends on the quadrature used in defining the fluctuation.

A descent least squares method is used on (8.62) to move the shocked nodes into a more accurate position. The procedure is interleaved with a descent least squares method on (8.39) for the smooth solution on either side.

When updating the nodal positions $\underline{X}_{i_L}$ and $\underline{X}_{i_R}$ it is required that they have the same update (so that the cell remains degenerate). The update comes from minimization with respect to their common position vector. Degenerate quadrilaterals can be used instead of degenerate triangles.

We reproduce two results using this technique, taken from [15].

8.5.2 Numerical Results for Discontinuous Solutions

Results are shown from two problems which exhibit discontinuous solutions, one for the Shallow Water equations and the other for the Euler equations of gasdynamics.

The Shallow Water equations system can be used to describe the problem of a transcritical constricted channel flow which exhibits a hydraulic jump in the constriction. The computational domain represents a channel of length 3 meters and width 1 meter with a 10% bump in the middle third. The freestream Froude number is defined to be $F_\infty = 0.55$, the freestream depth is $h_\infty = 1m$ [and the freestream velocity is given by $(u_\infty, v_\infty) = (1.72, 0)$]. An initial solution is found by the Elliptic-Hyperbolic Lax–Wendroff multidimensional upwinding scheme of Mesaros and Roe, see [17]. The hydraulic jump is then located and degenerate quadrilaterals added at the approximate position of the shock. The best position of the shock is then sought using a least squares descent method with degenerate triangles, moving the nodes to improve the position of the shock.

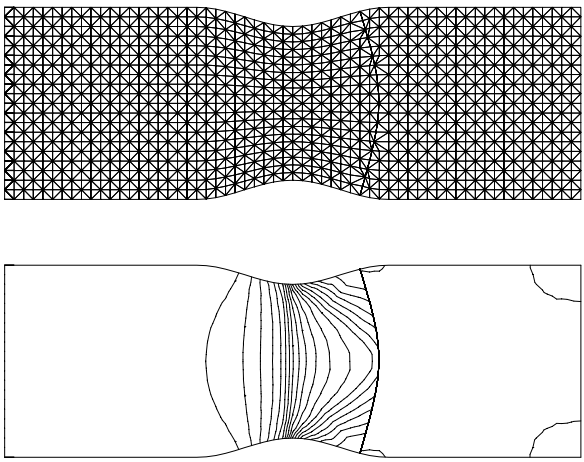


FIGURE 8.8
Mesh and height contours for the Shallow Water example.

Results are shown in Figure 8.8 which shows the meshes and solution depth contours obtained. A bow-shaped hydraulic jump which is strongest at the boundaries is predicted. This agrees with the solution obtained using a shock capturing solution on a very fine mesh. Here it is achieved sharply at very much less cost.

In the second example the Euler equations of gasdynamics are considered written in conserved variables.

The example chosen exhibits the shock fitting capabilities of the method for a purely supersonic flow which has an exact solution [18]. The computational domain is of length 3 meters and width 1 meter. Supersonic inflow boundary conditions, given by

$$\begin{aligned} \underline{U}(0, y) &= (1.0, 2.9, 0, 5.99073)^t \\ \underline{U}(x, 1) &= (1.69997, 4.45280, -0.86073, 9.87007)^t \,, \end{aligned} \tag{8.63}$$

are imposed on the left and upper boundaries, respectively. At the right-hand boundary supersonic outflow conditions are applied, while the lower boundary is treated as a solid wall.

The boundary conditions are chosen so that the shock enters the top left-hand corner of the region at an angle of 29° to the horizontal and is reflected by a flat plate on the lower boundary. The flow in regions away from shocks is constant. The same strategy is employed as in the previous example, with the results shown in Figure 8.9 where the mesh and the density are shown. The solution has a shock which comes in from the top left hand at an angle of 29.2° to the horizontal and is virtually constant apart from the discontinuities, in close agreement with the analytic solution.

The final section in this chapter highlights the links between the minimization procedures discussed previously and the ideas of equidistribution.

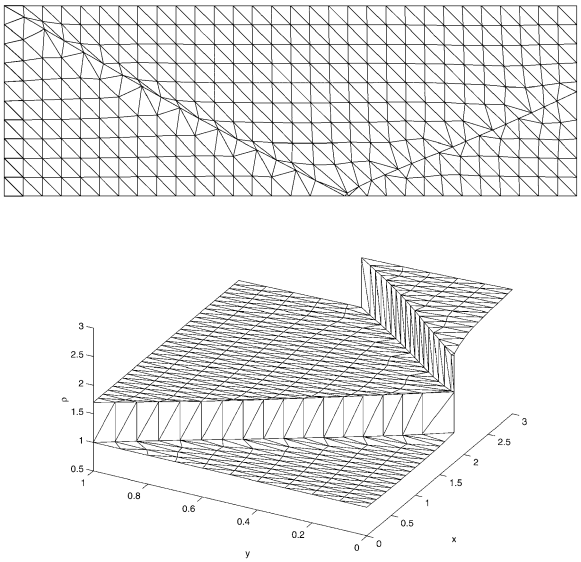


FIGURE 8.9
Mesh and density for the Euler equations example.

8.6 Links with Equidistribution

The well-known equidistribution principle (EP) in one dimension involves locating meshpoints such that some measure of a function is equalized over each subinterval [22]. In one dimension, denoting by x and ξ the physical and computational coordinates, respectively, define a coordinate transformation

$$x = x(\xi) \qquad \xi \in [0, 1] \tag{8.64}$$

with fixed end points $x(0) = a, x(1) = b$, say. The computational coordinates are given by

$$\xi_i = \frac{i}{N} \quad , \quad i = 0, 1, ..., N \tag{8.65}$$

where N is the number of mesh points.

A positive monitor function $M(u)$ is chosen that provides some desired measure of the solution u to be equidistributed. The integral form of the EP is then given by

$$\int_0^{x(\xi)} M(u)dx = \xi\theta \tag{8.66}$$

where $\theta = \int_0^1 M(u)dx$. Differentiating (8.66) twice with respect to ξ gives the alternative differential form

$$\frac{\partial}{\partial \xi}\left(M(u)\frac{\partial x}{\partial \xi}\right) = 0\,. \tag{8.67}$$

In practice a discretized form of (8.67)

$$M_{j-1/2}\left(x_j - x_{j-1}\right) = M_{j+1/2}\left(x_{j+1} - x_j\right) \tag{8.68}$$

may be solved subject to the boundary conditions $x(0) = a, x(1) = b$.

The method has become very popular in many contexts. However, different monitor functions are often required for different purposes [22].

Since the monitor function depends on u, which depends in turn on x, an iteration procedure is needed to solve (8.68). More specifically, the monitor function depends on the solution of the PDE, so Equation (8.68) should be thought of as just one step in an iterative algorithm for both the mesh and the solution.

In iteratively solving (8.68) a single step of an iteration for each equation may be generated and the mesh iterations interleaved with the iterations for solving the PDE. These iteration steps may be chosen to involve only one node at a time (so that the iteration is tantamount to a sweep through the mesh) and then we have a sequence of local problems as in Sections 8.3 through 8.5 above.

8.6.1 Approximate Multidimensional Equidistribution

Equidistribution was conceived as a technique for approximation in one dimension. Nevertheless, recently there have been important developments in multidimensional equidistribution, (see [23, 20]). However, we shall discuss only approximate generalizations to higher dimensions here, since these are simple to implement and relate to other ideas in this chapter.

A formal generalization of (8.68) is

$$\sum_{e\in\{T_j\}} M_e\left(\underline{\overline{x}}_e^n - \underline{x}_j^n\right) = 0 \tag{8.69}$$

where $\underline{\overline{x}}_e^n$ is the centroid of triangle e, M_e is a weight and $\{T_j\}$ is the set of triangles surrounding node j. (see Figure 8.1). Although the formula is convex, there are examples of meshes in which mesh tangling can take place in this case [19].

The formula (8.69) is not a statement of equidistribution, but it does have an interpolatory status, intuitively equidistributing in the two limits

(a) if $M_e = constant\,\forall e$, $\underline{x}_j$ is the average of the centroids of the surrounding triangles,

(b) if one M_e dominates, say M_E, then the nodes cluster toward the centroid of the element E.

The positions of the mesh vertices may also be interpreted as the solution of the least squares minimization problem (see [24, 25])

$$\min_{\underline{x}_j} \sum_e M_e \left(\underline{x}_j - \underline{x}_{ej} \right)^2 , \tag{8.70}$$

8.6.2 A Local Approach to Approximate Equidistribution

Again, the use of iterative techniques facilitates a local approach in which the iteration may be carried out one node at a time, sweeping through the mesh. Consider the iteration in which mesh points are moved to weighted averages of the positions of centroids of adjacent cells.

In one dimension an iteration for the solution of (8.68) is of the form

$$x_j^{n+1} = \frac{M_{j-\frac{1}{2}} \left(x_j^n + x_{j-1}^n \right) + M_{j+\frac{1}{2}} \left(x_j^n + x_{j+1}^n \right)}{2 \left(M_{j-\frac{1}{2}} + M_{j+\frac{1}{2}} \right)} . \tag{8.71}$$

which is convex, convergent, and does not allow mesh tangling [19]. In two dimensions a corresponding iteration for (8.69) is

$$\underline{x}_j^{n+1} = \frac{\sum_{e \in \{T_j\}} M_e \overline{\underline{x}}_e^n}{\sum_{e \in \{T_j\}} M_e} . \tag{8.72}$$

8.6.3 Approximate Equidistribution and Conservation

A link between discrete least squares and equidistribution is described in [20] where it is shown that least squares minimization of the residual of the divergence of a vector field is equivalent to that of a least squares measure of "equidistribution" of the residual.

The conservation law (8.51) is considered where $\underline{\mathbf{u}}$ is approximated by the continuous approximation $\underline{\mathbf{U}}$. The fluctuation $\mathbf{\Phi}_e$ is defined as in (8.53) and the average residual as in (8.54). Then the following identity holds.

$$\left(\sum_{i=1}^N S_i \right) \left(\sum_{i=1}^N \bar{\mathbf{R}}_T^T S_T \bar{\mathbf{R}}_T \right)$$

$$= \left(\sum_{i=1}^N \mathbf{\Phi}_i \right)^2 + \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \left(\bar{\mathbf{R}} T_i - \bar{\mathbf{R}} T_j \right)^T S_{T_i} S_{T_j} \left(\bar{\mathbf{R}}_{T_i} - \bar{\mathbf{R}}_{T_j} \right) . \tag{8.73}$$

Now

$$\sum_{i=1}^N S_{T_i} = \Omega \tag{8.74}$$

is equal to the total area of the union of the triangles Ω and may be taken to be constant. Moreover, by definition,

$$\sum_{i=1}^{N} \mathbf{\Phi}_i = -\sum_{i=1}^{N} \int_e div \underline{\mathbf{F}}(\underline{u}) d\underline{x} = \oint_{\partial\Omega} \underline{\mathbf{F}}\left(\underline{\mathbf{u}}\right) .d\underline{s} \tag{8.75}$$

by internal cancellation, which is independent of interior values of $\underline{\mathbf{F}}$ and interior mesh locations.

We may then write (8.73) as

$$\Omega \left\|\bar{\mathbf{R}}\right\|_{l_2}^2 = \left(\oint_{\partial\Omega} \underline{\mathbf{F}}.ds\right)^2 + \left|\left\|\bar{\mathbf{R}}\right\|\right|_{eq}^2 \tag{8.76}$$

where

$$\left\|\bar{\mathbf{R}}\right\|_{l_2}^2 = \sum_{i=1}^{N} \bar{\mathbf{R}}_T^T S_T \bar{\mathbf{R}}_T \tag{8.77}$$

corresponding to (8.39) and where

$$\left|\left\|\bar{\mathbf{R}}\right\|\right|_{eq}^2 = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \left(\bar{\mathbf{R}}_{T_i} - \bar{\mathbf{R}}_{j\,T}\right)^T S_{T_i} S_{T_j} \left(\bar{\mathbf{R}}_{T_i} - \bar{\mathbf{R}}_{T_j}\right) \tag{8.78}$$

which is a measure of equidistribution of the average residual $\bar{\mathbf{R}}_T$.

A similar result can be derived for the norm (8.40). The identity

$$N \left\|\mathbf{\Phi}\right\|_{l_2}^2 \equiv \left(\oint_{\partial\Omega} \underline{\mathbf{F}}.d\underline{s}\right)^2 + \left|\left\|\mathbf{\Phi}\right\|\right|_{eq}^2 \tag{8.79}$$

holds, where N is the number of cells and

$$\left\|\mathbf{\Phi}\right\|_{l_2}^2 = \sum_{i=1}^{N} \mathbf{\Phi}_{T_i}^T \mathbf{\Phi}_{T_i} \tag{8.80}$$

proportional to (8.40), and

$$\left|\left\|\mathbf{\Phi}\right\|\right|_{eq}^2 = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \left(\mathbf{\Phi}_{T_i} - \mathbf{\Phi}_{T_j}\right)^T \left(\mathbf{\Phi}_{T_i} - \mathbf{\Phi}_{T_j}\right) . \tag{8.81}$$

If we allow only interior mesh points to be varied, then (8.75) is a fixed quantity and in any minimization procedure the two norms (8.77) and (8.78) [or (8.80] and (8.81)] will be minimized simultaneously. The minimization of (8.77) [or (8.80)] [corresponding to finding a least squares approximation to the solution of (8.51)] is equivalent to minimizing a measure of equidistribution over the triangles in the sense

of (8.78) or (8.81). This result holds in any number of dimensions and in an iterative context encourages convergence to take place in a uniform way.

Finite volume methods of the type discussed here may not give very accurate solutions. However, as far as the mesh is concerned, high accuracy is not crucial. A finite volume approach may therefore be sufficiently accurate for the mesh locations but for a higher order solution a more sophisticated method, such as high order finite elements or multidimensional upwinding ([26, 27]), may be required for the solution on the optimal mesh.

8.7 Summary

The MFE method is a Galerkin method extended to include node movement. For the PDE

$$Lu = -\frac{\partial F}{\partial u} + \nabla . \frac{\partial F}{\partial \nabla u} = 0 \tag{8.82}$$

the steady MFE equations provide a local optimum for the variational problem

$$\min_{U_j, \underline{X}_j} \int F(U, \nabla U) d\underline{x} \tag{8.83}$$

in a piecewise linear approximation space with moving nodes.

Solutions of such PDEs may also be obtained by direct minimization of (8.83). A local approach is possible which is advantageous in reducing the complexity of the mesh location procedure and in applying constraints which preserve the integrity of the mesh. An approach of this kind was described in Section 8.3.

The Least Squares Moving Finite Element method (LSMFE) is a least squares method for steady first order PDEs which includes node movement. In the steady state the LSMFE equations for $Lu = 0$ are equivalent to the steady MFE weak forms for the PDE

$$-\frac{\partial (Lu)^2}{\partial u} + \nabla . \left(\frac{\partial (Lu)^2}{\partial \nabla u} \right) = 0 \tag{8.84}$$

and therefore provide a local minimum for the variational problem

$$\min_{U_j, \underline{X}_j} \int (LU)^2 \, d\underline{x} \tag{8.85}$$

Moreover, if LU is the divergence of a continuous flux function then the flux across element boundaries is *asymptotically* equidistributed over the elements.

A least squares finite volume fluctuation distribution scheme with mesh movement, given by Roe in [7] is an adaptive mesh method based on minimization of a weighted l_2

norm of the residual of a steady first order PDE over the solution and the mesh. It also uses a local approach and a steepest descent algorithm. It lacks the optimal property of LSMFE but has the property that, if LU is the divergence of a continuous flux function, then the flux across element boundaries is equidistributed over the elements in the sense of (8.78) or (8.81), thus proceeding to the steady limit in a uniform way.

For scalar problems convergence can be greatly accelerated by carrying out the iterations in an upwind manner.

For problems with discontinuities, the mesh movement technique enables improvement of the location of the discontinuity in a manner akin to shock fitting. By minimizing a measure of the fluctuation in degenerate triangles, an initially approximate position of the shock can be maneuvered into an accurate position. The descent least squares method may be used on either side of the shock to gain good approximations in the smooth regions of the flow.

References

[1] K. Miller, Moving finite elements I (with R.N.Miller) and II, *SIAM J. Num. Anal.,* **18**, 1019–1057, (1981).

[2] P.K. Jimack, *Local Minimization of Errors and Residuals Using the Moving Finite Element Method,* University of Leeds Report 98.17, School of Computer Science, (1998).

[3] Y. Tourigny and F. Hulsemann, A new moving mesh algorithm for the finite element solution of variational problems, *SIAM J. Num. Anal.,* **34**, 1416–1438, (1998).

[4] N.N. Carlson and K. Miller, Design and application of a gradient weighted moving finite element method I: in one dimension. II: in two dimensions, *SIAM J. Sci. Comp.,* **19**, 728–798, (1998).

[5] M.J. Baines, *Moving Finite Elements,* Oxford University Press, (1994).

[6] Y. Tourigny and M.J. Baines, Analysis of an algorithm for generating locally optimal meshes for L_2 approximation by discontinuous piecewise polynomials, *Math. Comp.,* **66**, 623–650, (1998).

[7] P.L. Roe, Compounded of many simples. In *Proceedings of Workshop on Barriers and Challenges in CFD,* ICASE, NASA Langley, August 1996, Ventakrishnan, Salas and Chakravarthy, eds., 241-, Kluwer (1998).

[8] M.J. Baines and S.J. Leary, Fluctuation and signals for scalar hyperbolic equations on adjustable meshes, *Com. Num. Meth. Eng.,* **15**, 877–886, (1999).

[9] M.J. Baines, Algorithms for optimal discontinuous piecewise linear and constant L_2 fits to continuous functions with adjustable nodes in one and two dimensions, *Math. Comp.,* **62**, 645–669, (1994).

[10] M. Delfour et al., An optimal triangulation for second order elliptic problems, *Comput. Meths. Applied Mech. Engrg.,* **50**, 231–261, (1985).

[11] K. Miller and M.J. Baines, *Least Squares Moving Finite Elements,* OUCL report 98/06, Oxford University Computing Laboratory, (1998), to appear in the *IMA Journal of Numerical Analysis.*

[12] G.F. Carey and H.T. Dinh, Grading functions and mesh distribution, *SIAM J. Num. An.,* **22**, 1028–1050, (1985).

[13] H. Deconinck, P.L. Roe, and R. Struijs, A multidimensional generalisation of Roe's flux difference splitter for the Euler equations, *Computers and Fluids,* **22**, 215, (1993).

[14] S.J. Leary, *Least Squares Methods with Adjustable Nodes for Steady Hyperbolic PDEs,* PhD Thesis, Department of Mathematics, University of Reading, UK, (1999).

[15] M.J. Baines, S.J. Leary, and M.E. Hubbard, Multidimensional least squares fluctuation distribution schemes with adaptive mesh movement for steady hyperbolic equations, (2000) (submitted to *SIAM J. Sci. Stat. Comp.*,). See also by the same authors A finite volume method for steady hyperbolic equations, in *Proceedings of Conference on Finite Volumes for Complex Applications II,* R. Vilsmeier, F. Benkhaldoun and D. Hanel, eds., Duisburg, July 1999, 787–794, Hermes, (1999).

[16] J.Y. Trepanier, M. Paraschivoiu, M. Reggio, and R. Camarero, A conservative shock fitting method on unstructured grids, *J. Comp. Phys.,* **126**, 421–433, (1996).

[17] L.M. Mesaros and P.L. Roe, Multidimensional fluctuation splitting schemes based on decomposition methods, *Proceedings of the 12th AIAA CFD Conference,* San Diego, (1995).

[18] H. Yee, R.F. Warming, and A. Harten, Implicit total variation diminishing (TVD) schemes for steady state calculations, *J. Comp. Phys.,* **57**, 327–366, (1985).

[19] M.J. Baines and M.E. Hubbard, Multidimensional upwinding with grid adaptation, in *Numerical Methods for Wave Propagation,* E.F. Toro and J.F. Clarke, eds., Kluwer, (1998).

[20] M.J. Baines, Least-squares and approximate equidistribution in multidimensions, *Numerical Methods for Partial Differential Equations,* **15**, 605–615, (1999).

[21] H. Nishikawa, The Discrete Least Squares Method for 2D Stokes Flow, Technical Report (unpublished), Department of Aerospace Engineering, University of Michigan, (1997).

[22] E.A. Dorfi, and L.O'C. Drury, Simple adaptive grids for 1D initial value problems, *J. Comput. Phys.,* **69**, 175–195, (1987).

[23] W. Huang and R.D. Russell, Moving mesh strategy based upon a gradient flow equation for two-dimensional problems, *SIAM J. Sci Stat. Comput.,* **20**, 998, (1999).

[24] D. Ait-Ali-Yahia, W.G. Habashi, A. Tam, M.G. Vallet, and M. Fortin, A directionally adaptive finite element method for high speed flows, *Int. J. for Num. Meths. in Fluids,* **23**, 673–690, (1996).

[25] J.A. Mackenzie, A Moving Mesh Finite Element Method for the Solution of Two-Dimensional Stefan Problems, Technical Report 99/26, Department of Mathematics, University of Strathclyde, UK, (1999).

[26] C. Johnson, *Finite Element Methods for Partial Differential Equations,* Cambridge University Press, (1993).

[27] M.E. Hubbard, Multidimensional Upwinding and Grid Adaptation for Conservation Laws, PhD Thesis, Department of Mathematics, University of Reading, UK, (1996).

[28] H. Nishikawa, M. Rad, and P.L. Roe, Grids and Solutions for Residual Distribution, Private communication, (2000).

Chapter 9

Linearly Implicit Adaptive Schemes for Singular Reaction-Diffusion Equations

Q. Sheng and A.Q.M. Khaliq

9.1 Introduction

Many important physical processes, such as the combustion of gases in a heat engine, can be described by nonlinear reaction-diffusion partial differential equations with singular or near-singular source terms. The rate of change of the solution of such equations may blow up in finite time, while the solution itself remains bounded, when certain physical quantities, such as the size of the combustor, reach their limits. The phenomenon is often referred to as quenching [1], [4]–[7], [11]–[13], [26]–[30], [32].

Mathematically, quenching phenomena can be interpreted as the blow-up of rates of change of solutions of nonlinear reaction-diffusion differential equations. This can occur when certain physical parameters, for example, the size of the combustor, reach their critical limits when particular gases are used. It has become extremely important to estimate such limit values efficiently and effectively for various reaction-diffusion models so that better controls and designs can be achieved in industrial applications.

Consider the following simplified reaction-diffusion problem with a highly nonlinear source function:

$$u_t = u_{xx} + f(u),\ 0 < x < a,\ 0 < t < T\ , \tag{9.1}$$

$$u(x,0) = u_0, 0 < x < a; u(0,t) = u(a,t) = 0, 0 < t < T, 0 \le u_0 < 1\ , \tag{9.2}$$

where $f(u) = 1/(1-u)^\theta, \theta > 0, T \le \infty$, and a is an important parameter playing in the combustion process. This model describes a steady-state combustion of two gases meeting in a gap between porous walls at distance a apart. Fuel diffuses at one wall, oxidant at the other with a zone of reaction between the walls, and dying out towards each wall. Here x is the distance from one of the walls, t is the time, and u is the uniformly scaled temperature. Further, θ is a physical property index of the gases involved [2, 12, 13]. Kawarada [17] discovered in 1975 that when $\theta = 1$ there exists

a critical value $a^* > 0$ such that for $a < a^*$, the solution of (9.1) and (9.2) exists globally and for $a \geq a^*$, there exists a finite T_a such that $\lim_{t \to T_a} u(a/2, t) = 1$, and furthermore $u(a/2, t) = \max\{u(x, t) : 0 \leq x \leq a\}$. The solution of (9.1) and (9.2) is said to quench at the time T_a in the latter case and a^* is defined as the *critical length* of the problem (9.1) and (9.2). Kawarada further observed that $a^* > 2\sqrt{2}$ and later, Acker and Walter improved the estimate to $a^* \approx 1.5303$ (see [1, 17, 32] and references therein).

Quenching phenomena have important practical meanings in manufacturing industries. They indicate the switch of the steady and unsteady combustion processes or burning and explosion of rocket fuel [2, 12]. A better understanding of quenching criteria may lead to an optimal design of the reaction-diffusion environments, especially when precision combustion processes are involved. Quenching phenomena are also important to the theory of ecology and related environmental research [6, 27, 29, 32].

However, numerical simulations of such reaction-diffusion equations have been among the most difficult tasks in fields of thermal engineering and scientific computations. A number of numerical methods have been constructed, investigated, and used for computing estimated limit values for quenching-type partial differential equations (9.1) and (9.2) [1], [4]–[7]. Most of the computations are carried out indirectly and based on Kawarada's original work, that is, approximating the critical length and quenching time through a reduced differential equation

$$u_{xx} = -f(u), \quad 0 < x < a$$

together with the condition (9.2). In this reduced problem it is possible to produce a sequence converging to the solution. Other numerical techniques, such as the boundary element method and finite difference algorithms, can also be used when multidimensional problems are considered. However, in many cases the computational procedures can be complicated with unexpected errors in certain practical applications and the numerical solution can hardly be satisfied [4]–[7], [13], [27], [31].

Much effort has been devoted in recent years to develop directly computing algorithms for evaluating the critical quantities for the nonlinear quenching models (9.1) and (9.2). A number of interesting techniques discussed by Khaliq and Voss et al. [18, 31] can be conveniently modified for attacking quenching problems. On the other hand, adaptive, or moving mesh, methods are developed for differential equation problems possessing singularities. Pareyra and Sewell [22] investigated mesh selection for solving ordinary equations. Later, Lang and Walter [19, 20] systematically studied the possibility of solving a wide class of nonlinear reaction-diffusion systems directly through combinations of semidiscretization/cascadic finite element methods and adaptive higher-order Runge–Kutta solvers. Recently, Budd et al. [3], Huang et al. [15], Ren and Russell [23], and Russell and Christiansen [24] proposed and investigated adaptive approaches for nonlinear blow-up problems in which the solution of the equations becomes unbounded in certain circumstances. However, in their discussions, the nonlinear source term was always assumed to be sufficiently smooth while in our case, the source term is always singular. Developments of adaptive schemes

for solving quenching-type singular reaction-diffusion equations are difficult and are not fulfilled until recent studies of Sheng, Khaliq, and Cheng [26]–[29].

We will consider a more general degenerative reaction-diffusion model,

$$x^q u_t = u_{xx} + c(x)u_x + f(u), \quad 0 < x < a,\ 0 < t < T,\ \ q \geq 0\,, \tag{9.3}$$

together with initial and boundary conditions (9.2). Discussions of the existence and uniqueness of its solutions can be found in [4, 7, 21] and references therein. Numerical techniques from the method of fundamental solutions, finite element approximations, and Douglas algorithms used for solving the equation are investigated by several authors (see [5, 6] and references therein). Most of the approaches, however, are still indirect and considerations of reduced problems of (9.3) are required. Less complicated, more efficient adaptive numerical methods have become necessary for computations of quenching solutions and critical values. This becomes the goal of our discussion.

Let $y = x/a$. Equation (9.3), together with (9.1), can be reformulated as

$$y^q u_t = \frac{1}{a^{q+2}} u_{yy} + \frac{c(ay)}{a^{q+1}} u_y + \tilde{f}(u), \quad 0 < y < 1,\ 0 < t < T\,, \tag{9.4}$$

$$u(ay, 0) = u_0,\ 0 < y < 1;$$

$$u(0, t) = u(a, t) = 0,\ 0 < t < T,\ 0 \leq u_0 < 1\,, \tag{9.5}$$

where $\tilde{f} = a^{-q} f$. A computational advantage of the above reformulation is that the discretization in space becomes simpler. We avoid dealing with a very sensitive a in the discretization, and move the quenching parameter directly into the differential equation. This will be helpful for introducing proper adaptive mechanisms later. We also note that in the particular case when $c \equiv 0$ and $a \geq 1$, (9.4) reduces to the standard parabolic equation with the singular source term, that is,

$$y^q u_t = \epsilon u_{yy} + f(u), \quad 0 < y < 1,\ 0 < t < T\,, \tag{9.6}$$

where $0 < \epsilon = 1/a^{q+2} \leq 1$.

In this study, we will construct efficient adaptive methods for computing the numerical solution, critical length, and quenching time of the nonlinear reaction-diffusion problem (9.4) and (9.5) directly. Nonlinear source functions with different indices θ will be considered in the numerical demonstrations. Techniques of semidiscretizations in spatial variables are used. For the system of nonlinear ordinary differential equations obtained, we introduce a two-stage Runge–Kutta solver, then L-stable rational functions with real and distinct poles for approximating derived matrix exponentials. The modified arc-length adaptive mechanism is established. Special consideration is given to the stability and efficiency in handling the degenerate and singular properties. The semi-adaptive method constructed is of second-order accuracy, while the fully adaptive scheme is of first-order accuracy. We then compare our results with existing results obtained by Acker and Walter [1], Chan et al. [5], and Walter [32], and show that our numerical method is accurate and reliable. An important feature of our algorithm may be that it does not depend on the structure

of the nonlinear source term presented. The numerical scheme can thus be extended for solving more sophisticated reaction-diffusion models with different inputs and sources, and for solving multidimensional problems without major difficulties.

9.2 The Semi-Adaptive Algorithm

9.2.1 The Discretization

Let $q \equiv 0$. For positive integer N, we define $h = 1/(N+1)$ as the spatial discretization parameter and let $u_k(t)$ be an approximation of the exact solution of (9.4) and (9.5) at (hk, t), $k = 0, 1, \ldots, N+1$. Replacing the first- and second-order spatial derivatives in (9.4) by central difference approximations

$$\begin{aligned} \frac{\partial v(y,t)}{\partial y} &= \frac{v(y+h,t) - v(y-h,t)}{2h} + O(h^2) , \\ \frac{\partial^2 v(y,t)}{\partial y^2} &= \frac{v(y+h,t) - 2v(y,t) + v(y-h,t)}{h^2} + O(h^2) , \end{aligned}$$

respectively, we may formulate the approximation of (9.4) and (9.5) as an initial value problem for the unknown function v through the method of lines. Namely,

$$v_t(t) = Av(t) + g(v(t)), \quad 0 < t < T , \tag{9.7}$$

$$v(0) = v_0 , \tag{9.8}$$

where $v(t) = (u_1(t), u_2(t), \ldots, u_N(t))^T$, $g(v(t)) = (f(u_1(t)), f(u_2(t)), \ldots, f(u_N(t)))^T$ and $v_0 = (u_0(t_1), u_0(t_2), \ldots, u_0(t_N))^T$. The matrix A is generated from the semidiscretization process. It is nonsingular in most cases, and is symmetric and negative definite when $c \equiv 0$ according to the central difference approximation used.

The formal solution of (9.7) and (9.8) can be expressed as

$$v(t) = E(tA)v_0 + \int_0^t E((t-\tau)A)g(v(\tau))d\tau, \quad 0 < t < T , \tag{9.9}$$

where $E(\xi Q) = \exp(\xi Q)$ is the analytic semigroup generated.

Formula (9.9) indicates good chances to construct highly efficient time integrators for solving systems of ordinary differential equations, regardless of any particular spatial discretization adopted. For instance, we may consider certain adaptive Runge–Kutta methods, in which variable time steps are generated through proper arc-length mechanisms. The matrix exponential operators obtained can be approximated by L-stable rational approximations [18, 31]. A consistent compound adaptive method may be developed based on the above considerations to assure an accuracy in the computation that will not be affected by the existing singularities. The computation of the numerical solution may well reflect the special feature of quenching singularities.

Let inequalities between vectors be in the componentwise sense. We consider the numerical analog of the formal solution (9.9) through the adaptive two-stage Runge–Kutta integrator:

$$w^{(1)} = v_0 \,, \tag{9.10}$$

$$w^{(2)} = R_0^{(2)}(\tau A)v_0 + \tau R_1^{(2)}(\tau A)g_1 \,, \tag{9.11}$$

$$\begin{aligned} w^{(3)} &= R_0^{(3)}(\tau A)v_0 + \tau \left\{ \left(\kappa R_1^{(3)}(\tau A) + (1-2\kappa)R_2^{(3)}(\tau A) \right) g_1 \right. \\ &\quad \left. + \left((1-\kappa)R_1^{(3)}(\tau A) + (2\kappa - 1)R_2^{(3)}(\tau A) \right) g_2 \right\} \,, \end{aligned} \tag{9.12}$$

$$v_1 = w^{(3)} \,, \tag{9.13}$$

where $0 \leq \kappa \leq 1$,

$$g_k = g\left(w^{(k)}(\tau) \right), \quad k = 1, 2 \,, \tag{9.14}$$

$$R_j^{(i)}(z) = \left(R_{j-1}^{(i)}(z) - I \right) z^{-1}, \quad i = 2, 3, \ j = 1, 2 \,, \tag{9.15}$$

and $R_0^{(i)}, i = 2, 3$, are proper approximations to E. The temporal discretization parameter, τ, will be determined through a properly defined adaptive mechanism during the computation. Note that functions $R_j^{(i)}(z),\ j = 1, 2$, possess the same denominator as $R_0^{(i)}(z)$. In fact, the factor z^{-1} can be canceled out during actual computations if $R_0^{(i)},\ i = 2, 3$, are properly chosen. This implies that the constructed algorithm is valid even when A is singular. The linearized formula offers a direct way for computing solutions of (9.1) through (9.3). The Runge–Kutta time integrator is stable when the real parts of the eigenvalues of A are negative.

Our purpose is to derive a method of consistency order two. To this end, at the same time of adopting the above Runge–Kutta process, we use an L-stable second-order rational approximation $R_0^{(i)},\ i = 2, 3$, for E.

9.2.2 The Adaptive Algorithms

For the algorithm (9.10) through (9.15), we consider rational approximations to E, in particular the second-order L-stable approximation with distinct real poles (see [18, 31] for details),

$$R(z) = \frac{w_1}{1 - b_1 z} + \frac{w_2}{1 - b_2 z} \,,$$

where $b_1 = 1/9,\ b_2 = 1/3,\ w_1 = -8,\ w_2 = 9$. By letting $A_1 = I - \frac{\tau}{4}A,\ A_2 = I - \frac{\tau}{3}A,\ \tau > 0$, based on the function R, we may define

$$R_0^{(i)}(\tau A) = -8A_1^{-1} + 9A_2^{-1} = \left(I + \frac{5\tau}{12}A \right) A_1^{-1} A_2^{-1}, \quad i = 2, 3 \,. \tag{9.16}$$

It follows from (9.15) and (9.16) that

$$R_1^{(2)}(\tau A) = -2A_1^{-1} + 3A_2^{-1} = \frac{1}{12}(12I - \tau A)A_1^{-1}A_2^{-1}\,, \tag{9.17}$$

$$R_1^{(3)}(\tau A) = R_1^{(2)}(\tau A)\,, \tag{9.18}$$

$$R_2^{(3)}(\tau A) = -\frac{1}{2}A_1^{-1} + A_2^{-1} = \frac{1}{12}(6I - \tau A)A_1^{-1}A_2^{-1}\,. \tag{9.19}$$

Let $t_k = k\tau$, $k = 0, 1, \ldots$, and $v_k = v(t_k)$, $k = 1, 2, \ldots$, be numerical solutions of (9.7) and (9.8) obtained by using the two-stage method (9.10) through (9.15). We denote

$$w^{(1)} = v_k, \quad k = 0, 1, \ldots\,, \tag{9.20}$$

$$w^{(2)} = R_0^{(2)}(\tau A)w^{(1)} + \tau R_1^{(2)}(\tau A)g_1\,. \tag{9.21}$$

Then the numerical solution can be formulated in the following embedded form,

$$v_{k+1} = R_0^{(3)}(\tau A)w^{(1)} + \tau\beta\,, \tag{9.22}$$

where v_0 is the initial value and

$$\beta = \left(\kappa R_1^{(3)}(\tau A) + (1 - 2\kappa)R_2^{(3)}(\tau A)\right)g_1 + \Big((1 - \kappa)R_1^{(3)}(\tau A) + (2\kappa - 1)R_2^{(3)}(\tau A)\Big)g_2\,.$$

Substituting (9.17) through (9.19) into (9.20) through (9.22), we obtain readily that

$$w^{(1)} = v_k, \quad k = 0, 1, \ldots\,, \tag{9.23}$$

$$A_1A_2w^{(2)} = \left(I + \frac{5\tau}{12}A\right)w^{(1)} + \frac{\tau}{12}(12I - \tau A)g_1\,, \tag{9.24}$$

$$A_1A_2v_{k+1} = \left(I + \frac{5\tau}{12}A\right)w^{(1)} + \tau\gamma\,, \tag{9.25}$$

where $\gamma = A_1A_2\beta$. Since A is a tridiagonal matrix, thus A_1A_2 is of quindiagonal. However, this will not add any extra cost in solving (9.23) through (9.25). In fact, we may observe that by denoting

$$p = \left(I + \frac{5\tau}{12}A\right)w^{(1)} + \frac{\tau}{12}(12I - \tau A)g_1\,;$$

$$q = \left(I + \frac{5\tau}{12}A\right)w^{(1)} + \tau\gamma\,,$$

systems of linear equations (9.23) through (9.25) can be reformulated as

$$w^{(1)} = v_k, \quad k = 0, 1, \ldots\,, \tag{9.26}$$

$$A_1y^{(1)} = p, \quad A_2w(2) = y^{(1)}\,, \tag{9.27}$$

$$A_1y^{(2)} = q\,, \tag{9.28}$$

$$A_2v_{k+1} = y^{(2)}, \quad k = 0, 1, \ldots\,, \tag{9.29}$$

which is of clearly tridiagonal. We have the following:

LEMMA 9.1
Let $c(x),\ 0 \le x \le 1$, *be continuous and let* $\sigma = \max_{0\le x\le a} |c(x)|$. *If*

(i) $\sigma = 0$, *or*

(ii) $h \le 2/\sigma$ *and* $(6h^2 + 2\tau)/h\tau \ge c(ah), -c(aNh), \sigma \ne 0$,

then matrices A_1, A_2 *are monotone and thus nonsingular. Their inverses are positive and monotone.*

PROOF Let $a^{(1)}_{i,j}, a^{(2)}_{i,j}, i, j = 1, 2, \ldots, N$, be elements of A_1, A_2, respectively. It is not difficult to see that both A_1, A_2 are irreducible. We only need to give a detailed proof regarding A_1 since the proof of the case for A_2 will be similar. The case when $\sigma = 0$ is obvious. Therefore, we may assume that $\sigma \ne 0$ throughout our discussion. We observe that for nontrivial elements of A_1 we have

$$a^{(1)}_{i,i} = 1 + \tau/2h^2, \quad i = 1, 2, \ldots, N ; \tag{9.30}$$

$$a^{(1)}_{i,i+1} = -\tau(2 + hc(aih))/8h^2, \quad i = 1, 2, \ldots, N-1 ; \tag{9.31}$$

$$a^{(1)}_{i+1,i} = -\tau(2 - hc(aih))/8h^2, \quad i = 2, 3, \ldots, N . \tag{9.32}$$

To meet the criteria $a^{(1)}_{i,j} \le 0,\ i \ne j$, it is necessary to have that $2 \pm hc(aih) \ge 0$, $i = 1, 2, \ldots, N-1$, [14]. These imply that

$$\frac{1}{h} \ge \frac{c(aih)}{2}, \frac{1}{h} \ge -\frac{c(aih)}{2},\ i = 1, 2, \ldots, N . \tag{9.33}$$

Further, it is found that

$$\sum_{j=1}^{N} a^{(1)}_{i,j} = 1,\ i = 2, 3, \ldots, N-1 ,$$

and

$$\sum_{j=1}^{N} a^{(1)}_{1,j} = 1 + \frac{\tau}{4h^2} - \frac{\tau c(ah)}{8h} ;$$

$$\sum_{j=1}^{N} a^{(1)}_{N,j} = 1 + \frac{\tau}{4h^2} + \frac{\tau c(aNh)}{8h} .$$

To let the above sums be nonnegative, respectively, it is necessary that

$$1 + \frac{\tau}{4h^2} - \frac{\tau c(ah)}{8h}, 1 + \frac{\tau}{4h^2} + \frac{\tau c(aNh)}{8h} \ge 0 . \tag{9.34}$$

Together with (9.33), condition (9.34) ensures that A_1 is monotone and A_1^{-1} is positive and thus monotone.

As for A_2, similarly, for nontrivial elements we have

$$\begin{aligned} a^{(2)}_{i,i} &= 1 + 2\tau/3h^2,\ i = 1, 2, \ldots, N\ ; \\ a^{(2)}_{i,i+1} &= -\tau(2 + hc(aih))/6h^2,\ i = 1, 2, \ldots, N-1\ ; \\ a^{(2)}_{i+1,i} &= -\tau(2 - hc(aih))/6h^2,\ i = 2, 3, \ldots, N\ . \end{aligned}$$

It follows that A_2 and A_2^{-1} are monotone if (9.33) and

$$1 + \frac{\tau}{3h^2} - \frac{\tau c(ah)}{6h}, 1 + \frac{\tau}{3h^2} + \frac{\tau c(aNh)}{6h} \geq 0\ . \tag{9.35}$$

Inequality (9.33) suggests the first condition in (ii), and a combination of (9.34) and (9.35) gives the rest of constraints. Hence the proof is completed. ∎

Further, we may prove the following.

THEOREM 9.1

Given $0 \leq u_0 << 1$ *and* $0 \leq \kappa \leq 1$, *let* $c(x), 0 \leq x \leq 1$, *be continuous and let* $\sigma = \max_{0 \leq x \leq a} |c(x)|$. *If*

(i) $\sigma = 0$, *or*

(ii) $h \leq 2/\sigma, (6h^2 + 2\tau)/h\tau \geq c(ah), -c(aNh)$ *and* $\tau/h^2 \leq 6/5,\ \sigma \neq 0$,

then solution vectors of (9.26) *through* (9.29), $\{v_k\}_{k=0}^{\infty}$,

(1) *form a monotonically increasing sequence;*

(2) *increase monotonically till unity is exceeded by an element of the solution vector, or converges to the steady solution of the problem* (9.7) *and* (9.8).

In the latter case, we do not have a quenching solution.

PROOF The theorem is obvious when $\sigma = 0$. Assuming $\sigma \neq 0$, we first consider the vector p. It is not difficult to see that nontrivial entries of the matrix function $B = I + \frac{5\tau}{12}A$ are

$$\begin{aligned} b_{i,i} &= 1 - 5\tau/6h^2,\ i = 1, 2, \ldots, N\ ; \\ b_{i,i+1} &= 5\tau(2 + hc(aih))/24h^2,\ i = 1, 2, \ldots, N-1\ ; \\ b_{i+1,i} &= 5\tau(2 - hc(aih))/24h^2,\ i = 2, 3, \ldots, N\ . \end{aligned}$$

Therefore, $B > 0$ if

$$\tau/h^2 \leq 6/5, h \leq 2/|c(aih)|,\ i = 1, 2, \ldots, N\ . \tag{9.36}$$

Further, we observe that the matrix $12I - \tau A$ is monotone if conditions (9.33) and (9.34) are satisfied.

On the other hand, according to its definition, $g(v)$ is an increasing function of $v, 0 \leq v < 1$, with a minimum value $g(0) = 1$. We may show that

$$0 \leq w^{(1)} \leq w^{(2)} < 1 \, . \tag{9.37}$$

According to (9.26) and (9.27), we have

$$A_2 A_1 w^{(2)} = p \, ,$$

where $A_2 A_1$ is monotone. Let $w = (1, 1, \ldots, 1)^T$ be the vector with N unit components, we have

$$A_2 A_1 \left(w - w^{(2)} \right) = A_2 A_1 w - p = A_2 A_1 w - B u_k - \frac{\tau}{12}(12I - \tau A) g_1 \, .$$

The right-hand side of the above equation is equivalent to

$$\begin{aligned} s &= \left(I - \frac{\tau}{4} A \right) \left(I - \frac{\tau}{3} A \right) w - \frac{\tau}{12}(12\tau - \tau A) w \\ &= \left\{ I - \frac{7\tau}{12} A + \frac{\tau^2}{12} A(I + A) \right\} w \, , \end{aligned}$$

where

$$\begin{aligned} s_1 &= 1 + \frac{\tau}{24h^2} \left[7 + \tau \left(2 - h^2 \right) \right] (2 - hc(ah)) \geq 0 \, ; \\ s_N &= 1 + \frac{\tau}{24h^2} \left[7 + \tau \left(2 - h^2 \right) \right] (2 + hc(aNh)) \geq 0 \, ; \\ s_j &= 1, \quad j = 3, 4, \ldots, N - 2 \, , \end{aligned}$$

and furthermore,

$$\begin{aligned} s_2 &= 1 - \frac{\tau^2}{48h^4}(2 - hc(ah))(2 - hc(2ah)) \\ &\geq 1 - \frac{\tau^2}{48h^4}(2 + h\sigma)^2 \geq 1 - \frac{4^2}{48} \times \left(\frac{6}{5} \right)^2 > 0 \, ; \\ s_{N-1} &= 1 - \frac{\tau^2}{48h^4}(2 + hc(a(N-1)h))(2 + hc(2aNh)) \\ &\geq 1 - \frac{\tau^2}{48h^4}(2 + h\sigma)^2 \geq 1 - \frac{4^2}{48} \times \left(\frac{6}{5} \right)^2 > 0 \, . \end{aligned}$$

The above investigation indicates that $A_2 A_1 \left(w - w^{(2)} \right) > 0$ and this implies that

$$w^{(2)} < 1 \, . \tag{9.38}$$

At the same time, we observe that

$$v_0 \le Bv_0 + \frac{\tau}{12}(12I - \tau A)g_1 = p \le A_1^{-1}p = y^{(1)} \le A_2^{-1}y^{(1)} = w^{(2)}$$

under condition (ii). Recall (9.26). From the above and (9.38), inequalities (9.37) become obvious.

It can be shown that $\gamma \ge 0$. To see this, we set $\gamma = \Gamma_1 g_1 + \Gamma_2 g_2$, where

$$\Gamma_1 = \frac{1}{12}(6I + \tau(\kappa - 1)A), \Gamma_2 = \frac{1}{12}(6I - \tau\kappa A), 0 \le \kappa \le 1, \tau > 0\ .$$

Recall the equality $A = \frac{4}{\tau}(I - A_1)$ and relations (9.30) through (9.32). We find immediately that $\Gamma_1, \Gamma_2 > 0$ and it follows that $\gamma \ge 0$.

Based upon the previous discussions, we may conclude that

$$\begin{aligned} 0 \le v_0 = w^{(1)} \le Bw^{(1)} &< \left(I + \frac{5\tau}{12}A\right) w^{(1)} + \tau\gamma \le A_1^{-1}q_i \\ &= y^{(2)} \le A_2^{-1}y^{(2)} = v_1\ , \end{aligned}$$

for the positivity of A_1^{-1}, A_2^{-1}, and B. Next, by replacing $w^{(1)}, w^{(2)}$ with more general notations $w^{(1,k)}, w^{(2,k)}$, respectively, and g_1, g_2 by $g_{1,k}, g_{2,k}$, respectively, in the computation of u_{k+1} from u_k, subsequently we obtain that

$$\begin{aligned} A_1A_2\,(v_{k+1} - v_k) =& B\,(v_k - v_{k-1}) \\ &+ \tau A_1^{-1}A_2^{-1}\left\{\Gamma_1(g_{1,k} - g_{1,k-1}) + \Gamma_2(g_{2,k} - g_{2,k-1})\right\}\ , \\ k =& 1, 2, \dots\ . \end{aligned}$$

Recall that $v_1 - v_0 > 0$, $g_{j,1} - g_{j,0} > 0$, $j = 1, 2$, and the fact that A_1A_2 is monotone and Γ_j, $j = 1, 2$, are positive under conditions given by the theorem. An inductive argument leads immediately to

$$v_0 < v_1 < v_2 < v_3 < \cdots < v_k < \cdots < 1\ ,$$

if $k\tau < T_a$ and the sequence exceeds unity if $k\tau \ge T_a$ by Nagumo's lemma [1]. ∎

It may be interesting to see the constraints in h and τ. The restrictions are necessary to guarantee the monotonicity required for approximating the solution of nonlinear quenching problems, as discussed in various publications (see [27, 28], for instance).

Now we consider a modified arc-length adaptive mechanism in time. Let $u_t(hk, t)$ be the time derivative of the solution of (9.4) and (9.5) at (hk, t), $k = 0, 1, \dots, N+1$. When $t - \Delta t > 0$, $0 < \Delta t << 1$, the arc-length of the function u_t between $(hk, t - \Delta t)$, (hk, t) can be approximated by $\left[(\Delta t)^2 + (u_t(hk, t) - u_t(hk, t - \Delta t))^2\right]^{\frac{1}{2}}$. Let $\tau < 1$ be the given initial time step size, and $\bar{\tau}_k$ be the time step reference for determining the actual time step size to be used in the next step computation. We require that

$$\frac{\sqrt{(\Delta t)^2 + (u_t(hk, t) - u_t(hk, t - \Delta t))^2}}{\Delta t} = \frac{\tau}{\bar{\tau}_k}\ .$$

It follows immediately that

$$\begin{aligned}\bar{\tau}_k =& \frac{\tau\,\Delta t}{\sqrt{(\Delta t)^2 + (u_t(hk,t) - u_t(hk,t-\Delta t))^2}} \\ =& \frac{\tau}{\sqrt{1 + \left(\frac{u_t(hk,t) - u_t(hk,t-\Delta t)}{\Delta t}\right)^2}} \leq \tau\ . \end{aligned} \tag{9.39}$$

It is observed that when $\Delta t \to 0$, we have

$$\bar{\tau}_k \to \frac{\tau}{\sqrt{1 + u_{tt}^2(hk,t)}}$$

which is similar to the monitoring function used in traditional adaptive algorithms [9, 10]. However, our arc-length adaptive mechanism is established based on the function u_t rather than u.

In practical computations, function values of u_t can be conveniently obtained through Equation (9.4), together with proper difference approximations. Instead of sophisticated smoothing processes, we may introduce a minimal time step size controller τ_0, $0 < \tau_0 << \tau$, to avoid unnecessarily large numbers of computations immediately before the blow-up of u_t and to indicate a proper stopping time for the computation. Under the consideration, the actual time step size used for computing the solution at a higher time level, $u(hk, t + \widetilde{\Delta t})$, can be determined uniquely by the following formula,

$$\widetilde{\Delta t} = \max\left\{\tau_0, \min_k\{\bar{\tau}_k\}\right\}\ . \tag{9.40}$$

9.3 The Fully Adaptive Algorithm

9.3.1 The Discretization

We now rewrite (9.4) and (9.5) into the following uniformed form:

$$\begin{aligned} u_t =& \frac{1}{a^{q+2}y^q}u_{yy} + \frac{c(ay)}{a^{q+1}y^q}u_y + \frac{1}{a^q y^q}f(u), \\ & 0 < y < 1,\ 0 < t < T\ , \end{aligned} \tag{9.41}$$

$$\begin{aligned} u(ay,0) =& u_0,\ 0 \leq y \leq 1;\ u(0,t) = u(a,t) = 0, \\ & 0 < t < T,\ q \geq 0\ . \end{aligned} \tag{9.42}$$

Given $N > 0$. We define a nonuniform partition Ω_N over the interval $[0, 1] : \Omega_N = \{y_0, y_1, \ldots, y_{N+1}\}$, where $y_0 = 0, y_j = y_{j-1} + h_j, h_j > 0, j = 1, 2, \ldots, N +$

1, and $y_{N+1} = 1$. Further, based on Ω_N, we introduce the following difference approximations for the first and second order derivatives in space, respectively,

$$D_+ w_j = \frac{w_{j+1} - w_j}{h_{j+1}}, \quad j = 1, 2, \ldots, N ; \tag{9.43}$$

$$D_+ D_- w_j = \frac{1}{h_{j+1}\tilde{h}_j} w_{j+1} - \frac{2}{h_j h_{j+1}} w_j + \frac{1}{h_j \tilde{h}_j} w_{j-1}, \quad j = 1, 2, \ldots, N , \tag{9.44}$$

where $\tilde{h}_j = (h_j + h_{j+1})/2$ and $w_j = w(y_j, t)$. Denote $u_j(t)$ as an approximation of the exact solution of (9.1) and (9.2) at the grid point (y_j, t), $j = 0, 1, \ldots, N+1$, and let $v(t) = (u_1(t), u_2(t), \ldots, u_N(t))^T$. It follows that, by replacing the spatial derivatives with the above differences and removing higher order truncation error terms, we arrive at the following system of nonlinear semidiscretized equations corresponding to (9.1) and (9.2):

$$v_t(t) = Av(t) + g(v(t)), \quad 0 < t < T , \tag{9.45}$$

$$v(0) = v_0 , \tag{9.46}$$

where $g(v(t)) = (f(u_1(t)), f(u_2(t)), \ldots, f(u_N(t)))^T$, $v_0 = (u_0(y_1), u_0(y_2), \ldots, u_0(y_N))^T$, and $A \in \mathcal{R}^{N\times N}$ is nonsingular. The order of accuracy of (9.2) and (9.3) is of one unless $h_j = h > 0$, $j = 1, 2, \ldots, N+1$, in which we have the order two. Similar to (9.10) through (9.13), a discrete analog of the abstract solution formula (9.9) can be given via the two-stage second-order adaptive Runge–Kutta method by the following:

$$w_k^{(1)} := \tilde{v}_k , \tag{9.47}$$

$$w_k^{(2)} := R_0^{(2)}(\tau_k A) w_k^{(1)} + \tau_k R_1^{(2)}(\tau_k A) g_1 , \tag{9.48}$$

$$w_k^{(3)} := R_0^{(3)}(\tau_k A) w_k^{(1)} + \tau_k \left\{ \left(\kappa R_1^{(3)}(\tau_k A) + (1 - 2\kappa) R_2^{(3)}(\tau_k A) \right) g_1 + \left((1-\kappa) R_1^{(3)}(\tau_k A) + (2\kappa - 1) R_2^{(3)}(\tau_k A) \right) g_2 \right\} , \tag{9.49}$$

$$\tilde{v}_{k+1} := w_k^{(3)}, \quad k = 0, 1, \ldots, K , \tag{9.50}$$

where $\tilde{v}_k$ is the approximation of v_k, $0 \le \kappa \le 1$, $\tau_k > 0$. Functions g_i, $R_j^{(\ell)}(z)$, $i, j = 1, 2$, $\ell = 2, 3$, are defined through (9.14) and (9.15). The algorithm offers a possible access for computing the solution of (9.41) and (9.42) through moving grid in time. The stability of the Runge–Kutta time integrator is again guaranteed when real parts of the eigenvalues of A are nonpositive. The temporal discretization parameter, τ_k, will be determined through our modified arc-length adaptive mechanism during the computation.

Let $\wedge$ be one of the operations $<, \le, >, \ge$, and $\alpha, \beta \in \mathcal{R}^N$. We introduce the following notations:

1. $\alpha \wedge \beta$ means $\alpha_i \wedge \beta_i$, $i = 1, 2, \ldots, N$;

2. $\alpha \wedge a$ means $\alpha_i \wedge a$, $i = 1, 2, \ldots, N$, for any given $a \in \mathcal{R}$.

9.3.2 The Monotone Convergence

Consider the following second-order L-acceptable rational approximation with distinct real poles (see [29, 31] for details),

$$R(z) = \frac{1 + a_1 z}{(1 - b_1 z)(1 - b_2 z)} ,$$

where

$$\begin{aligned} b_1 + b_2 + a_1 &= 1 , \\ b_1 + b_2 - b_1 b_2 &= \frac{1}{2} , \\ a_1, b_1, b_2 &> 0 . \end{aligned}$$

Similar to (9.16) through (9.19), by denoting $t_k = \sum_{k=0}^{K} \tau_k$, and letting $v_k = v(t_k)$ be the numerical solution of (9.45) and (9.46) obtained through the adaptive procedure (9.47) through (9.50) at stage t_k, we may show that the numerical solution satisfies the following procedure:

$$w_k^{(1)} = v_k , \tag{9.51}$$

$$w_k^{(2)} = R_0^{(2)}(\tau_k A) w_k^{(1)} + \tau_k R_1^{(2)}(\tau_k A) g_1 . \tag{9.52}$$

$$v_{k+1} = R_0^{(3)}(\tau_k A) w_k^{(1)} + \tau_k \beta_k , \tag{9.53}$$

where v_0 is the initial vector from (9.46) and

$$\begin{aligned} \beta_k = &\left(\kappa R_1^{(3)}(\tau_k A) + (1 - 2\kappa) R_2^{(3)}(\tau_k A) \right) g_1 \\ &+ \left((1 - \kappa) R_1^{(3)}(\tau_k A) + (2\kappa - 1) R_2^{(3)}(\tau_k A) \right) g_2 . \end{aligned}$$

We subsequently obtain that

$$w_k^{(1)} = v_k , \tag{9.54}$$

$$A_1 A_2 w_k^{(2)} = (I + a_1 \tau_k A) w_k^{(1)} + \tau_k (I - b_1 b_2 \tau_k A) g_1 , \tag{9.55}$$

$$A_1 A_2 v_{k+1} = (I + a_1 \tau_k A) w_k^{(1)} + \tau_k A_1 A_2 \beta_k, \ k = 0, 1, \ldots, K . \tag{9.56}$$

Note that A is of tridiagonal. In fact, for nonzero elements of $A = [a_{i,j}]_{i,j=1,\ldots,N}$, we may show that

$$\begin{aligned} a_{j,j} &= -\left(2 + a h_j c(a y_j)\right) \left(a^{q+2} y_j^q h_j h_{j+1} \right)^{-1} , \ j = 1, 2, \ldots, N ; \\ a_{j,j+1} &= \left(1 + a \tilde{h}_j c(a y_j)\right) \left(a^{q+2} y_j^q h_{j+1} \tilde{h}_j \right)^{-1} , \ j = 1, 2, \ldots, N - 1 ; \\ a_{j,j-1} &= \left(a^{q+2} y_j^q h_j \tilde{h}_j \right)^{-1} , \ j = 2, 3, \ldots, N . \end{aligned}$$

It follows immediately that $A_1 A_2$ is of quindiagonal.

Define

$$\begin{aligned} p_k &= (I + a_1\tau_k A)\, w_k^{(1)} + \tau_k(I - b_1 b_2 \tau_k A) g_1 \ , \\ q_k &= (I + a_1\tau_k A)\, w_k^{(1)} + \tau_k A_1 A_2 \beta_k \ . \end{aligned}$$

Systems (9.54) through (9.56) can be then simplified into the following:

$$w_k^{(1)} = v_k \ , \tag{9.57}$$

$$A_1 \xi_k^{(1)} = p_k, A_2 w_k^{(2)} = \xi^{(1)} \ , \tag{9.58}$$

$$A_1 \xi_k^{(2)} = q_k, A_2 v_{k+1} = \xi_k^{(2)}, \ k = 0, 1, \ldots, K \ . \tag{9.59}$$

LEMMA 9.2
Let function $c(x)$ be continuous on $[0, a]$, and $\sigma = \max_{0\le x\le a} |c(x)|$. If

(i) *$\sigma = 0$, or*

(ii) *$h_j < \frac{1}{\sigma a}, \ j = 1, 2, \ldots, N+1, \sigma \neq 0$,*

then the real parts of the eigenvalues of A are nonpositive. Further, at least one of the real parts is negative.

PROOF The proof is straightforward according to Gerschgorin theorem [16]. ∎

LEMMA 9.3
Let function $c(x)$ be continuous on $[0, a]$, and σ be the same as defined in Lemma 9.2. If

(i) *$\sigma = 0$, or*

(ii) *$h_j < \frac{1}{\sigma a}, \ j = 1, 2, \ldots, N+1, \sigma \neq 0$,*

then the matrices A_1, A_2 are monotone and nonsingular. Their inverses are positive.

PROOF The properties can be shown directly according to the structures of the matrices A_1, A_2. ∎

LEMMA 9.4
Let function $c(x)$ be continuous on $[0, a]$, and $\sigma = \max_{0\le x\le a} |c(x)|$. If

(i) *$\sigma = 0$, or $h_j < \frac{1}{\sigma a}, \ j = 1, 2, \ldots, N+1, \sigma \neq 0$,*

(ii) *$0 \le w^{(1)}, w^{(2)} < 1$,*

(iii) *$Aw^{(1)} + g(w^{(1)}) > 0$,*

then $v_k = w^{(1)} < w^{(2)} < v_{k+1}$. *Thus the sequence* $\{\tilde{v}_k\}$ *is monotonically increasing.*

PROOF Recalling the definition of A_1, A_2, we have

$$\begin{aligned} A_1A_2\left(w^{(2)} - w^{(1)}\right) &= \tau_k(I - b_1b_2\tau_k A)\left(Aw^{(1)} + g\left(w^{(1)}\right)\right), \\ A_1A_2\left(v_{k+1} - w^{(2)}\right) &= \tau_k\left(\frac{1}{2}I - \tau_k\kappa b_1b_2A\right)(g_2 - g_1). \end{aligned}$$

It follows that

$$\begin{aligned} A_1^{-1}(I - b_1b_2\tau_k A) &= (1 - b_2)A_1^{-1} + b_2I > 0, 0 < b_2 < \frac{1}{2}, \\ A_1^{-1}\left(\frac{1}{2}I - \tau_k\kappa b_1b_2A\right) &= \left(\frac{1}{2} - \kappa b_2\right)A_1^{-1} + \kappa b_2 I > 0. \end{aligned}$$

Thus, by previous lemmas, we obtain immediately that $v_k = w^{(1)} < w^{(2)} < v_{k+1}$ and this shows the required monotonicity. ∎

LEMMA 9.5

Let function $c(x)$ *be continuous on* $[0, a]$, *and let* $\sigma = \max_{0\le x\le a}|c(x)|$. *If*

(i) $\sigma = 0$, *or* $h_i < \min\left\{\frac{1}{\sigma a}, \frac{1}{Ma^{q+2}}\right\}$, $i = 1, 2, \ldots, N+1$,

(ii) $h_1h_2,\ h_{N-1}h_N < \frac{1}{2\xi a^{q+2}}$,

(iii) *there exists a constant* c_1 *with* $0 < c_1 < 1$ *such that*

$$\frac{1}{a^{2q+4}y_1^{2q}h_1h_2\tilde{h}_1\tilde{h}_2}, \frac{1}{a^{2q+4}y_{N-1}^{2q}\tilde{h}_{N-1}h_N\tilde{h}_Nh_{N+1}} < \frac{c_1}{\tau_0^2},$$
$$\tau_0 \le \min\left\{\frac{1}{2\xi}(1 - c_1), \frac{2}{M + \xi}(1 - c_1)\right\},$$

where $\xi = f(0)$ *and* $M = f(1/2)$, *then for any null vector* $w^{(1)}$ *we have* $w^{(2)} < 1/2, \tilde{v}_1 < 1$.

PROOF Let $w = (1, 1, \ldots, 1)^T$. We first show that $w^{(2)} < 1/2$ under the conditions given. Denote

$$\begin{aligned} s &= A_1A_2\left(\frac{1}{2}w - w^{(2)}\right) \\ &= \frac{1}{2}A_1A_2w - (I + a_1\tau_0A)w^{(1)} - \tau_0(I - b_1b_2\tau_0A)g_1 \\ &= \left\{\left(\frac{1}{2} - \xi\tau_0\right)I - \frac{b_1 + b_2}{2}\tau_0A + \frac{1}{2}b_1b_2\tau_0^2A(2\xi I + A)\right\}w. \end{aligned}$$

It can be shown that s_j, $j = 1, 2, \ldots, N$, are non-negative. This indicates that

$$A_1 A_2 \left(\frac{1}{2} w - w^{(2)} \right) > 0 \, .$$

Further, the monotonicity of A_1, A_2 leads to the conclusion that $\frac{1}{2} w - w^{(2)} > 0$ and this implies that $w^{(2)} < \frac{1}{2}$.

Second, we may show that $v_1 < 1$. It is observed that

$$\begin{aligned} A_1 A_2 (w - v_1) &= A_1 A_2 w - (I + a_1 \tau_0 A) w^{(1)} - \tau_0 \left(\frac{1}{2} I + \tau_0 (\kappa - 1) b_1 b_2 A \right) g_1 \\ &\quad - \tau_0 \left(\frac{1}{2} I - \tau_0 \kappa b_1 b_2 A \right) g_2 \, . \end{aligned}$$

Recall that $w^{(2)} < \frac{1}{2}$ and $g_2 = g(w^{(2)}) < M$. Similar to the proof of Lemma 9.4, we deduce that

$$A_1^{-1} \left(\frac{1}{2} I - \tau_0 \kappa b_1 b_2 A \right) > 0 \, .$$

It follows that

$$\begin{aligned} A_2 (w - v_1) \geq & A_1^{-1} (I - b_1 \tau_0 A)(I - b_2 \tau_0 A) w \\ & - \tau_0 A_1^{-1} \left(\frac{1}{2} I + \tau_0 (\kappa - 1) b_1 b_2 A \right) \xi w \\ & - \tau_0 A_1^{-1} \left(\frac{1}{2} I - \tau_0 \kappa b_1 b_2 A \right) M w = A_1^{-1} \tilde{s} \, , \end{aligned}$$

where

$$\begin{aligned} \tilde{s} &= \left[(I - b_1 \tau_0 A)(I - b_2 \tau_0 A) - \tau_0 \left(\frac{1}{2} I + \tau_0 (\kappa - 1) b_1 b_2 A \right) \xi \right. \\ &\quad \left. - \tau_0 \left(\frac{1}{2} I - \tau_0 \kappa b_1 b_2 A \right) M I \right] w \\ &= \left\{ \left[1 - \frac{\tau_0}{2} (M + \xi) \right] I - (b_1 + b_2) \tau_0 A + b_1 b_2 \tau_0^2 A [A + (\kappa M - \kappa \xi + \xi) I] \right\} w \, . \end{aligned}$$

Noting that $1 - (\kappa M - \kappa \xi + \xi) a^{q+2} y_1^q h_1 h_2 \geq 1 - M a^{q+2} h_1 h_2 > 0$, we have

$$\begin{aligned} \tilde{s}_1 &= 1 - \frac{\tau_0}{2} (M + \xi) + \frac{b_1 + b_2}{a^{q+2} y_1^q h_1 \tilde{h}_1} \tau_0 + \frac{b_1 b_2 \tau_0^2}{a^{2q+4} y_1^{2q} h_1^2 \tilde{h}_1 h_2} \\ &\quad \times \left[2 + a h_1 c (a y_1) - (\kappa M - \kappa \xi + \xi) a^{q+2} y_1^q h_1 h_2 \right] > 0 \, . \end{aligned}$$

By the same token, we may show that $\tilde{s}_j$, $j = 2, 3, \ldots, N$, are non-negative and this completes the proof. ∎

By means of the above lemmas, we have

THEOREM 9.2
Let function $c(x)$ be continuous on $[0, a]$, and let $\sigma = \max_{0\le x\le a} |c(x)|$. Assume that $\{v_k\}_{k=0}^{\infty}$ be a solution sequence given by the two-stage scheme (9.57) *through* (9.59). *If*

(i) $\sigma = 0$, *or* $h_j < \frac{1}{\sigma a}$, $j = 1, 2, \ldots, N+1, \sigma \neq 0$,

(ii) $Av + g(v) > 0$ *for* $v < 1$,

then $\{v_k\}_{k=0}^{\infty}$

(1) *forms a monotonically increasing sequence,*

(2) *increases monotonically until unity is exceeded by an element of the solution vector, or converges to the steady solution of the problem* (9.7) *and* (9.8).

In the later case, we do not have a quenching solution.

The adaptive grid distribution over the interval [0, 1] is determined by a modified arc-length adaptive principal. As stated before, the interval can be partitioned through $\{y_j,\ j = 0, 1, \ldots, N, N+1 : y_0 = 0, \ldots, N\}$, can be obtained via following grid equations

$$\int_{y_j}^{y_{j+1}} M(x,t)dx = \frac{1}{N}\int_0^1 M(x,t)\,dx,\ \ 0 \le j \le N\ ,$$

under a proper smoothness process. The adaptation in time can be achieved through an approach similar to [26].

9.3.3 The Error Control and Stopping Criterion

It has been essential to estimate the computational error development during the calculation. The information obtained is not only used for determining the time to stop, but also for optimizing the computation procedures. Practically, an error estimate formula for monitoring the local relative error at each time step is widely adopted. In this chapter, we consider a Milne-alike device for achieving this. The error assessment obtained is then used to help updating temporal discretization parameters τ_k and constructing a reasonable stopping criteria.

There are two frequently used error controlling strategies over the local error for a given tolerance $\epsilon > 0$. One is the *error control per step,* while the other is the *error control per unit step* including the standard Milne device [16]. Most of the traditional methods, including the Milne device, are not suitable to be used directly for adaptive time-step methods due to the fact that they require executing two numerical methods at the same time over a nonuniform grid.

Based on the fact that nonuniform grids spread both in the time and space directions in our applications, we adopt a Milne-alike mechanism for the local error

estimate. Let u_k be the numerical solution of (9.41) and (9.42) obtained through (9.57) through (9.59) at the time level $t_k, 0 < k \leq K$. Then the local error for computing u_{k+1} at $t_{k+1} = t_k + \tau$ is defined as

$$u(t_k + \tau) - u_{k+1} = \tau^{p+1}\Psi(t_k, u_k) + O\left(\tau^{p+2}\right), \ 0 < k \leq K \ , \tag{9.60}$$

where $p > 0$ is the order of accuracy, Ψ is the principal error function, and $u(t)$ is the exact solution of (9.41) and (9.42).

Now, instead of τ, we choose $\tau/2$ and repeat the computation. It follows that for the new solution $\tilde{u}_{k+1}$ at $t_k + \tau$ we have

$$u(t_k + \tau) - \tilde{u}_{k+1} = c\Big(\frac{\tau}{2}\Big)^{p+1}\Psi(t_j, u_j) + O(\tau^{p+2}) \ , \tag{9.61}$$

where for the positive constant c, $c = O(1)$. Subtracting (9.61) from (9.60), we readily find that

$$\tau^{p+1}\Psi(t_k, u_k) = \frac{\tilde{u}_{k+1} - u_{k+1}}{1 - \frac{c}{2^{p+1}}} + O(\tau^{p+2}) \ .$$

Therefore, it follows

$$\begin{aligned} u(t_k + \tau) - u_{k+1} &= \frac{1}{1 - \frac{c}{2^{p+1}}}\left(\tilde{u}_{k+1} - u_{k+1}\right) + O\left(\tau^{p+2}\right) \\ &\approx \frac{1}{1 - \frac{c}{2^{p+1}}}\left(\tilde{u}_{k+1} - u_{k+1}\right), \ 0 < k \leq K \ . \end{aligned} \tag{9.62}$$

Given $\delta > 0$. An error control per unit step approach is to require the local error κ satisfies

$$\kappa \leq \tau\delta \ ,$$

where

$$\kappa = \left|\frac{2^{p+1}}{2^{p+1} - c}\right| \|\tilde{u}_{k+1} - u_{k+1}\|$$

originates in (9.62). By evaluating κ, we decide if u_{k+1} is an acceptable approximation. If not, the computed solution at time step $k + 1$ is rejected: we go back to u_k and pick a smaller τ. Moreover, if κ is significantly smaller than $\tau\delta$, we take this as an indication that the time step is too small and may be increased. Based on the above criterion, we may further predict a suitable step size to be used in the next step computation. Note that the local error of a next step solution will follow:

$$u\left(t_{k+1} + \tau_{\text{new}}\right) - u_{k+2} = \tau_{\text{new}}^{p+1}\Psi\left(t_{k+1}, u_{k+1}\right) + O\left(\tau_{\text{new}}^{p+2}\right) \ .$$

According to (9.62), we may assume that $u_{k+1} = u_k + O(\tau)$. An appropriate differentiability of the principal error function suggests the estimate

$$\Psi(t_{k+1}, u_{k+1}) = \Psi(t_k, u_k) + O(\tau) \ .$$

With the observation, we predict that

$$\tau_{\text{new}} = \tau \left(\frac{\delta}{\kappa} \right)^{1/(p+1)} .$$

An actual time step to use is determined through a combination of the above error control per unit step criterion and a modified arc-length mechanism. The combination can also help in building a proper stopping criterion for the quenching computation.

It is observed that while quench has not occurred, the numerical solution via (9.57) through (9.59) converges and the computational error is mainly contributed from the truncation error and remains smooth. When the quench is about to occur, however, the computational error changes dramatically, especially when t_k is sufficiently close to the quenching time. Any standard control criterion may break down during this stage. The reasons are as follows. (1) As t_k is getting closer and closer to the quenching time, a sharp change in the derivative of the physical solution u starts. This demands tinier and tinier step sizes to be acquired and used according to a standard error feedback controller. The demand soon becomes impractical due to the increasing of the computational cost and rounding error; thus, the controller fails. (2) The physical solution breaks down at the quenching time and becomes undefined. A numerical solution becomes unsteady, or blows-up, near the quenching point and does not make any sense at that point. This may generate an uncontrollable error.

The actual stopping criterion we considered is as follows:

1. If $\|u_k\|_\infty \geq 1$, then we denote t_{k-1} as the computational quenching time and stop the computation;

2. Let $r_k = e_k/e_{k-1}$ be the error ratio, where $e_k = \|\tilde{u}_k - u_k\|_\infty$ is the error reference at time t_k. If $r_k > \lambda$ where λ is a controlling constant determined through the combination of the aforementioned error analysis and the arc-length criterion, $\lambda \gg 1$, then denote t_k as the computational quenching time and stop the computation.

3. Otherwise, stop when $t_k = T$.

9.4 Computational Examples and Conclusions

It has never been an easy task to approximate numerically critical values of a quenching problem. Our second-order accurate adaptive algorithm (9.26) through (9.29) provides a reliable way for solving the nonlinear partial differential equations (9.3). The compound structure of the adaptive scheme is relatively simple and takes advantage of several known computing techniques. Without loss of generality, the initial value u_0 is set to be zero. $\kappa = 0$ is considered. The spatial mesh step size is chosen as 0.1, while the initial time step varies from 0.01 to 0.001. The purpose

of choosing a smaller initial time step size is not for the stability of the numerical scheme, but for observing more accurately the quenching behavior. When numerical solutions are advanced to be near the quenching point, should quenching exist, they become very sensitive and the rates of change increase unboundedly with respect to time.

We also give the estimated order of convergence of the numerical solution to the exact solution for each of the examples.

Example 9.1

Let $c \equiv 0$. We consider the non-degenerate problem

$$u_t = u_{xx} + \frac{1}{(1-u)^\theta}, \ 0 < x < a, \ 0 < t < T \ , \tag{9.63}$$

$$u(x,0) = 0, 0 < x < a; u(0,t) = u(a,t) = 0, 0 < t < T, \theta > 0 \ . \tag{9.64}$$

We consider cases when $\theta = 1/2$, 1, and 2, respectively. According to investigations by Acker, Walter, and Kawarada [1, 17, 32], the critical length $a^* \approx 1.5303$ for $\theta = 1$. Our computations further indicate that $a^* \approx 1.8856$, 1.1832 for $\theta = 1/2$ and 2, respectively. Let $a = 1.55, 2, \pi, 10$, respectively. We compute the quenching time by means of the adaptive scheme developed. In the case of $\theta = 1, a = \pi$, Chan and Chen [4] show that $0.5 \le T_a \le 0.6772$. As for $\theta = 1$ and $a = 1.55, 2$, they later observe that $T_a = 3.963, 0.779$, respectively. Let $\tau_0 = 0.5 \times 10^{-4}$ and $h = 0.1 \times 2^{-s}$, $s = 0, 1, 2, 3, 4$, respectively. Further, we let T_a^M be the estimated quenching time obtained by several authors [2]–[5] via Crank–Nicolson type schemes and Newton iterations, and denote ∞ as the case where no possibility of finite quenching time is detected. In Table 9.1(a), for each of the $a > a^*$ given, we list the computed quenching time T_a by using our compound adaptive scheme. We only need to consider values of u at $x = a/2$ where maxima of the function u, $0 < x < a$, occur. Our results are almost identical to existing results but slightly less than those at third decimal places [1], [4]–[7]. Numerical solutions also demonstrate good stability of the scheme. □

In Figures 9.1(b)–9.1(d), we plot evolution profiles of the numerical solution u, as well as rates of change u_t, u_{tt} for different testing values of a at $x = a/2$. The monotone increases of the function values when $a \ge a^*$ again demonstrate the conclusions of the lemma and theorem. It is also noticed that values of u, u_t, and u_{tt} increase smoothly at the beginning, but u_t, u_{tt} grow exponentially while t approaches T_a. The phenomenon not only suggests the necessity of the use of finer time step sizes through proper adaptive mechanisms near the quenching point, but also implies that extra care is needed when designing or using a higher-order numerical method for problems possessing quenching singularities (see Table 9.1(d) for maximal values of functions u, u_t, u_{tt}). The rapid increase of higher derivative values may enlarge the error constants and may subsequently reduce the actual accuracy of a numerical method no matter how "higher order" is declared through a standard theoretical analysis.

Table 9.1 *(a)* The Computed Quenching Time T_a for Different θ and a ($\tau_0 = 0.5 \times 10^{-4}$)

θ	$a \setminus r$	0.1	0.4	1.6	6.4	25.6	T_a	T_a^M
0.5	1.55	∞	∞	∞	∞	∞	∞	N.A.
0.5	2	2.107	2.127	2.132	2.133	2.133	2.133	N.A.
0.5	π	0.776	0.774	0.774	0.773	0.773	0.773	N.A.
0.5	10	0.666	0.666	0.666	0.666	0.666	0.666	N.A.
1.0	1.55	3.669	3.893	3.942	3.957	3.961	3.961	3.963
1.0	2	0.778	0.778	0.779	0.779	0.779	0.779	0.779
1.0	π	0.539	0.539	0.538	0.537	0.537	0.537	0.538
1.0	10	0.5	0.5	0.5	0.5	0.5	0.5	0.5
2.0	1.55	0.531	0.532	0.532	0.532	0.532	0.532	N.A.
2.0	2	0.401	0.400	0.400	0.400	0.400	0.400	N.A.
2.0	π	0.343	0.342	0.341	0.341	0.341	0.341	N.A.
2.0	10	0.333	0.333	0.333	0.333	0.333	0.333	N.A.

Table 9.1 *(b)* Computed Maximal Values of u, u_t, u_{tt} Before Quench ($\theta = 1$)

Max. values $\setminus$ a	1.50	1.55	2.00	π	10.0
Max$\{u\}$	0.46	0.99	0.99	0.99	0.99
Max$\{u_t\}$	1.02703	25.2960	35.3781	23.7981	40.4025
Max$\{u_{tt}\}$	1.03107	12092.46	18254.49	15645.45	21013.38

Table 9.1 *(c)* The Monotone Convergence of T_a as $a \to \infty$

a	T_a	a	T_a	a	T_a	$a.T_a$	
1.55	3.961	1.80	0.999	3.00	0.546	5.00	0.503
1.60	2.007	1.90	0.871	π	0.537	10.00	0.500
1.70	1.257	2.00	0.779	4.00	0.511	50.00	0.500

Table 9.1 *(d)* The Approximation of α and β ($\theta = 1$; $h, r = 0.1$; $a = 2$, $\tau_0 = 0.5 \times 10^{-4}$)

t	$u(1,t)$	α	β	t	$u(1,t)$	α	β
0.7780	0.96359	0.53622	1.47869	0.7785	0.97518	0.57526	1.96670
0.7781	0.96559	0.54252	1.54548	0.7786	0.97817	0.75067	7.75870
0.7782	0.96772	0.55606	1.70212	0.7787	0.98241	0.84052	16.0813
0.7783	0.97003	0.55561	1.69658	0.7788	0.98749	2.08430	641212.14
0.7784	0.97249	0.56439	1.81075	0.7789	0.99705	–	–

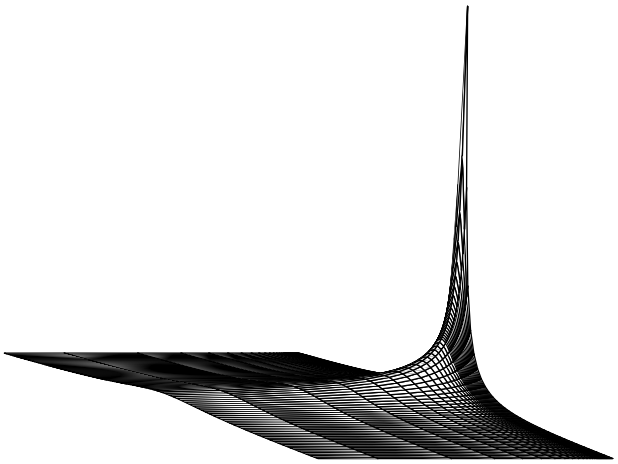


FIGURE 9.1
(a) **The blow-up profile of the function u_t corresponding to the solution of** (9.63) **and** (9.64)**. Semi-adaptive method is used ($\theta = 1$; $h = 0.1$; $\tau_0 = 0.5 \times 10^{-4}$, $a =$** 1.55)**. It is noticed that the increase of u_t becomes exponential when $t \geq 3$ and the solution reaches the quenching point at $t \approx 3.669$. u_t grows relatively slow while $t << 3$.**

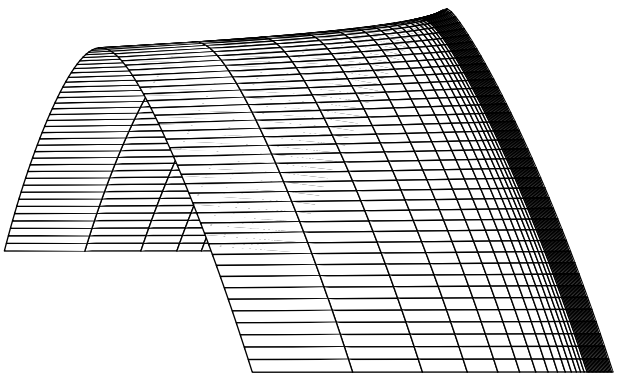


FIGURE 9.1
(b) **The profile of u under the same conditions of Figure 9.1(a). Semi-adaptive method is used. It is noticed that the increase of u increases rapidly while t approaches the quenching time. u grows relatively slow while $t << 3$.**

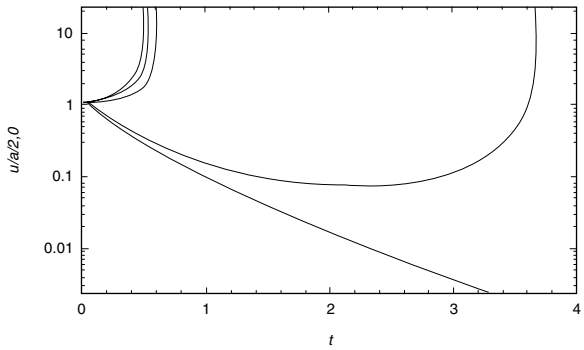


FIGURE 9.1
***(c)* Blow-up patterns of u_t ($\theta = 1; h = 0.1; \tau_0 = 0.5 \times 10^{-4}$). Curves from the left to right are for $a = 10, \pi, 2, 1.55$, and 1.5, respectively. Rate of changes of the function u becomes more noticeable.**

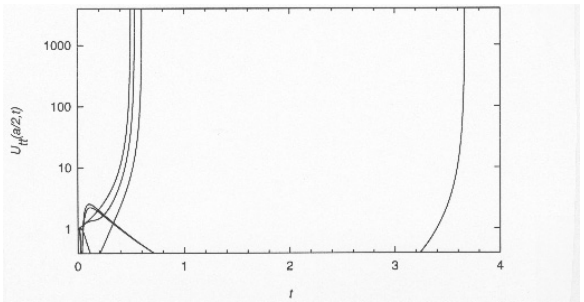


FIGURE 9.1
***(d)* Blow-up patterns of u_{tt} ($\theta = 1; h = 0.1; \tau_0 = 0.5 \times 10^{-4}$). Curves increasing exponentially from the left to right are for $a = 10, \pi, 2$, and 1.55, respectively. The profile of u_{tt} corresponding to $a = 1.55$ is divided into two stages since $u_{tt} < 0$ for $0.76 < t < 3.34$.**

In Figure 9.1(c), we observe that for $a = 1.5$, the rate of change of u decreases monotonically and becomes negative at $t \approx 3.35$. This again coincides with the theory that no quench may occur in the case. We also find that u quenches in finite time for $a \geq 1.55$. In Figure 9.1(d), it is found that the derivative values grow exponentially and are greater than 10^3 before t reaches 4. This exponential growth may seriously affect the actual accuracy of a higher order method used for solving the quenching problem. On the other hand, from Tables 9.1(a) and (c) and Figure 9.1(b), we may observe that, when $\theta = 1$, the quenching time T_a decays monotonically to 1/2 as a increases. Similar properties can be found in cases where different θ values are considered.

Let $a = 2$. According to Filippas and Guo [11], the quenching solution satisfies the following property:

$$\lim_{t \to T_a} u(1,t) = 1 - \sqrt{2}(T_a - t)^{1/2}, \ \theta = 1 \ . \tag{9.65}$$

We wish to see if our numerical solution approximately satisfies (9.65). For this, similar to Sanz–Serna et al. [30], we introduce an approximation of the formula (9.65) with two parameters α, β:

$$u(1,t) = 1 - \beta(T_a - t)^{\alpha}, \ \theta = 1 \ , \tag{9.66}$$

where t is sufficiently close to T_a. By using the results in Table 9.1(a) for (9.66), we immediately obtain Table 9.1(d) with estimated values of parameters α and β. Notice the rapid growth of the numerical error as t approaches T_a due to the quenching singularity and the sensitivity of the formula (9.66) when used in calculations. Taking the averages of their first six values from the table, we immediately obtain $\alpha = 0.55507$, $\beta = 1.70005$ which are good approximations to theoretical predictions.

Let u be the exact solution or its best known approximation, and $u_{h,\tau}$ the numerical solution of the problem (9.63) and (9.64). Supposing that the error in the time direction is negligible, we may denote $u_h = u_{h,\tau}$ and have $|u_h - u| \approx Ch^{\rho}$. Let the profile of τ be the same while h is reduced. The order of accuracy, ρ, can then be estimated by means of the formula

$$\rho \approx \frac{1}{\ln 2} \ln \frac{|u_h - u|}{|u_{h/2} - u|} \ . \tag{9.67}$$

To apply the formula, we set $h = 0.1$ and let u be the numerical solution at $\tilde{h} = 1/160$ in the uniformed spatial interval. We only consider the case of $\theta = 1, a = \pi$ as an example here and leave the more general discussion to the nonlinear degenerate problem in next example. Let $\tau_0 = 0.5 \times 10^{-5}$. We take the initial time step τ as 0.0001 so that the influence of error may be neglected. We compute the value of ρ at $(a/2, t)$ for $0.438 \le t = m\tau < T_a \approx 0.538$ and then consider the arithmetic average as the estimate of the order of accuracy. We obtain

$$\rho \approx 2.27557286$$

which fairly indicates that the actual order of accuracy is around 2. A similar estimate can be computed in the time direction by choosing $h^2 << \tau$ and adopting an analog of (9.67).

Finally, Figure 9.1(e) shows the profile of the adaptive temporal discretization parameter τ, while $\theta = 1, a = \pi$, initial $\tau = 0.001$ and $\tau_0 = 0.0.5 \times 10^{-4}$ are considered. Profiles of τ in other cases are similar.

Example 9.2

Let $N = 79$ for the normalized interval $[0, 1]$. We consider the initial spatial step size $h = 1/80$ in the space, while the initial temporal step being $\tau_0 = 0.001$ in our experiments. □

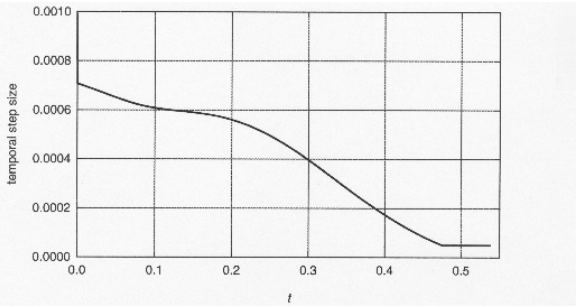


FIGURE 9.1

***(e)* The profile of τ. It is observed that the temporal step size decreases according to the monotone increase of u_t until the given minimal stepsize controller.**

Consider the following degenerate semilinear initial-boundary value problem:

$$x^q u_t = u_{xx} + cu_x + \frac{1}{(1-u)^\theta}, \ 0 < x < r, 0 < t < T, \ q \geq 0, \theta > 0 \ , \quad (9.68)$$

$$u(x, 0) = 0, 0 < x < r; \ u(0, t) = u(r, t) = 0, 0 \leq t < T \ . \quad (9.69)$$

We need only consider the case with $\theta = 1$ and other cases are similar. The function c is taken to be $b/(1+x)$ and b/x, respectively, where b is a constant. We note that in the latter case, the function c becomes unbounded while $x \to 0^+$ and this causes slight perturbation at the left end of the interval [0, 1] in the numerical solution. However, the amplitude of this oscillation decreases and is under control as the computation continuous.

For the standard quenching problem (9.68) and (9.69), we wish to predict the critical length $r_{q,c}$ and quenching time $T_{q,c}$ while computing the numerical solution, should they exist and be finite.

Figures 9.2(a) and (b) show profiles of the derivative function u_t in during the final stages before blowing up. The parameter $q = 1$ and interval length $r = \pi$ are used. Functions at 10 different time levels from 0.7434 to 0.7443 are displayed. The maximal value of the derivative function u_t increases monotonically but rapidly from 18.0961 when $t = 0.7434$ to 82.5939 as t reaches 0.7443. The location of the maximal value in the space is approximately 1.2570 which is slightly shifted to the left from the center of the interval.

In Table 9.2(a), we list locations of $\max_x u_t(x, t)$ immediately before quenching: The predicted quenching time in the case is $T_{q,c} = 0.7443^+$. In Figure 9.2(c), we show the profile of the solution u during the same stage before quenching. Values of $\max_x u(x, t)$ tend to the unity monotonically and steadily. We have $\max_{x,t\leq 0.7443} = 0.991742$ in Figure 9.2(c). The same set of parameters as in previous graphs is used. To see more precisely the shape of solution u, we darken the area under u, $0 < x < \pi$. We note that since that the coefficient function used, $c(x) = -2/(1 + x)$, is bounded throughout the computation, both the solution u and its derivative are smooth until quench is reached.

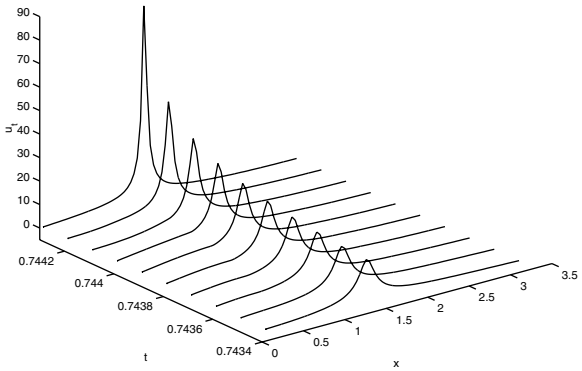


FIGURE 9.2

***(a)* The profile of the derivative function u_t. Because of the fully adaptive mesh structure, here we plot only the numerical solution at 10 selected time levels immediately before the quenching time ($q = 1, c = -2/(1+x), a = \pi$).**

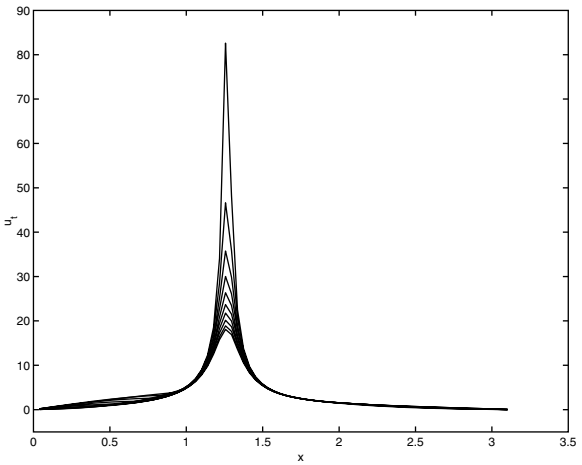


FIGURE 9.2

***(b)* The blow-up of the derivative function u_t ($q = 1; c = -2/(1+x), a = \pi$). Numerical solutions at the same 10 time levels as before are considered. We observe that the amplitude of u_t changes rapidly as t approaches the quenching time.**

Figure 9.2(d) is devoted to the distribution of the spatial step sizes immediately before quench occurs. In the first picture, we plot the diagram as the lengths of 80 spatial grids used. The height of each vertical bar is given by the step size h_i, $1 \leq i \leq 80$. We find that the step size reduces rapidly when it gets closer to the quenching location. The step sizes resume slightly at the predicted quenching location due to the influence of the cubic spline approximation used in between different time levels.

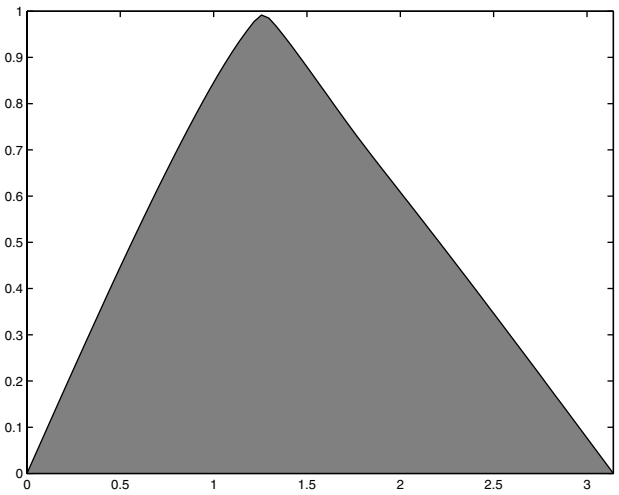


FIGURE 9.2
***(c)* The solution u at the final time stage** ($q = 1; c = -2/(1+x), a = \pi, t = 0.74412$)**. We purposely darken the area topped by the solution in order to view better the curve.**

Table 9.2 *(a)* Maximal Values of $u_t(x,t)$ and Their Locations

t	$x_{\max}$	$\max_x u_t$	t	$x_{\max}$	$\max_x u_t$	t	$x_{\max}$	$\max_x u_t$
0.7434	1.257	18.0961	0.7438	1.257	23.6767	0.7442	1.257	46.6237
0.7436	1.257	20.1446	0.7440	1.257	30.0193	0.7443	1.257	82.5939

Sharp increases of the function can be observed during the final stage of computations ($q = 1, c(x) = b/(1+x), b = -2, r = \pi$).

The second graph in Figure 9.2(d) displays the same distribution by plotting grid references (x_i, h_{i+1}), $i = 0, 1, \ldots, 80$, in the same map. We may again observe that mesh points are well distributed according to the profile of u_t, which is exhibited in Figures 9.2(a) and (b).

Table 9.2 *(b)* Maximal Values of $u_t(x,t)$ and Their Locations

t	$x_{\max}$	$\max_x u_t$	t	$x_{\max}$	$\max_x u_t$	t	$x_{\max}$	$\max_x u_t$
0.5720	1.335	21.2765	0.5724	1.335	28.5837	0.5728	1.335	64.3619
0.5722	1.335	24.0991	0.5726	1.335	37.0148	0.5729	1.335	280.586

Sharp increases of the function can be observed during the final stage of computations ($q = 0.2, c(x) = b/x, b = 0.4, r = \pi$).

Table 9.2(b) is for numerical experiments when a different coefficient function, $c(x) = 0.4/x$, together with $q = 0.2, r = \pi$ is employed. Again, the derivative

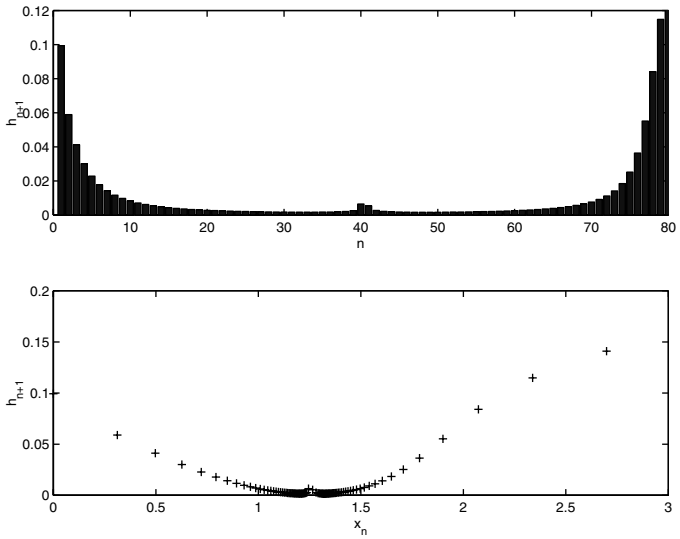


FIGURE 9.2
***(d)* The grid distribution in space at $t = 0.74421$, immediately before quenching. The first diagram is for the distribution against n while the second one is against the location x.**

Table 9.2 *(c)* The Evolution of the Error Ratio Function e_k/e_{k-1} in a Quenching Case ($q = 1, c(x) = b/(1+x), b = -2, r = 5$)

t_k	e_k/e_{k-1}	t_k	e_k/e_{k-1}	t_k	e_k/e_{k-1}	t_k	e_k/e_{k-1}
0.209500	0.999104	0.500509	0.996062	0.710001	1.00061	0.739001	1.00334
0.300500	0.999104	0.600017	0.996971	0.720001	1.00071	0.739101	1.00335
0.400038	0.999937	0.700001	1.000460	0.730001	1.00104	0.739201	243.771

It is observed, as predicted, the ratio blows up as $t \to T_{q,c}$.

function u_t in the final stage is displayed. The function is plotted in 10 different time levels with the stage as t varies from 0.5578 to 0.5587. The maximal value, $\max_x u_t(x,t)$, tends to infinite as t approaches $T_{q,c}$ which is approximately 0.5729^+ in the case. The computed maximal value of the derivative function u_t immediately before quenching is about 280.586.

In Table 9.2(c), we present values of the error ratio $e_k/e_{k-1}, k = 1, \ldots, K$, for a quenching case with $c(x) = -2/(1+x)$. Parameters $q = 1, r = 5$ are used. Predicted quenching time $T_{q,c} \approx 0.739201^+$. We observe that the ratio remains to be well bounded before the quenching time and increases abruptly when quenching time is reached. This provides an obvious stopping criterion as we discussed earlier in the last section. Figure 9.2(e) further illustrates such stopping criterion. In the graph, we plot out the error ratio e_k/e_{k-1} while the computational advice goes by. The particular

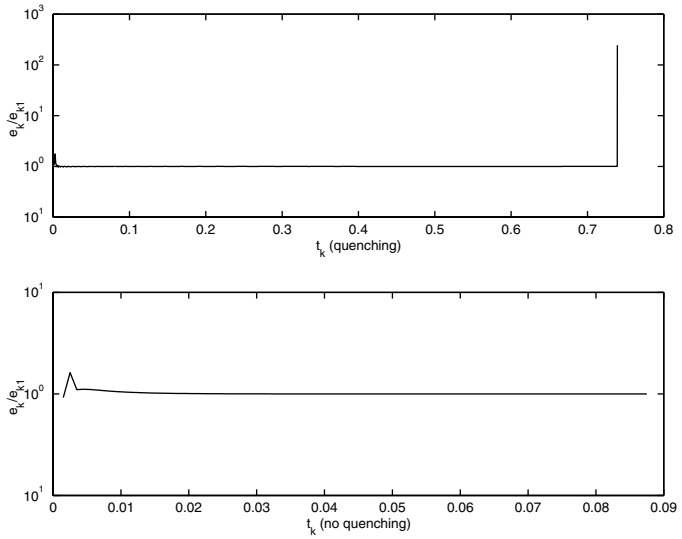


FIGURE 9.2

***(e)* Error ratio profiles. Though initially the error ratio function e_k/e_{k-1} looks flat, it increases rapidly while t approaches the quenching time, if quenching occurs. Settings $q = 1, b = -2/(1+x), a = 5$ (top), and $a = 0.5$ (bottom) are used.**

cases, quenching and non-quenching, are considered. For the quenching case (the first graph), this ratio remains nearly as constant 1 and does not change much until quenching occurs. At that point, it suddenly jumps from 1.00335 to 243.771 and this indicates the break down of the numerical computation. The same ratio remains steady in the second graph for the non-quenching case.

From the above experiments, we may conclude that:

1. The linearly implicit semi- and fully adaptive schemes are highly efficient and accurate for solving the nonlinear reaction-diffusion problems with singular source terms. The systems of equations derived with nonsingular tridiagonal coefficient matrices are relatively simple to solve and numerically stable. The adaptive designs work smoothly throughout the computation.

2. The adaptive methods developed are also reliable. In the fully adaptive method, we not only employ adaptive structures both in space and time, but also build a stopping criterion based on the error analysis. The numerical solution well follows the pattern of the physical solution. With the help of the adaptation in space, the break up of u_t in spatial directions can be more precisely monitored. This is in fact difficult to achieve for time-only adaptive algorithms.

3. The adaptive structures discussed can be conveniently extended for solving multidimensional singular reaction-diffusion problems defined in, say, rectan-

gular spatial domains. We may predict that method of dimensional splitting can be employed to further simplify the computational procedure. In that case, the critical length will be replaced by the critical index which combines both the physical size information and geometric pattern factor of the spatial domain involved.

Acknowledgments

The work of the first author is supported in part by the Louisiana State under the Grant No. LEQSF-(1997-00)-RD-B-15.

References

[1] A. Acker and W. Walter, The quenching problem for nonlinear parabolic differential equations, *Lecture Notes in Math.,* **564** (1976), 1–12, Springer-Verlag, New York.

[2] M.C. Branch, M.W. Beckstead, T.A. Litzinger, M.D. Smooke, and V.H. Yang, Nonsteady combustion mechanisms of advanced solid propellants, *Annual Technical Report,* **94-05**, Center for Combustion and Environmental Research, University of Colorado, Boulder, CO., 1994.

[3] C.J. Budd, G.P. Koomullil, and A.M. Stuart, On the solution of convection-diffusion boundary value problems using equidistributed grids, *SIAM J. Sci. Comput.,* **20**, (1998), 591–618.

[4] C.Y. Chan and C.S. Chen, A numerical method for semilinear singular parabolic mixed boundary-value problems, *Quart. Appl. Math.,* **47**, (1989), 45–57.

[5] C.Y. Chan, L. Ke, and A.S. Vatsala, Impulsive quenching for reaction-diffusion equations, *Nonlinear Anal.,* **22**, (1994), 1323–1328.

[6] C.S. Chen, The method of fundamental solutions for nonlinear thermal explosions, *Comm. Numer. Methods Engrg.,* **11**, (1995), 675–681.

[7] K. Deng and H.A. Levine, On the blow-up of u_t at quenching, *Proc. Amer. Math. Soc.,* **106**, (1989), 1049–1056.

[8] E.A. Dorfi and L.O.C. Drury. Simple adaptive grids for 1-D initial value problems, *J. Comput. Phys.,* **40**, (1981), 202.

[9] E.S. Fraga and J.Ll. Morris, An adaptive mesh refinement method for nonlinear dispersive wave equations, *J. Comp. Physics,* **101**, (1992), 94–103.

[10] R.M. Furzeland, J.G. Verwer, and P.A. Zegeling, A numerical study of three moving-grid methods for one-dimensional partial differential equations which are based on the method of lines, *J. Comp. Physics,* **89**, (1990), 349–388.

[11] S. Filippas and J.S. Guo, Quenching profiles for one-dimensional semilinear heat equations, *Quart. Appl. Math.,* **51**, (1993), 713–729.

[12] A. Ghafourian, C. Huyn, P. Johnson, S. Hevert, H. Dindi, S. Mahalingam, and J.W. Faily, Liquid rocket combustion instability, *Research Report,* **90-02**, (1990), Center for Combustion and Environmental Research, University of Colorado, Boulder, CO.

[13] J.S. Guo and B. Hu, The profile near quenching time for the solution of a singular semilinear heat equation, *Proc. Edinburgh Math. Soc., (2)* **40**, (1997), 437–456.

[14] P. Henrici, *Discrete Variable Methods in Ordinary Differential Equations,* John Wiley & Sons, Inc, New York, 1962.

[15] W. Huang, Y. Ren, and R.D. Russell, Moving mesh partial differential equations (MM-PDEs) based on the equidistribution principle, *SIAM J. Numer. Anal.,* **31**, (1994), 709–730.

[16] A. Iserles, *A First Course in the Numerical Analysis of Differential Equations,* Cambridge University Press, 1996.

[17] H. Kawarada, On solutions of initial-boundary value problem for $u_t = u_{xx} + 1/(1-u)$, *Publ. Res. Inst. Math. Sci.,* **10**, (1975), 729–736.

[18] A.Q.M. Khaliq, E.H. Twizell, and D.A. Voss, On parallel algorithms for semidiscretized parabolic partial differential equations based on subdiagonal padé approximations, *Num. Math. for Partial Diff. Eqns.,* **9**, (1993), 107–116.

[19] J. Lang, Two-dimensional fully adaptive solutions of reaction-diffusion equations, *Appl. Numer. Math.,* **18**, (1995), 223–240.

[20] J. Lang and A. Walter, An adaptive Rothe method for nonlinear reaction-diffusion systems, *Appl. Numer. Math.,* **13**, (1993), 135–146.

[21] H.A. Levine, Quenching, nonquenching, and beyond quenching for solutions of some parabolic equations, *Ann. Mat. Pure. Appl.,* **4**, (1989), 243–260.

[22] V. Pareyra and E.G. Sewell, Mesh selection for discrete solution of boundary value problems in ordinary differential equations, *Numer. Math.,* **23**, (1975) 261–268.

[23] Y. Ren and R.D. Russell, Moving mesh techniques based upon equidistribution, and their stability, *SIAM J. Sci. Stat. Comput.,* **13**, (1992) 1265–1286.

[24] R.D. Russell and J. Christiansen, Adaptive mesh selection strategies for solving boundary value problems, *SIAM J. Numer. Anal.,* **15**, (1978) 59–80.

[25] L.F. Shampine, *Numerical Solution of Ordinary Differential Equations,* Chapman & Hall, 1994.

[26] Q. Sheng, A monotonically convergent adaptive method for nonlinear combustion problems, *Integral Methods in Science & Engineering,* Research Notes in Mathematics 418, Chapman & Hall/CRC (2000), 310–315.

[27] Q. Sheng and H. Cheng, A moving mesh approach to the numerical solution of nonlinear degenerate quenching problems, *Dynamic Sys. Appl.,* **7**, (1999), 343–358.

[28] Q. Sheng and H. Cheng, An adaptive grid method for degenerate semilinear quenching problems, *Computers Math. Applications,* **39**, (2000), 57–71.

[29] Q. Sheng and A.Q.M. Khaliq, A compound adaptive approach to degenerate nonlinear quenching problems, *Numer. Meth. for PDEs,* **15**, (1999), 29–47.

[30] Y. Tourigny and J.M. Sanz-Serna, The numerical study of blow-up with application to a nonlinear Schrödinger equation, *J. Comp. Phys.,* **102**, (1992), 407–416.

[31] D.A. Voss and A.Q.M. Khaliq, Parallel LOD methods for second order time dependent PDEs, *Computers Math. Applic.,* **30**, (1995), 25–35.

[32] W. Walter, Parabolic differential equations with a singular nonlinear term, *Funkcial Ekvac.,* **19**, 271–277.

Chapter 10

Adaptive Linearly Implicit Methods for Heat and Mass Transfer Problems

J. Lang and B. Erdmann

10.1 Introduction

Dynamical process simulation is the central tool nowadays to assess the modeling process for large-scale physical problems arising in such fields as biology, chemistry, metallurgy, medicine, and environmental science. Moreover, successful numerical methods are very attractive to design and control economical plants at low costs in a short time. Due to the great complexity of the established models, the development of fast and reliable algorithms has been a topic of continuing investigation during recent years.

One of the important requirements that modern software must meet today is to judge the quality of its numerical approximations in order to assess safely the modeling process. Adaptive methods have proven to work efficiently providing *a posteriori* error estimates and appropriate strategies to improve the accuracy where needed. They are now entering into real-life applications and starting to become a standard feature in simulation programs. This chapter reports on one successful way to construct discretization methods adaptive in space and time, which are applicable to a wide range of practically relevant problems.

We concentrate on heat and mass transfer problems which can be written in the form

$$B(x,t,u,\nabla u)\partial_t u = \nabla \cdot (D(x,t,u,\nabla u)\nabla u) + F(x,t,u,\nabla u)\ , \tag{10.1}$$

supplemented with suitable boundary and initial conditions. The vector-valued solution $u = (u_1, \ldots, u_m)^T$ is supposed to be unique. This problem class includes the well-known reaction-diffusion equations and the Navier–Stokes equations as well.

In the classical method of lines (MOL) approach, the spatial discretization is done once and kept fixed during the time integration. Discrete solution values correspond to points on lines parallel to the time axis. Since adaptivity in space means to add or delete

points, in an adaptive MOL approach, new lines can arise and later disappear. Here, we allow a local spatial refinement in each time step, which results in a discretization sequence first in time then in space. The spatial discretization is considered as a perturbation, which has to be controlled within each time step. Combined with *a posteriori* error estimates, this approach is known as adaptive Rothe method. First theoretical investigations have been made by Bornemann [7] for linear parabolic equations. Lang and Walter [26] generalized the adaptive Rothe approach to reaction-diffusion systems. A rigorous analysis for nonlinear parabolic systems is given in Lang [28]. For a comparative study, we refer to Deuflhard et al. [16].

Since differential operators give rise to infinite stiffness, often an implicit method is applied to discretize in time. We use linearly implicit methods of Rosenbrock type, which are constructed by incorporating the Jacobian directly into the formula. These methods offer several advantages. They completely avoid the solution of nonlinear equations, which means no Newton iteration has to be controlled. There is no problem to construct Rosenbrock methods with optimum linear stability properties for stiff equations. According to their one-step nature, they allow a rapid change of step sizes and an efficient adaptation of the spatial discretization in each time step. Moreover, a simple embedding technique can be used to estimate the error in time satisfactorily. A description of the main idea of linearly implicit methods is given in Section 10.2.

Stabilized finite elements are used for the spatial discretization to prevent numerical instabilities caused by advection-dominated terms. To estimate the error in space, the hierarchical basis technique has been extended to Rosenbrock schemes in Lang [28]. Hierarchical error estimators have been accepted to provide efficient and reliable assessment of spatial errors. They can be used to steer a multilevel process, which aims at getting a successively improved spatial discretization, drastically reducing the size of the arising linear algebraic systems with respect to a prescribed tolerance (Bornemann et al. [8], Deuflhard et al. [17], Bank and Smith [2]). A brief introduction to multilevel finite element methods is given in Section 10.3.

The described algorithm has been coded in the fully adaptive software package KARDOS at the Konrad–Zuse–Zentrum in Berlin. Several types of embedded Rosenbrock solvers and adaptive finite elements were implemented. KARDOS is based on the KASKADE-toolbox [18], which is freely distributed at ftp://ftp.zib.de/pub/kaskade. Nowadays both codes are efficient and reliable workhorses to solve a wide class of PDEs in one, two, or three space dimensions. To demonstrate the performance of our adaptive approach, in Section 10.4 we will present two practically relevant problems occurring in combustion theory and brine transport in porous media.

10.2 Linearly Implicit Methods

In this section a short description of the linearly implicit discretization idea is given. More details can be found in the books of Hairer and Wanner [23], Deuflhard and

Bornemann [15], and Strehmel and Weiner [37]. For ease of presentation, we first set $B = I$ in (10.1) and consider the autonomous case. Then we can look at (10.1) as an abstract Cauchy problem of the form

$$\partial_t u = f(u) \,, \quad u(t_0) = u_0 \,, \quad t > t_0 \,, \tag{10.2}$$

where the differential operators and the boundary conditions are incorporated into the nonlinear function $f(u)$. Since differential operators give rise to infinite stiffness, often an implicit discretization method is applied to integrate in time. The simplest scheme is the implicit (backward) Euler method

$$u_{n+1} = u_n + \tau \, f(u_{n+1}) \,, \tag{10.3}$$

where $\tau = t_{n+1} - t_n$ is the step size and u_n denotes an approximation of $u(t)$ at $t = t_n$. This equation is implicit in u_{n+1} and thus usually a Newton-like iteration method has to be used to approximate the numerical solution itself. The implementation of an efficient nonlinear solver is the main problem for a fully implicit method.

Investigating the convergence of Newton's method in function space, Deuflhard [13] pointed out that one calculation of the Jacobian or an approximation of it per time step is sufficient to integrate stiff problems efficiently. Using u_n as an initial iterate in a Newton method applied to (10.3), we find

$$(I - \tau \, J_n) \, K_n = \tau f(u_n) \,, \tag{10.4}$$

$$u_{n+1} = u_n + K_n \,, \tag{10.5}$$

where J_n stands for the Jacobian matrix $\partial_u f(u_n)$. The arising scheme is known as the *linearly implicit* Euler method. The numerical solution is now effectively computed by solving the system of linear equations that defines the increment K_n. Among the methods that are capable of integrating stiff equations efficiently, linearly implicit methods are the easiest to program, since they completely avoid the numerical solution of nonlinear systems.

One important class of higher-order linearly implicit methods consists of extrapolation methods that are very effective in reducing the error, see Deuflhard [14]. However, in the case of higher spatial dimension, several drawbacks of extrapolation methods have shown up in numerical experiments made by Bornemann [6]. Another generalization of the linearly implicit approach we will follow here leads to Rosenbrock methods [35]. They have found wide-spread use in the ordinary differential equations (ODE) context. Applied to (10.2) a so-called s-stage Rosenbrock method has the recursive form

$$\begin{aligned} (I - \tau \gamma_{ii} \, J_n) \, K_{ni} &= \tau f \left(u_n + \sum_{j=1}^{i-1} \alpha_{ij} \, K_{nj} \right) \\ &\quad + \tau \, J_n \sum_{j=1}^{i-1} \gamma_{ij} \, K_{nj} \,, \; i = 1(1)s \,, \end{aligned} \tag{10.6}$$

$$u_{n+1} = u_n + \sum_{i=1}^{s} b_i \, K_{ni} \,, \tag{10.7}$$

where the step number s and the defining formula coefficients b_i, α_{ij}, and γ_{ij} are chosen to obtain a desired order of consistency and good stability properties for stiff equations (see, e.g., [23, IV.7]). We assume $\gamma_{ii} = \gamma > 0$ for all i, which is the standard simplification to derive Rosenbrock methods with one and the same operator on the left-hand side of (10.6). The linearly implicit Euler method mentioned above is recovered for $s = 1$ and $\gamma = 1$.

For the general system

$$B(t,u)\partial_t u = f(t,u)\,, \quad u(t_0) = u_0\,, \quad t > t_0\,, \tag{10.8}$$

an efficient implementation that avoids matrix-vector multiplications with the Jacobian was given by Lubich and Roche [31]. In the case of a time- or solution-dependent matrix B, an approximation of $\partial_t u$ has to be taken into account, leading to the generalized Rosenbrock method of the form

$$\begin{aligned}\left(\frac{1}{\tau\gamma}B(t_n,u_n) - J_n\right)U_{ni} &= f(t_i,U_i) - B(t_n,u_n)\sum_{j=1}^{i-1}\frac{c_{ij}}{\tau}U_{nj} + \tau\gamma_i C_n \\ &\quad + (B(t_n,u_n) - B(t_i,U_i))\,Z_i\,, \quad i = 1(1)s\,,\end{aligned} \tag{10.9}$$

where the internal values are given by

$$t_i = t_n + \alpha_i\tau\,, \quad U_i = u_n + \sum_{j=1}^{i-1} a_{ij}\,U_{nj}\,, \quad Z_i = (1-\sigma_i)z_n + \sum_{j=1}^{i-1}\frac{s_{ij}}{\tau}\,U_{nj}\,,$$

and the Jacobians are defined by

$$\begin{aligned} J_n &:= \partial_u(f(t,u) - B(t,u)z)_{|u=u_n,t=t_n,z=z_n}\,, \\ C_n &:= \partial_t(f(t,u) - B(t,u)z)_{|u=u_n,t=t_n,z=z_n}\,. \end{aligned}$$

This yields the new solution

$$u_{n+1} = u_n + \sum_{i=1}^{s} m_i\,U_{ni}$$

and an approximation of the temporal derivative $\partial_t u$

$$z_{n+1} = z_n + \sum_{i=1}^{s} m_i\left(\frac{1}{\tau}\sum_{j=1}^{i}(c_{ij} - s_{ij})U_{nj} + (\sigma_i - 1)z_n\right)\,.$$

The new coefficients can be derived from α_{ij}, γ_{ij}, and b_i [31]. In the special case $B(t,u) = I$, we get (10.6) setting $U_{ni} = \tau\sum_{j=1,\ldots,i}\gamma_{ij}K_{nj}$, $i = 1,\ldots,s$.

Various Rosenbrock solvers have been constructed to integrate systems of the form (10.8). An important fact is that the formulation (10.8) includes problems of higher differential index. Thus, the coefficients of the Rosenbrock methods have

to be specially designed to obtain a certain order of convergence. Otherwise, order reduction might happen. In [32, 31], the solver ROWDAIND2 was presented, which is suitable for semi-explicit index 2 problems. Among the Rosenbrock methods suitable for index 1 problems we mention those given in [12, 29, 33, 36]. More information can be found in [28]. For the convenience of the reader, we give the defining formula coefficients for ROS2 [12] and ROWDAIND2 in Tables 10.1 and 10.2, respectively. Both Rosenbrock solvers have been used in our simulations presented here.

Table 10.1 Set of Coefficients for ROS2 [12]

γ	$= 1.707106781186547e+00$		
a_{11}	$= 0.000000000000000e+00$	α_1	$= 0.000000000000000e+00$
a_{21}	$= 5.857864376269050e-01$	α_2	$= 1.000000000000000e+00$
a_{22}	$= 0.000000000000000e+00$		
c_{11}	$= 5.857864376269050e-01$	s_{11}	$= 0.000000000000000e+00$
c_{21}	$= 1.171572875253810e+00$	s_{21}	$= 3.431457505076198e-01$
c_{22}	$= 5.857864376269050e-01$	s_{22}	$= 0.000000000000000e+00$
γ_1	$= 1.707106781186547e+00$	σ_1	$= 0.000000000000000e+00$
γ_2	$= -1.707106781186547e+00$	σ_2	$= 5.857864376269050e-01$
m_1	$= 8.786796564403575e-01$	$\hat{m}_1$	$= 5.857864376269050e-01$
m_2	$= 2.928932188134525e-01$	$\hat{m}_2$	$= 0.000000000000000e+00$

Usually, one wishes to adapt the step size in order to control the temporal error. For linearly implicit methods of Rosenbrock type, a second solution of inferior order, say $\hat{p}$, can be computed by a so-called embedded formula

$$\begin{aligned} \hat{u}_{n+1} &= u_n + \sum_{i=1}^{s} \hat{m}_i U_{ni} \ , \\ \hat{z}_{n+1} &= z_n + \sum_{i=1}^{s} \hat{m}_i \left(\frac{1}{\tau} \sum_{j=1}^{i} (c_{ij} - s_{ij}) U_{nj} + (\sigma_i - 1) z_n \right) \ , \end{aligned}$$

where the original weights m_i are simply replaced by $\hat{m}_i$. If p is the order of u_{n+1}, we call such a pair of formulas to be of order $p(\hat{p})$. Introducing an appropriate scaled norm $\| \cdot \|$, the local error estimator

$$r_{n+1} = \| u_{n+1} - \hat{u}_{n+1} \| + \| \tau (z_{n+1} - \hat{z}_{n+1}) \| \tag{10.10}$$

can be used to propose a new time step by

$$\tau_{n+1} = \frac{\tau_n}{\tau_{n-1}} \left(\frac{TOL_t \, r_n}{r_{n+1} \, r_{n+1}} \right)^{1/(\hat{p}+1)} \tau_n \ . \tag{10.11}$$

Table 10.2 Set of Coefficients for ROWDAIND2 [32, 31]

γ =	$3.000000000000000e-01$		
a_{11} =	$0.000000000000000e+00$	α_1 =	$0.000000000000000e+00$
a_{21} =	$1.666666666666667e+00$	α_2 =	$5.000000000000000e-01$
a_{22} =	$0.000000000000000e+00$	α_3 =	$1.000000000000000e+00$
a_{31} =	$1.830769230769234e+00$	α_4 =	$1.000000000000000e+00$
a_{32} =	$2.400000000000000e+00$		
a_{33} =	$0.000000000000000e+00$		
a_{41} =	$1.830769230769234e+00$		
a_{42} =	$2.400000000000000e+00$		
a_{43} =	$0.000000000000000e+00$		
a_{44} =	$0.000000000000000e+00$		
c_{11} =	$3.333333333333333e+00$	s_{11} =	$0.000000000000000e+00$
c_{21} =	$1.246438746438751e+00$	s_{21} =	$5.555555555555556e+00$
c_{22} =	$3.333333333333333e+00$	s_{22} =	$0.000000000000000e+00$
c_{31} =	$-1.226780626780621e+01$	s_{31} =	$-4.239316239316217e+00$
c_{32} =	$4.266666666666667e+01$	s_{32} =	$8.000000000000000e+00$
c_{33} =	$3.333333333333333e+00$	s_{33} =	$0.000000000000000e+00$
c_{41} =	$5.824628046850726e-02$	s_{41} =	$-4.239316239316217e+00$
c_{42} =	$3.259259259259259e+00$	s_{42} =	$8.000000000000000e+00$
c_{43} =	$-3.703703703703704e-01$	s_{43} =	$0.000000000000000e+00$
c_{44} =	$3.333333333333333e+00$	s_{44} =	$0.000000000000000e+00$
γ_1 =	$3.000000000000000e-01$	σ_1 =	$0.000000000000000e+00$
γ_2 =	$1.878205128205124e-01$	σ_2 =	$1.666666666666667e+00$
γ_3 =	$-1.000000000000000e+00$	σ_3 =	$2.307692307692341e-01$
γ_4 =	$0.000000000000000e+00$	σ_4 =	$2.307692307692341e-01$
m_1 =	$1.830769230769234e+00$	$\hat{m}_1$ =	$2.214433650496747e+00$
m_2 =	$2.400000000000000e+00$	$\hat{m}_2$ =	$1.831186394371970e+00$
m_3 =	$0.000000000000000e+00$	$\hat{m}_3$ =	$8.264462809917363e-03$
m_4 =	$1.000000000000000e+00$	$\hat{m}_4$ =	$0.000000000000000e+00$

Here, TOL_t is a desired tolerance prescribed by the user. This formula is related to a discrete PI-controller first established in the pioneering works of Gustaffson et al. [21, 20]. A more standard step-size selection strategy can be found in Hairer et al. [22, II.4].

Rosenbrock methods offer several structural advantages. They preserve conservation properties like fully implicit methods. There is no problem to construct Rosenbrock methods with optimum linear stability properties for stiff equations. Because of their one-step nature, they allow a rapid change of step sizes and an efficient adaptation of the underlying spatial discretizations as will be seen in the next section. Thus, they are attractive for solving real world problems.

10.3 Multilevel Finite Elements

In the context of partial differential equations (PDEs), system (10.9) consists of linear elliptic boundary value problems possibly advection dominated. In the spirit of spatial adaptivity, a multilevel finite element method is used to solve this system. The main idea of the multilevel technique consists of replacing the solution space by a sequence of discrete spaces with successively increasing dimension to improve their approximation property. *A posteriori* error estimates provide the appropriate framework to determine where a mesh refinement is necessary and where degrees of freedom are no longer needed. Adaptive multilevel methods have proven to be a useful tool for drastically reducing the size of the arising linear algebraic systems and to achieve high and controlled accuracy of the spatial discretization (see, e.g., [1, 17, 27]).

Let T_h be an admissible finite element mesh at $t = t_n$ and S_h^q be the associated finite dimensional space consisting of all continuous functions which are polynomials of order q on each finite element $T \in T_h$. Then the standard Galerkin finite element approximation $U_{ni}^h \in S_h^q$ of the intermediate values U_{ni} satisfies the equation

$$\left(L_n\, U_{ni}^h, \phi\right) = (r_{ni}, \phi) \quad \text{for all } \phi \in S_h^q \ , \tag{10.12}$$

where L_n is the weak representation of the differential operator on the left-hand side in (10.9) and r_{ni} stands for the entire right-hand side in (10.9). Since the operator L_n is independent of i its calculation is required only once within each time step.

It is a well-known inconvenience that the solutions U_{ni}^h may suffer from numerical oscillations caused by dominating convective and reactive terms as well. An attractive way to overcome this drawback is to add locally weighted residuals to get a stabilized discretization of the form

$$\left(L_n\, U_{ni}^h, \phi\right) + \sum_{T \in T_h} \left(L_n\, U_{ni}^h, w(\phi)\right)_T = (r_{ni}, \phi) + \sum_{T \in T_h} (r_{ni}, w(\phi))_T \ , \tag{10.13}$$

where $w(\phi)$ has to be defined with respect to the operator L_n (see, e.g., [19, 30, 38]). Two important classes of stabilized methods are the streamline diffusion and the more general Galerkin/least-squares finite element method.

The linear systems are solved by direct or iterative methods. While direct methods work quite satisfactorily in one-dimensional and even two-dimensional applications, iterative solvers such as Krylov subspace methods perform considerably better with respect to CPU-time and memory requirements for large two- and three-dimensional problems. We mainly use the BICGSTAB-algorithm [40] with ILU-preconditioning.

After computing the approximate intermediate values U_{ni}^h *a posteriori* error estimates can be used to give specific assessment of the error distribution. Considering a hierarchical decomposition

$$S_h^{q+1} = S_h^q \oplus Z_h^{q+1} \ , \tag{10.14}$$

where Z_h^{q+1} is the subspace that corresponds to the span of all additional basis functions needed to extend the space S_h^q to higher order, an attractive idea of an efficient error estimation is to bound the spatial error by evaluating its components in the space Z_h^{q+1} only. This technique is known as hierarchical error estimation and has been accepted to provide efficient and reliable assessment of spatial errors [8, 17, 2]. In [28], the hierarchical basis technique has been carried over to time-dependent nonlinear problems. Defining an *a posteriori* error estimator $E_{n+1}^h \in Z_h^{q+1}$ by

$$E_{n+1}^h = E_{n0}^h + \sum_{i=1}^{s} m_i E_{ni}^h \ , \tag{10.15}$$

with E_{n0}^h approximating the projection error of the initial value u_n in Z_h^{q+1} and E_{ni}^h estimating the spatial error of the intermediate value U_{ni}^h, the local spatial error for a finite element $T \in T_h$ can be estimated by $\eta_T := \|E_{n+1}^h\|_T$. The error estimator E_{n+1}^h is computed by linear systems which can be derived from (10.13). For practical computations the spatially global calculation of E_{n+1}^h is normally approximated by a small element-by-element calculation. This leads to an efficient algorithm for computing *a posteriori* error estimates which can be used to determine an adaptive strategy to improve the accuracy of the numerical approximation where needed. A rigorous *a posteriori* error analysis for a Rosenbrock–Galerkin finite element method applied to nonlinear parabolic systems is given in Lang [28]. In our applications we applied linear finite elements and measured the spatial errors in the space of quadratic functions.

In order to produce a nearly optimal mesh, those finite elements T having an error η_T larger than a certain threshold are refined. After the refinement improved finite element solutions U_{ni}^h defined by (10.13) are computed. The whole procedure solve-estimate-refine is applied several times until a prescribed spatial tolerance $\|E_{n+1}^h\| \leq TOL_x$ is reached. To maintain the nesting property of the finite element subspaces, coarsening takes place only after an accepted time step before starting the multilevel process at a new time. Regions of small errors are identified by their η-values.

10.4 Applications

10.4.1 Stability of Flame Balls

The profound understanding of premixed gas flames near extinction or stability limits is important for the design of efficient, clean-burning combustion engines and for the assessment of fire and explosion hazards in oil refineries, mine shafts, etc. Surprisingly, the near-limit behavior of very simple flames is still not well-known.

Since these phenomena are influenced by buoyant convection, typically experiments are performed in a μg environment. Under these conditions, transport mechanisms such as radiation and small Lewis number effects, the ratio of thermal diffusivity to the mass diffusivity, come into the play. Seemingly stable flame balls are one of the most exciting appearances which were accidentally discovered in drop-tower experiments by Ronney [34] and confirmed later in parabolic aircraft flights. First theoretical investigations on purely diffusion-controlled stationary spherical flames were done by Zeldovich [42]. Forty years later his flame balls were predicted to be unstable [11]. However, encouraged by the above new experimental discoveries, Buckmaster and collaborators [9] have shown that for low Lewis numbers flame balls can be stabilized including radiant heat loss which was not considered before (see Figure 10.1 for a configuration of a stationary flame ball). Nowadays there is an increasing interest in high-quality μg space experiments necessary to assess the steady properties and stability limits of flame balls (see NASA information at http://cpl.usc.edu/SOFBALL/sofball.html).

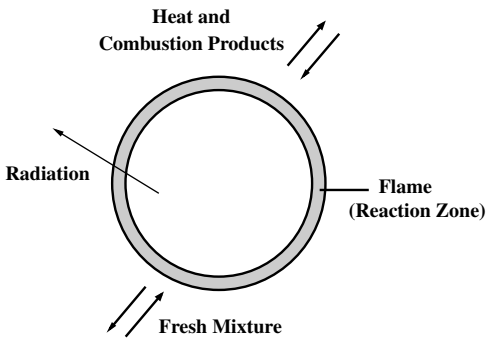


FIGURE 10.1
Configuration of a stationary flame ball. Diffusional fluxes of heat and combustion products (outwards) and of fresh mixture (inwards) together with radiative heat loss cause a zero mass-averaged velocity.

Although analytical modeling has identified the key physical ingredients of spherical premixed flames, quantitative confirmation can only come from detailed numerical simulations. Usually, spherically symmetric one-dimensional flame codes are used to investigate steady properties, stability limits, and dynamics of flame balls (see, e.g., [10, 41]). Higher dimensional simulations are very rare due to their great demand for local mesh adaptation in order to resolve the thin reaction layers. In [4] and [25] two-dimensional computations of flame balls were presented. Three-dimensional investigations using parallel architectures were published in [5].

The mathematical model we shall adopt is that of [4], which is based on the constant density assumption. In dimensionless form it reads

$$\begin{aligned} \partial_t T - \nabla^2 T &= w - s \,, \\ \partial_t Y - \frac{1}{Le} \nabla^2 Y &= -w \,, \\ w &= \frac{\beta^2}{2\,Le}\, Y \, \exp\left(\frac{\beta(T-1)}{1+\alpha(T-1)} \right) \,, \\ s &= c\, \frac{\overline{T}^4 - \overline{T}_u^4}{(\overline{T}_b - \overline{T}_u)^4} \,. \end{aligned} \tag{10.16}$$

Here, $T := (\overline{T} - \overline{T}_u)/(\overline{T}_b - \overline{T}_u)$ is the nondimensional temperature determined by the dimensional temperatures $\overline{T}$, $\overline{T}_u$, and $\overline{T}_b$, where the indices u and b refer to the unburnt and burnt state of an adiabatic plane flame, respectively. Y represents the mass fraction of the deficient component of the mixture. The chemical reaction rate w is modeled by a one-step Arrhenius term incorporating the dimensionless activation energy β, the Lewis number Le, and the heat release parameter $\alpha := (\overline{T}_b - \overline{T}_u)/\overline{T}_b$. Heat loss is generated by a radiation term s modeled for the optically thin limit. The strength of the radiative loss is mainly determined by the constant c, which depends on the Stefan–Boltzmann constant and the Planck length. These relatively simple equations are widely accepted to capture much of the essential physics of flame balls [9, 10]. Comparisons of analytical treatments to experimental results provide strong evidence of the model's validity.

In the following computations, the conditions have been chosen similar to the experiments made by Ronney [34] for a 6.5% H_2-air flame. We set $\overline{T}_u = 300K$, $\overline{T}_b = 830K$, $Le = 0.3$, $\beta = 10$, and derive $\alpha = 0.64$. In [34], additional CF_3Br as a tracer concentration in the mixture was used to increase the heat loss by radiation. Low concentration of CF_3Br yields cellular instability of the flame balls, whereas for increased heat loss due to an increased concentration of the tracer, stable flame balls can be observed. To simulate this behavior we use different values of c in (10.16), here $c = 0.01$ and $c = 0.1$.

The computational domain has to be sufficiently large in order to avoid any disturbance caused by the boundary. Typically, sizes of 100 times the flame ball radius are needed to obtain domain-independent solutions due to the long far-field thermal profiles [41]. In those cases, the conductive fluxes at the outer boundary are zero. We consider domains $\Omega = [-L, L]^d$, $d = 2, 3$, with $L = 200$ according to the initial flame radius $r_0 \in [0.2, 2.5]$. As initial conditions we take the analytic solution for a steady plane flame in the high activation-energy limit [4, 5] and in some calculations use a local stretching to generate an elliptic front. In the following we report on two different scenarios, unstable and quasi-stationary flame balls.

Unstable two-dimensional flame balls. We set $c = 0.01$ and take an initial elliptic flame with axis' ratio of 1:4. After a short time an instability develops which results in a local quenching of the flame as can be seen in Figures 10.2 and 10.3. After a while the flame is split into two separate smaller flames, which separate again and

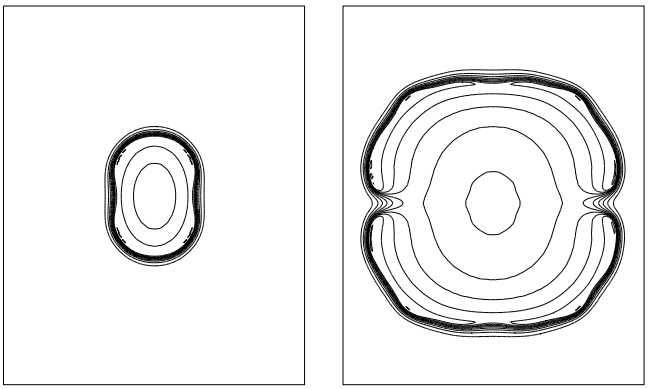


FIGURE 10.2
Two-dimensional flame ball with $Le = 0.3$**,** $c = 0.01$**. Iso-thermals** $T = 0.1, 0.2, \ldots, 1.0$ **at times** $t = 10$ **and** 30**.**

continue propagating. It can be seen that the dynamic spatial mesh chosen by our adaptive algorithm for $\mathrm{TOL}_t = \mathrm{TOL}_x = 0.0025$ is well-fitted to the behavior of the solution. More grid points are automatically placed in regions of high activity in order to resolve the steep solution gradients within the thin reaction layer.

Quasi-stationary two-dimensional flame balls. Fixing $c = 0.1$ in (10.16) and varying the initial radii for a circular flame in a large number of calculations, we found quasi-stationary flame ball configurations. In Figure 10.4 we have plotted the evolution of the integrated reaction rate $\int_\Omega w(t, x, y)\, dx\, dy$ for selected initial radii. For too small and too large radii, the flame is quickly extinguished. In between we observe a convergence process to a quasi-steady state characterized by a very slow decrease of the integrated reaction rate. The corresponding flame diameter is around 2. Similar results for $c = 0.05$ were reported in [4].

Splitting of three-dimensional flame balls. In the three-dimensional case we get a more complex pattern formation. Just to give an impression, we select one typical example taken from [5]. Starting with an ellipsoid having an axis ratio of 1:1:2, the flame ball is split along the z-axis due to the thermo-diffusive instabilities and further splitting occurs afterwards (see Figure 10.5). Although we were able to detect certain parameter regions for extinction resulting from excessive heat loss, we have not yet found configurations that are stable for longer time periods. This is the subject of current research.

10.4.2 Brine Transport in Porous Media

High-level radioactive waste is often disposed of in salt domes. The safety assessment of such a repository requires the study of groundwater flow enriched with salt. The observed salt concentration can be very high with respect to seawater, leading to sharp and moving freshwater-saltwater fronts. In such a situation, the basic equa-

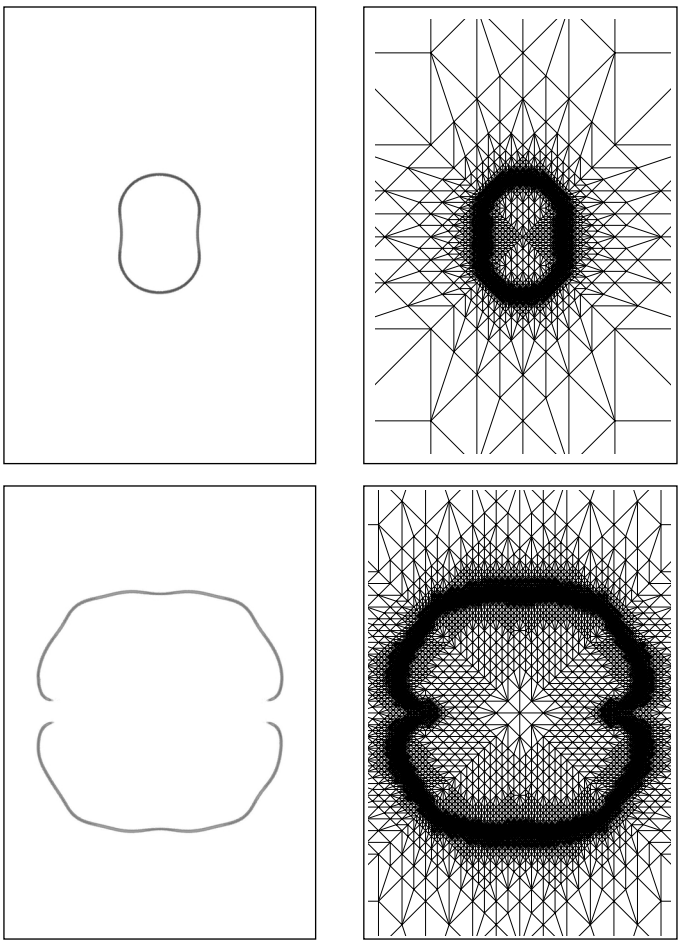


FIGURE 10.3
Two-dimensional flame ball with $Le = 0.3$, $c = 0.01$. Reaction rate w at times $t = 10, 30$, and corresponding grids.

tions of groundwater flow and solute transport have to be modified [24]. We use the physical model proposed by Trompert et al. [39] for a non-isothermal, single-phase, two-component saturated flow. It consists of the brine flow equation, the salt transport equation, and the temperature equation, and reads

$$n\rho\,(\beta\,\partial_t p + \gamma\,\partial_t w + \alpha\,\partial_t T) + \nabla\cdot(\rho\mathbf{q}) = 0\,, \tag{10.17}$$

$$n\rho\,\partial_t w + \rho\mathbf{q}\cdot\nabla w + \nabla\cdot\left(\rho J^w\right) = 0\,, \tag{10.18}$$

$$(nc\rho + (1-n)\rho^s c^s)\partial_t T + \rho\, c\,\mathbf{q}\cdot\nabla T + \nabla\cdot J^T = 0\,, \tag{10.19}$$

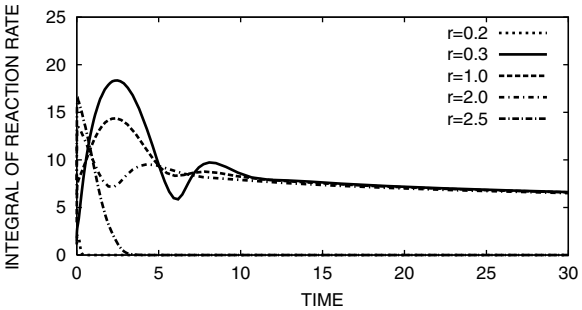


FIGURE 10.4
Two-dimensional flame ball with $Le = 0.3$, $c = 0.1$. Integrated reaction rate for different initial radii.

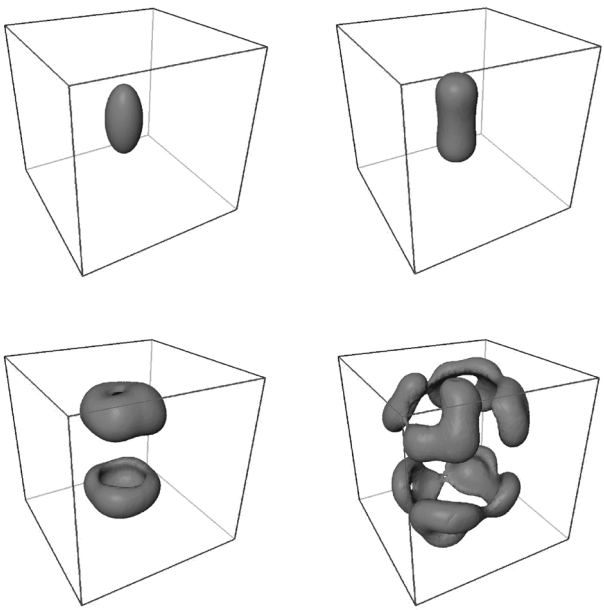


FIGURE 10.5
Three-dimensional flame ball with $Le = 0.3$, $c = 0.1$. Iso-thermals $T = 0.8$ at times $t = 0.0, 2.0, 5.0,$ and 8.0.

supplemented with the state equations for the density ρ and the viscosity μ of the fluid

$$\begin{aligned} \rho &= \rho_0 \exp\left(\alpha(T - T_0) + \beta(p - p_0) + \gamma w\right) , \\ \mu &= \mu_0 \left(1.0 + 1.85w - 4.0w^2\right) . \end{aligned}$$

Here, the pressure p, the salt mass fraction w, and the temperature T are the independent variables, which form a coupled system of nonlinear parabolic equations.

The Darcy velocity $\mathbf{q}$ of the fluid is defined as

$$\mathbf{q} = -\frac{K}{\mu}\left(\nabla p - \rho \mathbf{g}\right) \ ,$$

where K is the permeability tensor of the porous medium, which is supposed to be of the form $K = \mathrm{diag}(k)$, and $\mathbf{g}$ is the acceleration of gravity vector. The salt dispersion flux vector J^w and the heat flux vector J^T are defined as

$$\begin{aligned} J^w &= -\left((n d_m + \alpha_T |\mathbf{q}|)\, I + \frac{\alpha_L - \alpha_T}{|\mathbf{q}|} \mathbf{q}\mathbf{q}^T \right) \nabla w \ , \\ J^T &= -\left((\kappa + \lambda_T |\mathbf{q}|)\, I + \frac{\lambda_L - \lambda_T}{|\mathbf{q}|} \mathbf{q}\mathbf{q}^T \right) \nabla T \ , \end{aligned}$$

where $|\mathbf{q}| = \sqrt{\mathbf{q}^T \mathbf{q}}$.

Writing the system of the three balance equations (10.17) through (10.19) in the form (10.8), we find for the 3×3 matrix B

$$B(p, w, T) = \begin{pmatrix} n\rho\beta & n\rho\gamma & n\rho\alpha \\ 0 & n\rho & 0 \\ 0 & 0 & nc\rho + (1-n)\rho^s c^s \end{pmatrix} .$$

Since the compressibility coefficient β is very small, the matrix B is nearly singular and, as known [23, VI.6], linearly implicit time integrators suitable for differential algebraic systems of index 1 do not give precise results. This is mainly due to the fact that for $\beta = 0$ the matrix B becomes singular and additional consistency conditions have to be satisfied to avoid order reduction. We have applied the Rosenbrock solver ROWDAIND2 [31], which handles both situations, $\beta = 0$ and $\beta \neq 0$.

An additional feature of the model is that the salt transport equation (10.18) is usually dominated by the advection term. In practice, global Peclet numbers can range between 10^2 and 10^4, as reported in [39]. On the other hand, the temperature and the flow equation are of standard parabolic type with convection terms of moderate size.

Two-dimensional warm brine injection. This problem was taken from [39]. We consider a (very) thin vertical column filled with a porous medium. This justifies the use of a two-dimensional flow domain $\Omega = \{(x, y) : 0 < x, y < 1\}$ representing a vertical cross-section. The acceleration of gravity vector points downward and takes the form $\mathbf{g} = (0, -g)^T$, where the gravity constant g is set to 9.81. The initial values at $t = 0$ are

$$p(x, y, 0) = p_0 + (1 - y)\rho_0 g, \quad w(x, y, 0) = 0 \,, \ \text{and} \ T(x, y, 0) = T_0 \,.$$

The boundary conditions are described in Figure 10.6. We set $w_b = 0.25$, $T_b = 292.0$, and $q_b = 10^{-4}$. The remaining parameters used in the model are given in Table 10.3.

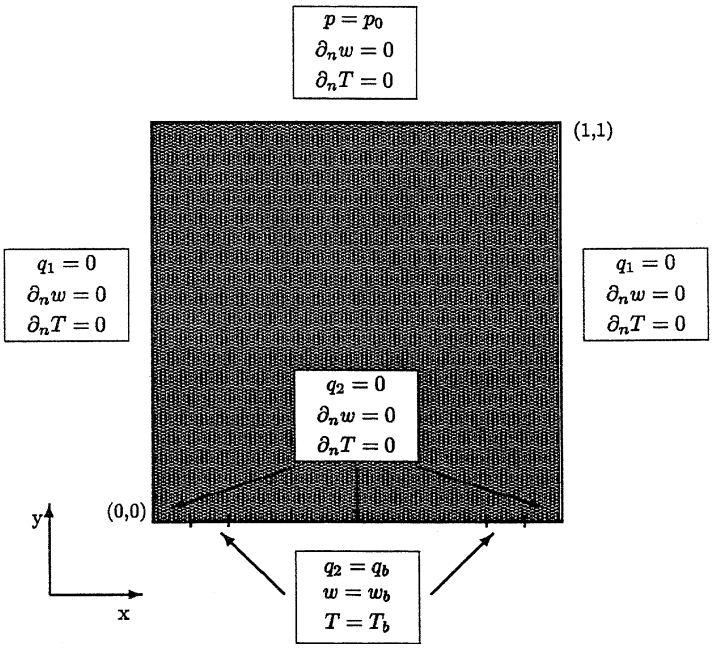


FIGURE 10.6
Two-dimensional brine transport. Computational domain and boundary conditions for $t > 0$. The two gates where warm brine is injected are located at $(x, y) : \frac{1}{11} \le x \le \frac{2}{11}$, $\frac{9}{11} \le x \le \frac{10}{11}$, $y = 0$.

Table 10.3 Parameters of the Two-Dimensional Brine Transport Model

n	Porosity	0.4	
k	Permeability	10^{-10}	m^2
d_m	Molecular diffusion	0.0	$m^2 s^{-1}$
α_T	Transversal dispersivity	0.002	m
α_L	Longitudinal dispersivity	0.01	m
c	Heat capacity	4182	$J\,kg^{-1}K^{-1}$
c^s	Solid heat capacity	840	$J\,kg^{-1}K^{-1}$
κ	Heat conductivity	4.0	$J\,s^{-1}m^{-1}K^{-1}$
λ_T	Transversal heat conductivity	0.001	$J\,m^{-2}K^{-1}$
λ_L	Longitudinal heat conductivity	0.01	$J\,m^{-2}K^{-1}$
ρ^s	Solid density	2500	$kg\,m^{-3}$
ρ_0	Freshwater density	1000	$kg\,m^{-3}$
T_0	Reference temperature	290	K
p_0	Reference pressure	10^5	$kg\,m^{-1}s^{-2}$
α	Temperature coefficient	$-3.0 \cdot 10^{-4}$	K^{-1}
β	Compressibility coefficient	$4.45 \cdot 10^{-10}$	$m\,s^2 kg^{-1}$
γ	Salt coefficient	$ln(1.2)$	
μ_0	Reference viscosity	10^{-3}	$kg\,m^{-1}s^{-1}$

Warm brine is injected through two gates at the bottom. This gives rise to sharp fronts between salt and fresh water, which have to be resolved with fine meshes in the neighborhood of the gates, see Figure 10.7. Later the solutions smooth out with time until the porous medium is filled completely with brine. Our computational results are comparable to those obtained in [39] with a method of lines approach coupled with a local uniform grid refinement. In Figure 10.8 we show the time steps and the degrees of freedom chosen by the KARDOS solver to integrate over $t \in [0, 2 \cdot 10^4]$. The curves nicely reflect the high dynamics at the beginning in both time and space, while larger time steps and coarser grids are selected in the final part of the simulation.

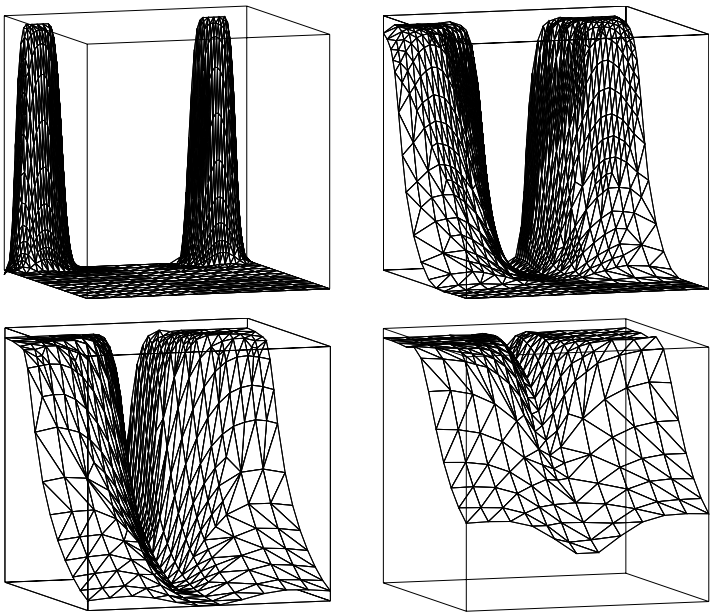


FIGURE 10.7
Two-dimensional brine transport. Distribution of salt concentration at $t = 500, 5000, 10000$**, and** 20000 **with corresponding spatial grids.**

Three-dimensional pollution with salt water. Here, we consider Problem III of [3] and simulate a salt pollution of fresh water flowing from left to right through a tank $\Omega = \{(x, y, z) : 0 \le x \le 2.5, 0 \le y \le 0.5, 0 \le z \le 1.0\}$ filled with a porous medium. The flow is supposed to be isothermal ($\alpha = 0$) and incompressible ($\beta = 0$). Hence, the problem now consists of two PDEs with a singular 2×2 matrix $B(p, w)$ multiplying the vector of temporal derivatives. The acceleration of gravity vector takes the form $\mathbf{g} = (0, 0, -g)^T$.

The brine having a salt mass fraction $w_b = 0.0935$ is injected through a small slit $S = \{(x, y, 1) : 0.375 \le x \le 0.4375,\ 0.25 \le y \le 0.3125\}$ at the top of the tank. We note that the slit chosen here differs slightly from that used in [3]. The initial

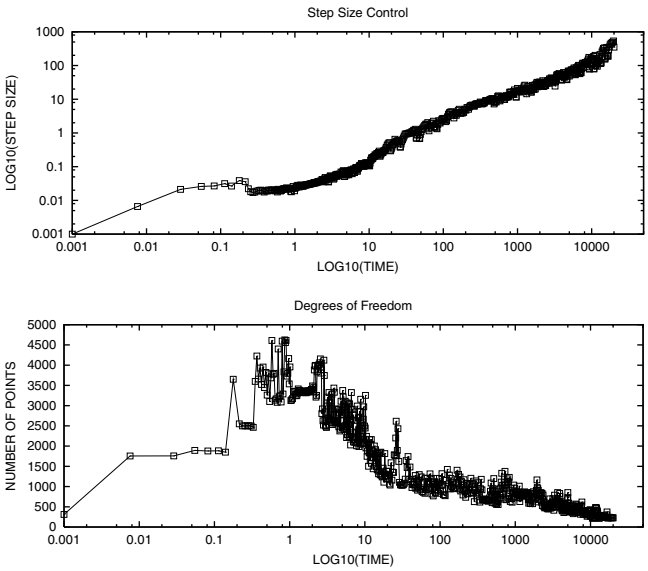


FIGURE 10.8
Two-dimensional brine transport. Evolution of time steps and number of spatial discretization points for $TOL_t = TOL_x = 0.005$.

values at $t = 0$ are taken as

$$p(x, y, z, 0) = p_0 + (0.03 - 0.012x + 1.0 - z)\rho_0 g, \quad w(x, y, z, 0) = 0\,,$$

and the boundary conditions are

$$\begin{aligned}
p &= p(x, y, z, 0)\,,\ w = 0\,, && \text{on } x = 0\,,\\
p &= p(x, y, z, 0)\,,\ \partial_n w = 0\,, && \text{on } x = 2.5\,,\\
q_2 &= 0\,,\ \partial_n w = 0\,, && \text{on } y = 0 \text{ and } y = 1\,,\\
q_3 &= 0\,,\ \partial_n w = 0\,, && \text{on } z = 0 \text{ and } \{z = 1\} \setminus S\,,\\
\rho q_3 &= -0.0495,\ w = w_b = 0.0935\,, && \text{on } S\,.
\end{aligned}$$

The parameters used in the three-dimensional simulation are given in Table 10.4. Additionally, the state equation for the viscosity of the fluid is modified to

$$\mu = \mu_0 \left(1.0 + 1.85w - 4.1w^2 + 44.5w^3\right)\,.$$

In Figure 10.9 we show the distribution of the salt concentration in the plane $y = 0.28125$ after 2 and 4 h. The pollutant is slowly transported by the flow while sinking to the bottom of the tank. The steepness of the solution is higher in the back of the pollution front, which causes fine meshes in this region. Despite the dominating convection terms, no wiggles are visible, especially at the inlet. An interesting observation is the unexpected drift in front of the solution — a phenomenon which was also observed by Blom and Verwer [3].

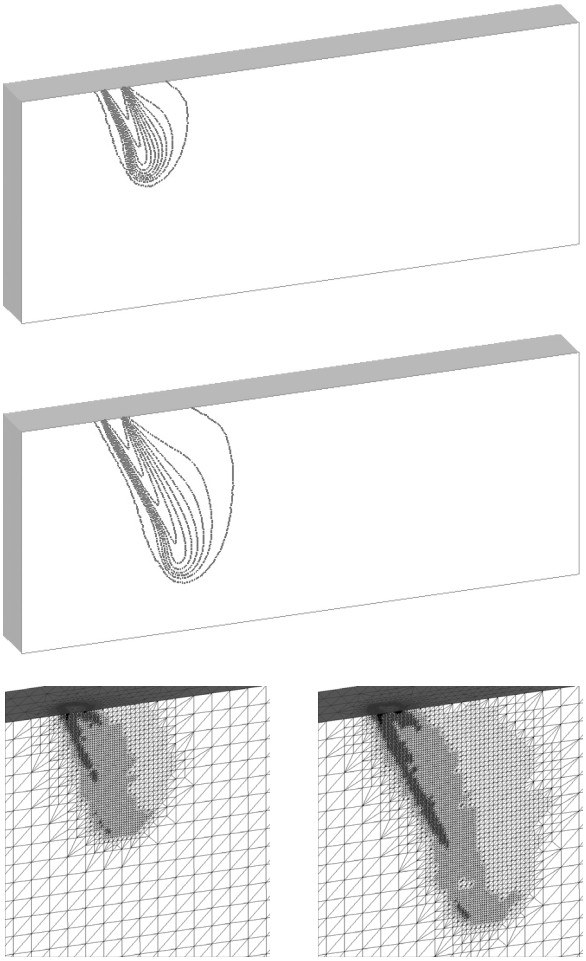


FIGURE 10.9
Three-dimensional brine transport. Level lines of the salt concentration $w = 0.0, 0.01, \ldots, 0.09$, **in the plane** $y = 0.28125$ **after 2 h (top) and 4 h (middle), and corresponding spatial grids (bottom) in the neighborhood of the inlet.**

10.5 Conclusion

Dynamical process simulation of complex real-life problems advises the use of modern algorithms, which are able to judge the quality of their numerical approximations and to determine an adaptation strategy to improve their accuracy in both the time and the space discretization. This chapter presented a combination of efficient linearly implicit time integrators of Rosenbrock type and error-controlled grid

Table 10.4 Parameters of the Three-Dimensional Brine Transport Model

n	=	0.35	γ	=	$ln(2)$	nd_m	=	10^{-9}
κ	=	$7.18 \cdot 10^{-11}$	α_T	=	0.001	α_L	=	0.01
p_0	=	0.0	μ_0	=	0.001	ρ_0	=	1000

improvement based on a multilevel finite element method. This approach leads to a minimization of the degrees of freedom necessary to reach a prescribed error tolerance. The savings in computing time are substantial and allow the solution of even complex problems in a moderate range of time.

References

[1] R.E. Bank, PLTMG: A Software Package for Solving Elliptic Partial Differential Equations — User's Guide 8.0, *SIAM,* 1998.

[2] R.E. Bank and R.K. Smith, A Posteriori Error Estimates Based on Hierarchical Bases, *SIAM J. Numer. Anal.,* **30**, (1993), 921–935.

[3] J.G. Blom and J.G. Verwer, VLUGR3: A Vectorizable Adaptive Grid Solver for PDEs in 3D, I. Algorithmic Aspects and Applications, *Appl. Numer. Math.,* **16**, (1994), 129–156.

[4] H. Bockhorn, J. Fröhlich, and K. Schneider, An Adaptive Two-Dimensional Wavelet-Vaguellette Algorithm for the Computation of Flame Balls, *Combust. Theory Modeling,* **3**, (1999), 177–198.

[5] H. Bockhorn, J. Fröhlich, W. Gerlinger, and K. Schneider, Numerical Investigations on the Stability of Flame Balls, in: K. Papailiou et al., eds., *Computational Fluid Dynamics '98, Vol. 1*, John Wiley & Sons, New York, (1998), 990–995.

[6] F.A. Bornemann, An Adaptive Multilevel Approach to Parabolic Equations. II. Variable-Order Time Discretization Based on a Multiplicative Error Correction, *IMPACT of Comput. in Sci. and Engrg.,* **3**, (1991), 93–122.

[7] F.A. Bornemann, An Adaptive Multilevel Approach to Parabolic Equations. III. 2D Error Estimation and Multilevel Preconditioning, *IMPACT of Comput. in Sci. and Engrg.,* **4**, (1992), 1–45.

[8] F.A. Bornemann, B. Erdmann, and R. Kornhuber, A Posteriori Error Estimates for Elliptic Problems in Two and Three Space Dimensions, *SIAM J. Numer. Anal.,* **33**, (1996), 1188–1204.

[9] J.D. Buckmaster, G. Joulin, and P.D. Ronney, Effects of Heat Loss on the Structure and Stability of Flame Balls, *Combust. Flame,* **79**, (1990), 381–392.

[10] J.D. Buckmaster, M. Smooke, and V. Giovangili, Analytical and Numerical Modeling of Flame-Balls in Hydrogen-Air Mixtures, *Combust. Flame,* **94**, (1993), 113–124.

[11] J.D. Buckmaster and S. Weeratunga, The Stability and Structure of Flame-Bubbles, *Comb. Sci. Tech.,* **35**, (1984), 287–296.

[12] K. Dekker and J.G. Verwer, *Stability of Runge–Kutta Methods for Stiff Nonlinear Differential Equations,* North-Holland Elsevier Science Publishers, 1984.

[13] P. Deuflhard, Uniqueness Theorems for Stiff ODE Initial Value Problems, in: D.F. Griffiths and G.A. Watson, eds., *Numerical Analysis 1989, Proceedings of the 13th Dundee Conference, Pitman Research Notes in Mathematics Series 228,* Longman Scientific and Technical, (1990), 74–87.

[14] P. Deuflhard, Recent Progress in Extrapolation Methods for Ordinary Differential Equations, *SIAM Rev.,* **27**, (1985), 505–535.

[15] P. Deuflhard and F. Bornemann, *Numerische Mathematik II, Integration Gewöhnlicher Differentialgleichungen,* De Gruyter Lehrbuch, Berlin, New York, 1994.

[16] P. Deuflhard, J. Lang, and U. Nowak, Adaptive Algorithms in Dynamical Process Simulation, in: H. Neunzert, ed., *Progress in Industrial Mathematics at ECMI'94,* Wiley–Teubner, (1996), 122–137.

[17] P. Deuflhard, P. Leinen, and H. Yserentant, Concepts of an Adaptive Hierarchical Finite Element Code, *IMPACT of Comput. in Sci. and Engrg.,* **1**, (1989), 3–35.

[18] B. Erdmann, J. Lang, and R. Roitzsch, KASKADE Manual, Version 2.0, Report TR93-5, Konrad–Zuse–Zentrum für Informationstechnik, Berlin, 1993.

[19] L.P. Franca and S.L. Frey, Stabilized Finite Element Methods, *Comput. Methods Appl. Mech. Engrg.,* **99**, (1992), 209–233.

[20] K. Gustafsson, Control-Theoretic Techniques for Stepsize Selection in Implicit Runge–Kutta Methods, *ACM Trans. Math. Software,* **20**, (1994), 496–517.

[21] K. Gustafsson, M. Lundh, and G. Söderlind, A PI Stepsize Control for the Numerical Solution of Ordinary Differential Equations. *BIT* **28**, (1988), 270–287.

[22] E. Hairer, S.P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations I, Nonstiff Problems,* Springer-Verlag, Berlin, 1987.

[23] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II, Stiff and Differential-Algebraic Problems,* Second Revised Edition, Springer-Verlag, Berlin, 1996.

[24] S.M. Hassanizadeh and T. Leijnse, On the Modeling of Brine Transport in Porous Media, *Water Resources Research,* **24**, (1988), 321–330.

[25] L. Kagan and G. Sivashinski, Self-Fragmentation of Nonadiabatic Cellular Flames, *Combust. Flames,* **108** (1997), 220–226.

[26] J. Lang and A. Walter, A Finite Element Method Adaptive in Space and Time for Nonlinear Reaction-Diffusion Systems, *IMPACT of Comput. in Sci. and Engrg.,* **4**, (1992), 269–314.

[27] J. Lang, Adaptive FEM for Reaction-Diffusion Equations, *Appl. Numer. Math.,* **26**, (1998), 105–116.

[28] J. Lang, Adaptive Multilevel Solution of Nonlinear Parabolic PDE Systems. Theory, Algorithm, and Applications, *Lecture Notes in Computational Science and Engineering,* Vol. 16, Springer-Verlag, Berlin, 2000.

[29] J. Lang and J. Verwer, ROS3P — an Accurate Third-Order Rosenbrock Solver Designed for Parabolic Problems, *Report MAS-R0013, CWI, Amsterdam,* 2000.

[30] G. Lube and D. Weiss, Stabilized Finite Element Methods for Singularly Perturbed Parabolic Problems, *Appl. Numer. Math.,* **17**, (1995), 431–459.

[31] Ch. Lubich and M. Roche, Rosenbrock Methods for Differential-Algebraic Systems with Solution-Dependent Singular Matrix Multiplying the Derivative, *Comput.,* **43**, (1990), 325–342.

[32] M. Roche, Runge–Kutta and Rosenbrock Methods for Differential-Algebraic Equations and Stiff ODEs, Ph.D. thesis, Université de Genève, 1988.

[33] M. Roche, Rosenbrock Methods for Differential Algebraic Equations, *Numer. Math.,* **52**, (1988), 45–63.

[34] P.D. Ronney, Near-Limit Flame Structures at Low Lewis Number, *Combust. Flame,* **82**, (1990), 1–14.

[35] H.H. Rosenbrock, Some General Implicit Processes for the Numerical Solution of Differential Equations, *Computer J.,* (1963), 329–331.

[36] G. Steinebach, Order-Reduction of ROW-methods for DAEs and Method of Lines Applications, Preprint 1741, Technische Hochschule Darmstadt, Germany, 1995.

[37] K. Strehmel and R. Weiner, Linear-implizite Runge–Kutta–Methoden und ihre Anwendungen, Teubner Texte zur Mathematik 127, Teubner Stuttgart, Leipzig, 1992.

[38] L. Tobiska and R. Verfürth, Analysis of a Streamline Diffusion Finite Element Method for the Stokes and Navier–Stokes Equation, *SIAM J. Numer. Anal.,* **33**, (1996), 107–127.

[39] R.A. Trompert, J.G. Verwer, and J.G. Blom, Computing Brine Transport in Porous Media with an Adaptive-Grid Method, *Int. J. Numer. Meth. Fluids,* **16**, (1993), 43–63.

[40] H.A. van der Vorst, BI-CGSTAB: A fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems, *SIAM J. Sci. Stat.,* **13**, (1992), 631–644.

[41] M.S. Wu, P.D. Ronney, R.O. Colantonio, and D.M. Vanzandt, Detailed Numerical Simulation of Flame Ball Structure and Dynamics, *Combust. Flame,* **116**, (1999), 387–397.

[42] Ya.B. Zeldovich, *Theory of Combustion and Detonation of Gases,* Academy of Sciences (USSR), 1944.

Chapter 11

Unstructured Adaptive Mesh MOL Solvers for Atmospheric Reacting-Flow Problems

M. Berzins, A.S. Tomlin, S. Ghorai, I. Ahmad, and J. Ware

11.1 Introduction

In this chapter, the method of lines (MOL) is applied to computational models of reacting flow arising from atmospheric applications. These computational models describe the chemical transformations and transport of species in the troposphere and have an essential role in understanding the complex processes which lead to the formation of pollutants such as greenhouse gases, acid rain, and photochemical oxidants. In order to make good comparisons with the limited experimental data available, it is important to have a high degree of computational resolution, but at the same time to model emissions from many different sources and over large physical domains. This chapter is thus concerned with how to achieve this by using MOL combined with spatial mesh adaptation techniques.

Achieving high resolution in air pollution models is a difficult challenge because of the large number of species present in the atmosphere. The number of chemical rate equations that need to be solved rises with the number of species, and for high resolution 3-dimensional calculations, detailed chemical schemes can become prohibitively large. The range of reaction time-scales often leads to stiff systems of differential equations which require more expensive implicit numerical solvers. Previous work has shown [31, 32, 33, 12, 13] that coarse horizontal resolution can have the effect of increasing horizontal diffusion to values many times greater than that described by models, resulting in the smearing of pollutant profiles and an underestimation of maximum concentration levels. A review paper by Peters et al. [22] highlights the importance of developing more efficient grid systems for the next generation of air pollution models in order to "capture important smaller-scale atmospheric phenomena."

In general, the effects of mesh resolution have been well noted by the atmospheric modeling community and attempts have been made to improve mesh resolution at

the same time as trying to avoid excessive extra computational work. The usual approach is to use nested or telescopic grids, where the mesh is refined in certain regions of the horizontal domain which are considered of interest [15, 24, 30, 26]. This may include, for example, regions of high emissions such as urban areas, or close to regions where significant monitoring is taking place. Previous work [32] has shown, however, that such telescopic grids often cannot resolve plume structures occurring outside of the nested regions and that adaptive refinement in the horizontal domain can provide higher accuracy without entailing large extra computational costs. The primary reason is that away from concentrated sources, such models use large grids of up to 50 km. Since dispersion can carry species distances of hundreds of kilometers from the source, such predescribed telescopic gridding models could still lead to inaccurate downwind profiles as the plumes travel into those areas with larger grids. This is a particular problem when modeling species such as ozone, where the chemical time-scale of pollutant formation is such that the main pollution episodes occur at very long distances downwind of the sources of photochemical precursors. The regions of steep spatial gradients of species such as ozone will move with time according to the wind-field present and the spatial distribution of emissions. A reliable solution can only be obtained if the mesh can be refined accordingly. The fine-scale grids used in present regional scale models are of the order of 10 to 20 km. For a power plant plume with a width of approximately 20 km, it is impossible to resolve the fine structure within the plume using grids of this size. Furthermore, to refine the mesh *a priori,* according to the path of the plume, would be an impossible task since the plume position is a complicated function of many factors, including reaction, deposition, and transport. There is a need for the application of methods which can refine the grid according to where the solution requires it, i.e., time-dependent adaptive algorithms. While there have been some applications of adaptive grids for environmental modeling, e.g., Skamarock et al. [27], as yet these methods have not been implemented in standard air quality models.

This chapter is based on the work done by the authors in applying adaptive gridding techniques, which automatically refine the mesh in regions of high spatial error, and illustrates the benefits this can bring over the telescopic approach in which mesh refinement is only used close to a pollution source. The first part of this chapter (Sections 11.2 to 11.4) describes the algorithms used and presents results for the 1D hyperbolic conservation law with a nonlinear source term, of Leveque and Yee [18]. This deceptively simple problem may be used to show that spurious numerical solution phenomena, such as incorrect wave speeds may occur when insufficient spatial and temporal resolution are used. Sections 11.5 to 11.10 of the chapter will provide a summary of the results for more complex two-dimensional atmospheric problems (see [32]) while three-dimensional problems (see [33, 12]) are considered in Sections 11.10 to 11.14. The general approach used here is to employ positivity-preserving spatial discretization schemes in the method of lines to reduce the partial differential equation (PDE) to a system of ordinary differential equations (ODEs) in time. For reacting-flow problems, the numerical results will show that spatial mesh points should be chosen with great care to reflect the true solution of the PDE and to

avoid generating significant but spurious numerical solution features. This is achieved here by using adaptive mesh algorithms [3] to control the spatial discretization error by refining and coarsening the mesh.

As reacting-flow problems require the use of implicit methods to resolve the fast transients associated with some chemistry species, the cost of using implicit methods may be high unless great care is taken with numerical linear algebra. In the present work this is done by making use of an approach developed for atmospheric chemistry solvers [35, 2]. This approach uses a Gauss–Seidel iteration applied to the source terms alone. The advective terms are effectively treated explicitly but without introducing a splitting error. In three dimensions because of the need to preserve positivity of the solution and to be more concerned about efficiency, we have also used a more traditional operator-splitting approach. In particular, the overall conclusion to be drawn from the computational evidence for one-, two-, and three-dimensional problems is that having good mesh resolution in certain parts of the solution domain is of critical importance with regard to obtaining a meaningful solution.

11.2 Spatial Discretization and Time Integration

The 1D Leveque and Yee problem [18] is given by

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = -\psi(u) \quad x \in [0, \infty], \quad \psi(u) = \mu u(u-1)(u-0.5) \tag{11.1}$$

and is linear advection with a source term that is "stiff" for large μ. The initial and boundary values (at $x = 0$) are defined by

$$u(x, 0) = u_0(x) = u_L = 1, \quad x \leq x_d; \quad u_R = 0, \quad x > x_d$$

where $x_d = 0.1$ or 0.3 in the cases considered here. The infinite domain will also be truncated to [0, 1] for the cases considered here, as this is sufficient to demonstrate the behavior of the methods employed. A simple outflow boundary condition is then used at $x = 1$. The solution of Equation (11.1) is a discontinuity moving with constant speed and has a potentially large source term that only becomes active at the discontinuity [18].

Define a spatial mesh $0 = x_1 < \cdots < x_N = 1$ and the vector of values $\underline{U}$ with components $U_i(t) \approx u(x_i, t)$ where $u(x, t)$ is the exact solution to the PDE. We define $U_i(t)$ as the exact solution to the ordinary differential equation (ODE) system derived by spatial semi-discretization of the PDE and given by

$$\dot{\underline{U}} = \underline{F}_N(t, \underline{U}(t)), \qquad \underline{U}(0) \text{ given} . \tag{11.2}$$

This true solution $[\underline{U}(t_n)]_{n=0}^{p}$ is approximated by $[\underline{V}(t_n)]_{n=0}^{p}$ at a set of discrete times $0 = t_0 < t_1 < \cdots < t_p = t_e$ by the time integrator. The form of the ODE system

given by Equation (11.2) at time t is given by

$$\underline{F}_N(t_n, \underline{U}(t_n)) = \underline{F}_N^f(t_n, \underline{U}(t_n)) + \underline{F}_N^s(t_n, \underline{U}(t_n)) \,, \tag{11.3}$$

where the superscripts f and s denote the flow and source term parts of the function $\underline{F}$ as defined below. The function $\underline{F}_N^f(t_n, \underline{U}(t_n))$ is the second-order limited discretization of the advective terms in Equation (11.1) whose components are given by

$$F_j^f(t, \underline{U}(t)) \;=\; -\left[1 + \frac{(B(r_j, 1)}{2} - \frac{B(r_{j-1}, 1)}{2r_{j-1}}\right] \frac{(U_j(t) - U_{j-1}(t))}{\triangle x} \,. \tag{11.4}$$

The function B is a limiter such as that of van Leer: (see [3])

$$B(r_j, 1) \;=\; \frac{r_j \;+\; |\, r_j \,|}{1 \;+\; r_j}, \;\text{ and }\; r_j \;=\; \frac{U_{j+1}(t) - U_j(t)}{U_j(t) - U_{j-1}(t)} \,. \tag{11.5}$$

The vector $\underline{F}_N^s(t, \underline{U}(t))$ represents the approximate spatial integration of the source term which is defined by $\frac{1}{\triangle x}\int_{x_{j-\frac{1}{2}}}^{x_{j+\frac{1}{2}}} \psi(U(x,t))dx$ and is evaluated by using the midpoint quadrature rule so that its jth component is:

$$F_j^s(t, U_j(t)) \;=\; \psi(U_j(t)) \,. \tag{11.6}$$

The time integration method used here (mostly for simplicity of analysis) is the Backward Euler method defined by

$$\underline{V}(t_{n+1}) \;=\; \underline{V}(t_n) \;+ \underline{F}_N(t_{n+1}, \underline{V}(t_{n+1})) \,. \tag{11.7}$$

In the case when a modified Newton method is used to solve the nonlinear equations at each timestep, this constitutes the major computational task of a method of lines calculation. In cases where large ODE systems result from the discretization of flow problems with many chemical species, the CPU times may be excessive unless special iterative methods are used.

The approach of [4] is used to neglect the advective terms $J_f = \frac{\partial \underline{F}^f}{\partial \underline{V}}$, and to concentrate on the Jacobian of the source terms $J_s = \frac{\partial \underline{F}^s}{\partial \underline{V}}$ when forming the Newton iteration matrix. This approach, in the case when no source terms are present, corresponds to using functional iteration for the advective calculation, see [2, 4]. The matrix $I - \triangle t \gamma J_s$ is the Newton iteration matrix of that part of the ODE system corresponding to the discretization of the time derivatives and the source terms alone. This matrix is thus block-diagonal with as many blocks as there are spatial elements and with each block having as many rows and columns as there are PDEs. The fact that a single block relates only to the chemistry within one cell means that each block's equations may be solved independently by using a Gauss–Seidel iteration. This approach has been used with great success for atmospheric chemistry problems [35]. The nonlinear equations iteration employed here may thus be written as

$$[I - \triangle t \; J_s]\left[\underline{V}^{m+1}(t_{n+1}) - \underline{V}^m(t_{n+1})\right] = \underline{r}\left(t_{n+1}^m\right) \tag{11.8}$$

where $\underline{r}\left(t_{n+1}^m\right) = -\underline{V}^m(t_{n+1}) + \underline{V}(t_n) + \triangle t\,\underline{F}_N(t_{n+1}, \underline{V}^m(t_{n+1}))$. Providing that the iteration converges, this approximation has no adverse impact on accuracy. In order for this iteration to converge with a rate of convergence r_c it is necessary [2] that

$$|| \, [I - \triangle t\, J_s]^{-1} \, \triangle t \; J_f \, || = r_c \qquad \text{where} \;\; r_c < 1 \, . \tag{11.9}$$

Using the identity $\| \, ab \, \| \leq \| \, a \, \| \| \, b \, \|$, and defining J_f^* as $J_f^* = (\triangle x) J_f$ gives:

$$\frac{\triangle t}{\triangle x} || \; J_f^* \; || \;\; \leq \;\; r_c \; || \; [I - \triangle t\, J_s] \; || \, . \tag{11.10}$$

Hence the convergence restriction may be interpreted as a CFL type condition. For example, in the case of the PDE in (11.1), $[I - \triangle t\, J_s]$ is a diagonal matrix with entries $1 + \triangle t \mu \, \frac{\partial \psi}{\partial V}$ where

$$\frac{\partial \psi}{\partial V} = p(V) \tag{11.11}$$

and where $p(V) = 3V^2 - 3V + 0.5$ gives a CFL type condition that allows larger timesteps as μ increases. The function $p(V)$ is bounded between the values 0.5 and -0.25 for solution values in the range [0, 1].

11.3 Space-Time Error Balancing Control

Hyperbolic PDEs are often solved by using a CFL condition to select the timestep. The topic of choosing a stable stepsize for such problems has been considered in detail by Berzins and Ware [6]. Although a CFL condition indicates when the underlying flow without reactions is stable, it is still necessary to get the required accuracy for the chemistry terms. In most time dependent PDE codes, either a CFL stability control is employed or a standard ODE solver is used which controls the local error $\underline{l}_{n+1}(t_{n+1})$ with respect to a user supplied accuracy tolerance. Efficient time integration requires that the spatial and temporal errors are roughly the same order of magnitude. The need for spatial error estimates unpolluted by temporal error requires that the spatial error is the larger of the two. One alternative approach developed by Berzins [3, 4] is to use a local error per unit step control in which the time local error (denoted by $\underline{le}(t)$) is controlled so as to be smaller than the local growth in the spatial error over the timestep (denoted by $\underline{est}(t)$). In the case of the Backward Euler method, the standard local error estimate at t_{n+1} is defined as $\underline{le}(t_{n+1})$ and is estimated in standard ODE codes by

$$\begin{aligned} \underline{le}(t_{n+1}) \quad &= \quad \frac{\triangle t}{2} \left[\underline{F}_N(t_{n+1}, \underline{V}(t_{n+1})) - \underline{F}_N(t_n, \underline{V}(t_n)) \right] \, . \\ &\approx \quad \frac{\triangle t^2}{2} \ddot{\underline{V}}(t_{n+1}) \end{aligned} \tag{11.12}$$

where the function $\underline{F}$ is defined by Equation (11.2). The error control of [3] is defined by

$$\| \underline{le}_{n+1}(t_{n+1}) \| \leq \epsilon \| \underline{est}(t_{n+1}) \| \tag{11.13}$$

where $0 < \epsilon < 1$ is a balancing factor and $\underline{est}(t_{n+1})$ represents the local growth in time of the spatial discretization error from t_n to t_{n+1}, assuming that the error is zero at t_n. Once the primary solution has been computed using the method of Section 11.2, a secondary solution is estimated at the same time step with an upwind scheme of different order and a different quadrature rule for source-term integration. The difference of these two computed solutions is then taken as an estimate of the local growth in time of the spatial discretization error in the same way as in [3]. The primary solution $\underline{V}(t_{n+1})$ starting from $\underline{V}(t_n)$ is computed in the standard way as described in Section 11.2. The secondary solution $\underline{W}(t_{n+1})$ is computed by solving

$$\dot{\underline{W}}(t) = \underline{G}^f(t, \underline{W}(t)) + \underline{G}^s(t, \underline{W}(t)), \quad \underline{W}(t_n) = \underline{V}(t_n) . \tag{11.14}$$

with initial value $\underline{V}_n$, where$\underline{G}^f$ and $\underline{G}^s$ are the first-order advective term and the source terms which are evaluated using a linear approximation on each interval and the trapeziodal rule, i.e.,

$$\begin{aligned} G_j^f(t, W_j(t)) &= -\frac{(W_j(t) - W_{j-1}(t)}{\triangle x} \\ G_j^s(t, W_j(t)) &= \frac{1}{4}(\psi(W_{j-1}(t)) + 2\psi(W_j(t)) + \psi(W_{j+1}(t))) . \end{aligned} \tag{11.15}$$

Estimating $\underline{est}(t_{n+1})$ by applying the Backward Euler Method to (11.14) subtracted from (11.7) with one iteration of the modified Newton iteration of the previous section, as in [4], gives

$$\begin{aligned} [I - \triangle t J_s]\left[\underline{est}\,(t_{n+1})\right] = \triangle t \Big[& \underline{F}^f\left(t_{n+1}, \underline{V}\,(t_{n+1})\right) - \underline{G}^f\left(t_{n+1}, \underline{V}\,(t_{n+1})\right) \\ & + \underline{F}^s\left(t_{n+1}, \underline{V}\,(t_{n+1})\right) - \underline{G}^s\left(t_{n+1}, \underline{V}\,(t_{n+1})\right) \Big] \end{aligned} \tag{11.16}$$

where $\underline{est}(t_{n+1}) \approx \underline{V}(t_{n+1}) - \underline{W}(t_{n+1})$.

11.4 Fixed and Adaptive Mesh Solutions

In the case of the problem defined by Equation (11.1), comparisons were made between the standard local error control approach in which absolute and relative tolerances RTOL and ATOL are defined (see, [21]), and the new approach defined by (11.13). The choice of the parameter ϵ is an important factor in the performance of the second approach. In selecting this parameter the local growth in the spatial

discretization error should dominate the temporal error and the work needed should not be excessive. Obviously the larger the value of ϵ the fewer ODE time steps there will be, and the smaller the value of ϵ the more steps there will be. A good compromise between efficiency and accuracy is to use ϵ in the range of 0.1 to 0.3. The numerical experiments described by Ahmad [1] confirm the results of Berzins [3], although it is noted that for some combustion problems, ϵ may have to be reduced to below 0.1.

An important feature of solving the problem defined by Equation (11.1) is that the numerical solution may move with an incorrect wave speed. Leveque and Yee [18] showed that the step size and the mesh size should be $O(\frac{1}{\mu})$, to avoid spurious solutions being generated. In order to illustrate these results we have taken $x_d = 0.3$ in Equation (11.1), $\triangle x = 0.02$ and used a fixed time step $\triangle t = 0.015$. The product of time step $\triangle t$ and the reaction rate μ determines the stiffness of the system. Figure 11.1 shows the comparison of the computed solution and exact solution at $t = 0.3$ for $\mu =$ 100, and 1000 ($\triangle t\mu = 1.5$ and 15), respectively. It is evident from Figure 11.1

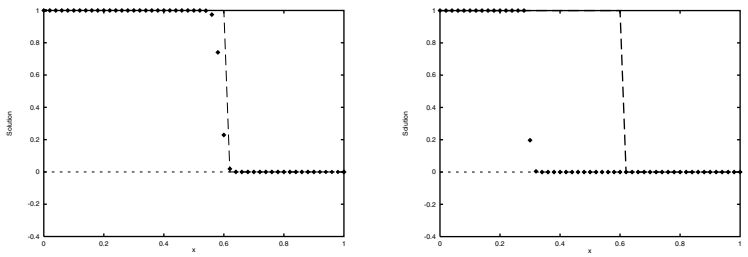


FIGURE 11.1
Comparison between true solution (line) and numerical solution (dots) using local error control with 0.01 relative tolerance and 1×10^{-5} absolute tolerance.

that for smaller $\triangle t\mu$ the strategy works well and good results are obtained. When $\triangle t\mu = 15$, the discontinuity has stopped at x = 0.3 and when a trapezoidal quadrature rule was used for the source term, a large undershoot and overshoot occurred in the numerical solution. Leveque and Yee [18] pointed out that the source of difficulty is the discontinuity in the solution and that a much finer grid is needed there. They suggested deploying a method that is capable of increasing the spatial resolution near the discontinuity rather than excessive refinement of the overall grid.

For this purpose a monitor function was used here to guide the decision as to where to refine or coarsen the mesh. A commonly used monitor function is the second spatial derivative which, however, tends to infinity around a shock [21] as the mesh is refined. In order to overcome this we have introduced a new monitor function based upon the local growth in time spatial error $\underline{est}$ as defined by Equation (11.13). This leads to the use of local grid refinement, and with the help of the error balancing approach described in Section 11.3 it is possible to create a new refined grid directly surrounding the location of the source. For this purpose we have modified the approach described by Pennington and Berzins [21]. The remesh routine bisects the mesh cell if the monitor function is too large or combines two cells into one if the monitor function is

well below the required value. In the experiments here the remeshing routine is called on every second time step. The adaptive mesh initially starts with 26 points and when the error is larger than the specified limit, then the corresponding cell is subdivided into two with 75 points in total being allowed for the case shown in Figure 11.2, which shows the front moving correctly. The conclusion from these experiments is that for problems combining reaction type terms and advection operators, the use of adaptive mesh techniques within an MOL framework may be a critical factor in ensuring that a good numerical solution is obtained. The remainder of this chapter will show that this conclusion also applies to atmospheric modeling problems in two and three space dimensions.

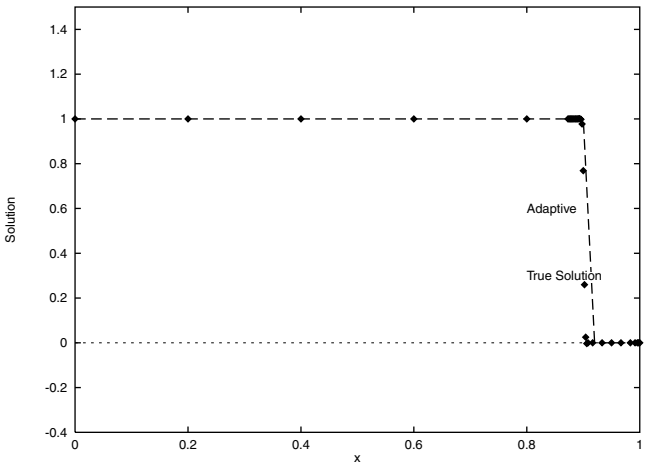


FIGURE 11.2
True solution (lines) vs. adaptive mesh solution (dots), $t = 0.6$**.**

11.5 Atmospheric Modeling Problem

In order to illustrate the application of the MOL to atmospheric modeling problems, the model problem considered here involves the interaction of a power plant plume with background emissions. Such a power plant plume is a highly concentrated source of NOx (NO and NO_2) emissions, which can be carried through the atmosphere for hundreds of kilometers, and so provides a stringent test of whether adaptive gridding methods can lead to more reliable results for complex multi-scale models. The test conducted here involves considering the interaction of the plume with its surroundings, and in the model we look at background scenarios of both clean and polluted air [32]. The test case model covers a region of 300 $\times$ 500 km. To keep the model simple, and therefore reveal particular issues related to the mesh, we have

used a reduced chemical scheme with idealized dispersion conditions. The domain is approximated by an unstructured triangular mesh in two space dimensions and by a tetrahedral mesh in three space dimensions. In both cases the mesh can then be adapted to higher and higher levels of refinement according to errors in solution components. The solution technique is based on the spatial discretization of a set of advection/diffusion equations on the unstructured mesh using a finite volume, flux-limited scheme.

The atmospheric diffusion equation in three space dimensions is given by:

$$\frac{\partial c_s}{\partial t} = -\frac{\partial (uc_s)}{\partial x} - \frac{\partial (wc_s)}{\partial y} - \frac{\partial (vc_s)}{\partial z} + \frac{\partial}{\partial x}\left(K_x \frac{\partial c_s}{\partial x}\right) + \frac{\partial}{\partial y}\left(K_y \frac{\partial c_s}{\partial y}\right) + \frac{\partial}{\partial z}\left(K_z \frac{\partial c_s}{\partial z}\right) + R_s(c_1, c_2, \ldots, c_q) + E_s - (\kappa_{1s} + \kappa_{2s})c_s \ , \qquad (11.17)$$

where c_s is the concentration of the sth compound, u, w, and v are wind velocities, K_x, K_y, and K_z are turbulent diffusivity coefficients, and κ_{1s} and κ_{2s} are dry and wet deposition velocities, respectively. E_s describes the distribution of emission sources for the sth compound and R_s is the chemical reaction term which may contain nonlinear terms in c_s. For *npde* chemical species an *npde*-dimensional set of PDEs is formed describing the rates of change of species concentration over time and space, where each may be coupled through the nonlinear chemical reaction terms.

In the first instance the restriction to two space dimensions has the advantage that it is possible to concentrate on showing that standard adaptive numerical methods have the potential to reveal detail not previously observed in plume models. The extension to three dimensions will then show that the same conclusions can be drawn but that there are additional benefits from using mesh refinement vertically.

The simplified chemical mechanism used is shown in Table 1 of Tomlin et al. [32] and contains only 10 species. Despite its simplicity, it represents the main features of a tropospheric mechanism, namely the competition of the fast equilibrating inorganic reactions:

$$\begin{aligned} \mathrm{NO_2} + \mathrm{h}\nu \quad &\xrightarrow{\mathrm{O_2}} \quad \mathrm{O_3} + \mathrm{NO} \\ \mathrm{NO} + \mathrm{O_3} \quad &\rightarrow \quad \mathrm{NO_2} + \mathrm{O_2}, \end{aligned}$$

with the chemistry of volatile organic compounds (voc's), which occurs on a much slower time-scale. This separation in time-scales generates stiffness in the resulting equations. The voc reactions are represented by reactions of a single species, formaldehyde. This is unrealistic in terms of the actual emissions generated in the environment, but the investigation of fully speciated voc's is not the purpose of the present study. We therefore wished to include the minimum number of reactions which would lead to the generation of ozone at large distances from the NOx source. Deposition processes have not been included in the first instance.

In the work of Tomlin et al. [32], the model was used to represent three separate scenarios of a plume of concentrated NOx emissions being dispersed through a back-

ground of clean and polluted air. Only one set of these results is shown here. This case represents a clean air situation where the background levels for NOx and voc's are low. Initial conditions for background concentrations are NO_2: 1.00×10^8 (molecule cm^{-3}), NO : 1.00×10^8 (molecule cm^{-3}), O_3: 5.00×10^{11} (molecule cm^{-3}), HCHO : 1.00×10^{10} (molecule cm^{-3}). Concentrations in the background change diurnally as the chemical transformations take place according to photolysis rates, temperature, and concentration changes.

The power station was taken to be a separate source of NOx and this source was represented in a slightly different way. In this case, the chimney region is treated as a subdomain and the concentration in the chimney set as an internal boundary condition. In terms of the mesh generation, this ensures that the initial grid will contain more elements close to the concentrated emission source. This is similar in methodology to the telescopic approach. The concentration in the chimney corresponds to an emission rate of NOx of 400 kg hr^{-1}. We have considered only 10% of the NOx to be emitted as NO_2.

A constant wind speed of 5 ms^{-1} in the x-direction was used and the eddy diffusion parameters K_x and K_y were set at 300 m^2s^{-1} for all species.

11.6 Triangular Finite Volume Space Discretization Method

The basis of the numerical method is the spatial discretization of the PDEs in Equation (11.17) on unstructured triangular meshes as used in the software SPRINT2D [7]. The MOL approach then leads to a system of ODEs in time which can then be solved as an initial value problem, and a variety of powerful software tools exist for this purpose [5]. For advection-dominated problems it is important to choose a discretization scheme that preserves the physical range of the solution.

Unstructured triangular meshes are popular with finite volume/element practitioners because of their ability to deal with general two-dimensional geometries. In terms of application to multi-scale atmospheric problems, we are not dealing with complex physical geometries, but unstructured meshes provide a good method of resolving the complex structures formed by the interaction of chemistry and flow in the atmosphere and by the varying types of emission sources. The term unstructured represents the fact that each node in the mesh may be surrounded by any number of triangles, whereas in a structured mesh this number would be fixed. The discretization of advection/diffusion/reaction equations on unstructured meshes will now be discussed.

For systems of equations such as (11.17) it is useful to consider the advective and diffusive fluxes separately in terms of the discretization. In the present work, a flux-limited, cell-centered, finite-volume discretization scheme of Berzins and Ware [6, 4] was chosen. This method enables accurate solutions to be determined for both smooth and discontinuous flows by making use of the local Riemann solver flux techniques

(originally developed for the Euler equations) for the advective parts of the fluxes, and centered schemes for the diffusive part. The scheme used for the treatment of the advective terms is an extension to irregular triangular meshes of the nonlinear scheme described by Spekreijse [29] for regular Cartesian meshes. The scheme of Berzins and Ware has the desirable properties (see Chock [11]) of preserving positivity, eliminating spurious oscillations, and restricting the amount of diffusion by the use of a nonlinear limiter function. Recent surveys of methods for the advection equation [34, 36] have suggested the use of a very similar scheme to Spekreijse for regular Cartesian meshes, preferring it to schemes such as flux-corrected transport.

To illustrate this method, consider the advection-reaction equation that extends Equation (11.1) to two space dimensions:

$$\frac{\partial c}{\partial t} = -\frac{\partial uc}{\partial x} - \frac{\partial wc}{\partial y} + R(c)\,, t \in (0, t_e), (x, y) \in \Omega \tag{11.18}$$

with appropriate boundary and initial conditions. A finite volume type approach is adopted in which the solution value at the centroid of triangle i, (x_i, y_i), is c_i and the solutions at the centroids of the triangles surrounding triangle i are c_l, c_j, and c_k. Integration of Equation (11.18) on the ith triangle, which has area A_i, use of the divergence theorem, and the evaluation of the line integral along each edge by the midpoint quadrature rule gives an ODE in time:

$$\begin{aligned}\frac{dc_i}{dt} &= -\frac{1}{A_i}\Big(uc_{ik}\Delta y_{0,1} - vc_{ik}\Delta x_{0,1} + uc_{ij}\Delta y_{1,2} \\ &\quad - vc_{ij}\Delta x_{1,2} + uc_{il}\Delta y_{2,0} - vc_{il}\Delta x_{2,0}\Big) + R(c_i)\,,\end{aligned} \tag{11.19}$$

where $\Delta x_{ij} = x_j - x_i$, $\Delta y_{ij} = y_j - y_i$. The fluxes uc_{ij} and vc_{ij} in the x and y directions, respectively, are evaluated at the midpoint of the triangle edge separating the triangles associated with c_i and c_j. These fluxes are evaluated by taking account of the flow directions with respect to the orientation of the triangle. This is achieved by using either the *left* or *right* solution values depending on the direction of advection and how each edge is aligned. These *left* and *right* solution values for each edge in a triangle are defined as the *left* solution value being that internal to the ith triangle, and the *right* solution value being that external to triangle i. Consider, for example, the case shown in Figure 11.3 when u is positive and $x_i < x_j$. This means that the x component of the advection is flowing from node i to node j, and so $c_{ij} = c^l_{ij}$. Similarly when v is positive the y component of the wind is blowing from node k to node i and so $c_{ik} = c^r_{ik}$. Hence, Equation (11.19) may be written as

$$\begin{aligned}\frac{dc_i}{dt} &= -\frac{1}{A_i}\Big(uc^l_{ik}\Delta y_{0,1} - vc^r_{ik}\Delta x_{0,1} + uc^l_{ij}\Delta y_{1,2} \\ &\quad - vc^l_{ij}\Delta x_{1,2} + uc^r_{il}\Delta y_{2,0} - vc^l_{il}\Delta x_{2,0}\Big) + R(c_i)\,.\end{aligned} \tag{11.20}$$

A simple first-order scheme uses $c^l_{ij} = c_i$, $c^r_{ij} = c_j$ on the edge between triangles i and j. This scheme is too diffusive and so Berzins and Ware [6] use a complex

interpolation scheme to obtain the *left* and *right* values on each edge. The interpolants in this second order scheme use a constrained or limited form of the solution obtained from the six triangles surrounding an edge giving a 10-triangle stencil for the discretization of the convective terms on each triangle. For example, the value

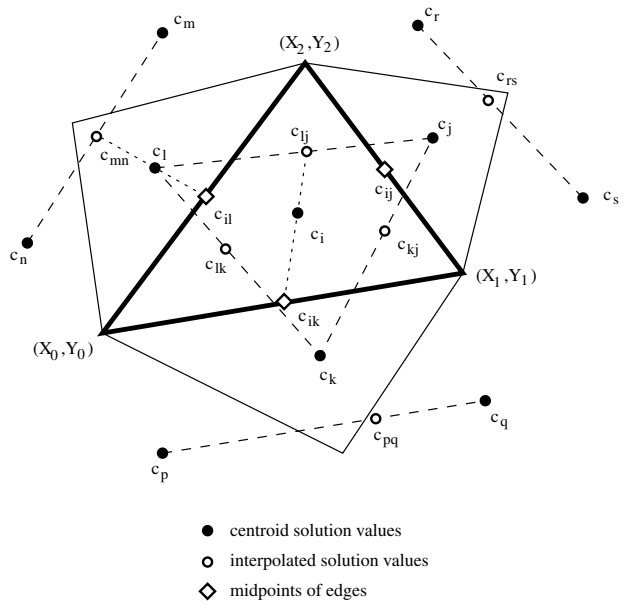


FIGURE 11.3
Interpolants used in irregular mesh flux calculation.

c_{ij}^{l} is constructed by forming a linear interpolant using the solution values c_i, c_k, and c_l at the three centroids. An alternative interpretation is that linear extrapolation is being used based on the solution value c_i and an intermediate solution value (itself calculated by linear interpolation) c_{lk} which lies on the line joining the centroids at which c_l and c_k are defined (see Figure 11.3), i.e.,

$$c_{ij}^{l} = c_i + \Phi(S_{ij})\, d_{ij,i} \frac{c_i - c_{lk}}{d_{i,lk}} \,, \tag{11.21}$$

where the argument S is a ratio of solution gradients defined in a way similar to the ratio r_j in Equation (11.5), see [6], and the generic term $d_{a,b}$ denotes the positive distance between points a and b. For example $d_{ij,i}$ denotes the positive distance between points ij and i, see Figure 11.3, as defined by

$$d_{i,ij} = \sqrt{(x_i - x_{ij})^2 + (y_i - y_{ij})^2} \,, \tag{11.22}$$

where (x_{ij}, y_{ij}) are the coordinates of c_{ij}. In order to preserve positivity in the numerical solution, the limiter function Φ is used and has to satisfy $\Phi(S)/S \leq 1$,

see [6]. These conditions are satisfied, for example, by a modified van Leer limiter defined by:

$$\Phi(S) = (S + |S|)/(1 + \text{Max}(1, |S|)) . \tag{11.23}$$

The value c_{ij}^r is defined in a way similar to using the centroid values c_j, c_s, and c_r. This scheme is of second-order accuracy, see [6]. The diffusion terms are discretized using a finite-volume approach to reduce the integrals of second derivatives to the evaluation of first derivatives at the midpoints of edges. These first derivatives are then evaluated by differentiating a bilinear interpolant based on four midpoint values, see [7]. The boundary conditions are implemented by including them in the definitions of the advective and diffusive fluxes at the boundary.

11.7 Time Integration

An MOL approach with the above spatial discretization scheme results in a system of ODEs in time which are integrated using the code SPRINT [5] with the Theta or BDF options which are specially designed for the solution of stiff systems with moderate accuracy and automatic control of the local error in time. Once the PDEs have been discretized in space we are left with a large system of coupled ODEs of dimension $N = m \times npde$ where m is the number of triangles in the mesh, and $npde$ the number of species. These equations may now be written in the same form as Equation (11.2) as

$$\dot{\underline{c}} = \underline{F}_N(\, t,\ \underline{c}(t)\,), \underline{c}(0) \ \text{ given} , \tag{11.24}$$

where, in the case of a single species, the vector, $\underline{c}(t)$, is defined by $\underline{c}(t) = [c(x_1, y_1, t), \ldots, c(x_N, y_N, t)]^T$. The point x_i, y_i is the center of the ith cell and $C_i(t)$ is defined as a numerical approximation to the exact solution to the PDE evaluated at the centroid, i.e., $c(x_i, y_i, t)$. The MOL approach is used to numerically integrate Equation (11.24) thus generating an approximation, $\underline{C}(t)$, to the vector of exact PDE solution values at the mesh points, $\underline{c}(t)$.

The Theta method [6], which has been used for the experiments described here, defines the numerical solution at $t_{n+1} = t_n + \triangle t$, where $\triangle t$ is the time-step size, as denoted by $\underline{C}(t_{n+1})$, by:

$$\underline{C}(t_{n+1}) = \underline{C}(t_n) + (1 - \theta)\triangle t\ \dot{\underline{C}}(t_n) + \theta\ \triangle t\ \underline{F}_N(t_{n+1}, \underline{C}(t_{n+1})) , \tag{11.25}$$

in which $\underline{C}(t_n)$ and $\dot{\underline{C}}(t_n)$ are the numerical solution and its time derivative at the previous time t_n and $\theta = 0.55$. This system of equations is solved by using the approach described in Section 11.2. In this case, the matrix J_s is block-diagonal with as many blocks as there are triangles and with each block having as many rows and columns as there are PDEs. The fact that the blocks relate only to the chemistry

plus source/sink terms within each cell, means that the equations may be solved independently using LU decomposition, or even more efficiently by using Gauss–Seidel iterations, see [35]. This approach may also be interpreted as approximating the flow term $[I - \triangle t\theta J_f]$ by the identity matrix, as is done when using functional iteration with the Theta method applied to flow alone [3]. Since the spatial discretization method connects each triangle to as many as 10 others, it follows that the matrix $\left[I - \triangle t\theta J_f\right]$ may have a much more complex sparsity pattern than that of the block-diagonal matrix $[I - \triangle t\theta J_s]$. Approximating the matrix $\left[I - \triangle t\theta J_f\right]$ by the identity matrix [6] thus eliminates a large number of the full Jacobian entries. Moreover, the use of Gauss–Seidel iteration makes it possible to solve these problems without any matrices being stored. This approach is particularly useful in three space dimensional problems.

The original approach of Berzins [3] was only extended to source-term problems by Ahmad and Berzins [1]. As a consequence the calculations performed by Tomlin et al. [32] used the standard local error approach given by:

$$|| \ \underline{le}(t_{n+1}) \ || \ < \ TOL \ , \tag{11.26}$$

where $\underline{le}$ is the local error defined as in Equations (11.12) and (11.13).

11.8 Mesh Generation and Adaptivity

The initial unstructured meshes used in SPRINT2D are created from a geometry description using the Geompack [16] mesh generator. These meshes are then refined and coarsened by the Triad adaptivity module, which uses data structures to enable efficient mesh adaptation.

Since the initial mesh is unstructured we have to be very careful in choosing a data structure that provides the necessary information for refining and derefining the mesh. When using a structured mesh it is possible to number mesh vertices or elements explicitly. This is not possible for unstructured meshes and therefore the data structure must provide the necessary connectivity. The important factor is to maintain the quality of the triangle as the mesh is refined and coarsened. This is achieved using a tree-like data structure with a method of refinement based on the regular subdivision of triangles. These may later be coalesced into the parent triangle when coarsening the mesh. This process is called local h-refinement, since the nodes of the original mesh do not move and we are simply subdividing the original elements. Three examples of adaptive meshes for a single moving front at different times are shown in Figure 11.4. These meshes show how the adaptive mesh follows the front as it moves in time across the spatial domain. Similar procedures are used extensively with a wide range of both finite element and volume methods for a very broad range of physical problems. Once a method of refinement and derefinement has been implemented, it remains to decide on a suitable criterion for the application

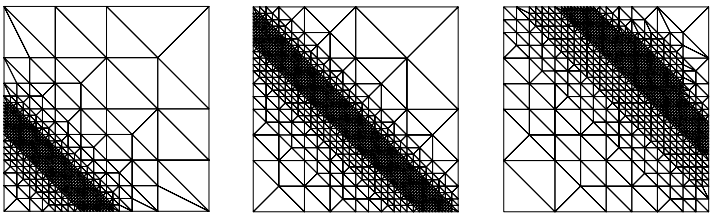


FIGURE 11.4
Sequence of refined meshes.

of the adaptivity. The ideal situation would be that the decision to refine or derefine would be made on a fully automatic basis with no user input necessary. In practice a combination of an automatic technique and some knowledge of the physical properties of the system is used. The technique used in this work is based on the calculation of spatial error estimates. Low- and high-order solutions are obtained and the difference between them gives the spatial error, as in Section 11.3 and in [3] but without the extension to source terms in [1]. The algorithm can then choose to refine in regions of high spatial error by comparison with a user defined tolerance. For the ith PDE component on the jth triangle, a local error estimate $e_{i,j}(t)$ is calculated from the difference between the solution using a first-order method and that using a second-order method. For time-dependent PDEs this estimate shows how the spatial error grows locally over a time step. A refinement indicator for the jth triangle is defined by an average scaled error ($serr_j$) measurement over all $npde$ PDEs using supplied absolute and relative tolerances:

$$serr_j = \sum_{i=1}^{npde} \frac{e_{i,j}(t)}{\text{atol}_i / A_j + \text{rtol}_i \times C_{i,j}} , \tag{11.27}$$

where atol and rtol are the absolute and relative error tolerances. This formulation for the scaled error provides a flexible way to weight the refinement towards any PDE error. An integer refinement level indicator is calculated from this scaled error to give the number of times the triangle should be refined or derefined. Since the error estimate is applied at the end of a time step, it is too late to make the refinement decision. Methods are therefore used for the prediction of the growth of the spatial error using linear or quadratic interpolants. The decision about whether to refine a triangle is based on these predictions, and the estimate made at the end of the time step can be used to predict errors at future time steps. Generally it is found that large spatial errors coincide with regions of steep spatial gradients. The spatial error estimate can also be used to indicate when the solution is being solved too accurately and can indicate which regions can be coarsened.

For applications such as atmospheric modeling it is important that a maximum level of refinement can be set to prevent the code from adapting to too high a level in regions with concentrated emissions. This is especially important around point or highly concentrated area sources. Here, because of the nature of the source, steep

spatial gradients are likely to persist down to very high levels of refinement. This would have the consequence that the number of elements on which the PDEs had to be discretized would become prohibitively large. For the following test problems the maximum level of refinement was therefore limited to level 3.

11.9 Single-Source Pollution Plume Example

The example used here to illustrate the effectiveness of the adaptive mesh is that of a single plume pollution source. In this case the initial two-dimensional mesh was generated with only 100 elements. It is difficult to relate the size of unstructured meshes directly to regular rectangular ones, but our original mesh was comparable to the size of mesh generally used in regional scale atmospheric models, the largest grid cell being approximately 60 km along its longest edge. Close to the chimney the mesh was refined to elements of length 5 km ensuring that it would be refined to a reasonable resolution in this region of steep gradients. If we allow the mesh to refine two levels, then the smallest possible mesh size close to the chimney will be 1.25 km in length. Spatial errors in the concentration of NO were chosen as the criterion from which to further refine the mesh. Test runs showed that regions of high spatial error coincided with steep spatial gradients. The mesh can therefore be considered to adapt around steep NO concentration gradients. Each run was carried out over a period of 48 h starting from midnight on Day 1, so that the diurnal variations could be observed. We present here only a selection of the results that illustrate the main features relating to the mesh adaptation.

Figures 11.5 and 11.6 allow a comparison to be made between the structure of

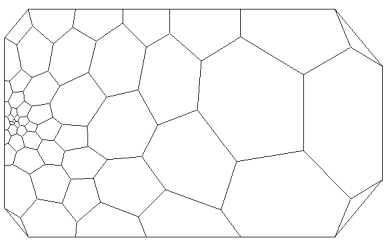


FIGURE 11.5
The structure of the level 0 mesh. The length of the domain is 300 km and the width 200 km. The smallest and largest mesh lengths are approximately 5 and 60 km, respectively, for the level zero domain.

the base mesh and a mesh that has been adapted up to level 2 at 14.00 on Day 2. In these figures the sides of the polygons represent the distance between cell centers on the triangular mesh. The main area of mesh refinement is along the plume edges close to the chimney, indicating that there is a high level of structure in these regions.

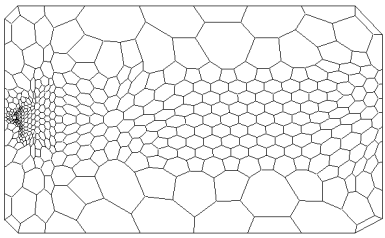


FIGURE 11.6
The structure of the level 2 adaptive mesh.

On the coarse mesh the plume is dispersed over a much larger area than on the fine mesh and most of the plume structure is lost. Close to the stack the concentration of O_3 is much lower than that in the background because of high NOx concentrations. The inorganic chemistry is dominant in this region and the ozone is consumed by the reaction: $NO + O_3 \rightarrow NO_2 + O_2$.

In Figure 11.7 we present a cross-plume profile of the NO_2 concentrations at a

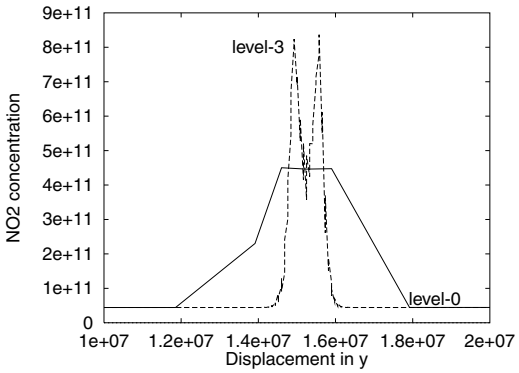


FIGURE 11.7
Cross plume NO_2 profiles 10 km from stack in molecules cm^{-3}, showing how the level 3 solution captures the structure of the plume.

distance of 10 km downwind of the chimney stack for Case A at the same time as the previous figure. The figure clearly shows the features at the edge of the plume which are revealed by the adaptive solution. From the base mesh, where the distance between elements along the y-axis close to the stack is 20 km, it appears that the concentration of NO_2 rises to a peak in the center of the plume. If the mesh is refined to higher levels, then we start to see the true structure of the plume emerging. With a level 3 solution we can see that the peak concentrations are actually found along the edges of the plume and that the concentration of NO_2 drops to very low levels at the plume center. From the area under these curves it is found that there is a 30% difference between the overall level 0 and the level 3 concentrations. This shows

that not only the peak concentrations, but the total integrated concentrations are very different for the different levels of mesh adaptation. It is clear therefore that using a very coarse grid in regions of steep spatial gradients can lead to an over-estimate of total pollutant concentrations for systems with nonlinear chemical schemes.

Figures 11.8 and 11.9 show that in the case considered here the plume is over-

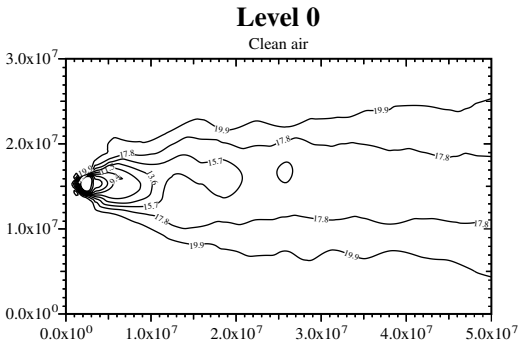


FIGURE 11.8
Ozone contours for Case C, clean air, level 0 calculation.

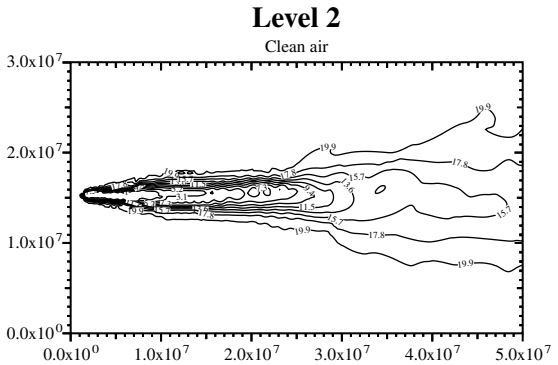


FIGURE 11.9
Ozone contours for Case C, clean air, level 2 calculations.

dispersed in the level 0 case and the spatial distribution of ozone is therefore inaccurately represented. For the clean air case, the levels of ozone drop considerably in the plume compared to the background since the levels of NO are much higher there. For the level 0 case these lowered concentrations spread over very large distances owing to the over-dispersion of the plume. The location of reduced/raised concentrations will therefore be incorrect for the level 0 results in all three cases. For each scenario, the level 0 solution leads to a smoothing out of the ozone profiles so that the true structure caused by the interaction of the plume with background air is missed.

The striking result is that the adaptive solution reveals features such as peak levels of NO_2 and O_3 which could not be detected using a coarse mesh. The change in mesh refinement also resulted in a change in overall or integrated concentration levels. This indicates that due to strongly nonlinear terms in the chemical reaction rates, the source terms in the PDE will be mesh dependent. Without using a fine mesh over the whole domain so that the concentrations in neighboring cells differ only very little, the effects of this nonlinearity could be quite significant. To reduce the effects it is important to refine the mesh at least in regions of steep spatial gradients. This has been partially addressed by the telescopic methods presently used in air quality models. However, the present test case has shown that steep gradients can occur at long distances downwind from the source, for example the change in ozone concentrations along the edges of the plume. Adaptive algorithms seem to present a successful method of achieving accuracy in such regions and can do so in an automatic way. The main limitation of the above approach is that only two space dimensions have been considered. The next issue to be resolved is whether mesh adaptation is necessary in the vertical direction and how appropriate an MOL approach is in three space dimensions. These are the issues considered in the next five sections.

11.10 Three Space Dimensional Computations

The standard approach with three space dimensional atmospheric dispersion problems is that in the vertical domain usually a stretched mesh is used, placing more solution points close to the ground. As in the horizontal domain, the resolution of the mesh in the vertical direction affects the vertical mixing of pollutant species. The use of adaptive meshes in the vertical domain has thus far received little attention.

In the work described here we have used two approaches for solving three space dimensional atmospheric dispersion problems. Both approaches use a fully 3D unstructured mesh based on tetrahedral elements. The first approach is described in [17] and is the closer of the two approaches to the two-dimensional case described in Section 11.6 in that a cell-centered, finite-volume scheme is used for the spatial discretization. In this case a conventional MOL approach is used based on a modified version of the SPRINT time-integration package. The linear algebra approach of Section 11.7 is used with a simple first-order spatial discretization approach. The disadvantage of this approach is that it requires a much larger number of unknowns for a given mesh than if a cell-vertex approach is used with the solution unknowns being positioned at the nodes of the mesh. The price that is paid for this reduction in the number of unknowns is an increase in the complexity of the discretization method. There is also the well-known difficulty that the cell-vertex discretization may not preserve the positivity of the solution on certain meshes due to the discretization of the diffusion operator [9]. Although it may be possible to address this issue within an

MOL framework, the need to preserve positivity and the different time-scales needed for advection and chemistry have led us to employ an operator-splitting approach.

The next section describes the 3D unstructured mesh discretization method and the flow-integration scheme which advances the solution in time. Section 11.12 contains the mesh-adaptation strategy which changes the connectivity in the data structure of the mesh in response to changes in the solutions. Section 11.13 explains the implicit-explicit method used to solve the transport equation. Section 11.14 contains the test examples which have been designed to determine the importance of mesh structure on both horizontal and vertical mixing for typical meteorological conditions. The test problem describes the dispersion of pollutants from a single source due to typical boundary-layer wind profiles. Finally, we draw conclusions in Section 11.15 regarding the importance of adaptive-mesh method in solving 3D atmospheric-flow problems.

11.11 Three Space Dimensional Discretization

The atmospheric-diffusion equation is discretized over special volumes that form the dual mesh. The dual mesh is formed by constructing non-overlapping volumes, referred to as dual cells, around each node. The dual mesh for a tetrahedral grid is constructed by dividing each tetrahedron into four hexehedra of equal volumes, by connecting the mid-edge points, face-centroids, and the centroid of the tetrahedron. The control volume around a node 0 is thus formed by a polyhedral hull which is the union of all such hexahedra that share that node. The quadrilateral faces that constitute the dual mesh may not all be planer. Each component of the diffusion Equations (11.17) is discretized using the same method. Hence, for simplicity, instead of treating the vector $\underline{c}$, we choose one of its components, say c, and describe its discretization.

11.11.1 Flux Evaluation Using Edge-Based Operation

The evaluation of flux around a dual cell can be cast in an edge-based operation. Let us discretize the divergence term

$$\frac{\partial f}{\partial x} + \frac{\partial g}{\partial y} + \frac{\partial h}{\partial z}$$

over the control volume Ω_0 enclosing the node 0. This divergence form is converted to flux form using Gauss divergence theorem:

$$\begin{aligned} \int_{\Omega_0} \left(\frac{\partial f}{\partial x} + \frac{\partial g}{\partial y} + \frac{\partial h}{\partial z} \right) d\Omega &= \int_{\partial\Omega_0} (f\, n_x + g\, n_y + h\, n_z)\, dS \\ &= \sum_k \left(f\, S_x + g\, S_y + h\, S_z \right) , \end{aligned} \tag{11.28}$$

where the summation is over all the dual mesh faces that form the boundary of the control volume around the node 0 and the areas S_x, S_y, S_z are projections of the dual quadrilateral face.

Consider edge i, formed by nodes 0 and N(i). The quadrilateral faces of the dual mesh that are connected to the edge at its mid-point P are shown in Figure 11.10. The number of such quadrilateral faces attached to an edge depends on the number

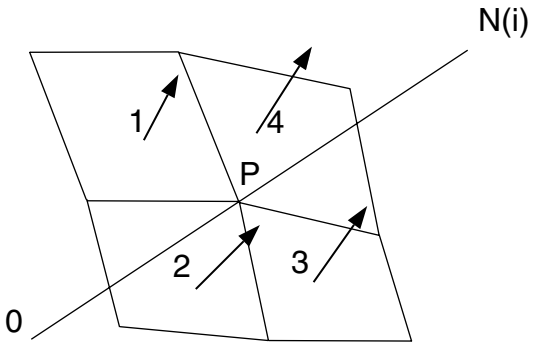


FIGURE 11.10
Dual mesh faces attached to an edge.

of tetrahedra neighbors to that edge. There are four tetrahedra sharing the edge i in Figure 11.10. The projected area, A_i, associated with the edge i is calculated in terms of the quadrilateral face areas, a_1, a_2, a_3, a_4, as

$$(A_i)_x = \sum_{j=1}^{4} (a_j)_x, \quad (A_i)_y = \sum_{j=1}^{4} (a_j)_y, \quad (A_i)_z = \sum_{j=1}^{4} (a_j)_z \, . \tag{11.29}$$

The projections are computed so that the area vector points outward from the control volume surface associated with a node. The boundary of the control volume around the node 0 is formed by the union of all such areas A_i associated with each edge i that shares the node 0. The contribution of the edge i to the fluxes across the faces of the control volume surrounding the node 0 is given by

$$f_p \, (A_i)_x + g_p \, (A_i)_y + h_p \, (A_i)_z \, .$$

Hence, Equation (11.28) is replaced by

$$\int_{\Omega_0} \left(\frac{\partial f}{\partial x} + \frac{\partial g}{\partial y} + \frac{\partial h}{\partial z} \right) d\Omega = \sum_i \left(f_p \, (A_i)_x + g_p \, (A_i)_y + h_p \, (A_i)_z \right) \, , \tag{11.30}$$

where the sum is over the edges that share the node 0. The fluxes are thus calculated on an edge-wise basis and conservation is enforced by producing a positive flux contribution to one node and an equally opposite contribution to the other node that forms the edge.

11.11.2 Adjustments of Wind Field

In an atmospheric pollution model, we often use observed wind data which are not mass conservative. Even mass-conserving wind data might not be mass conservative in the numerical sense when interpolated onto an unstructured grid. Thus, we want to adjust the wind data in such a way that the observed data are minimally changed while still satisfying the mass-conservative property numerically. If u, v, w are the wind velocities, then they must satisfy

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0 \,. \tag{11.31}$$

Here we enforce mass conservation using the variational calculus technique of Mathur and Peters [20]. The technique attempts to adjust the wind velocity in a manner such that the interpolated data are minimally changed in a least-squares sense, and at the same time, the adjusted values satisfy the mass-conservation constraint. The details are provided by Ghorai et al. [12].

We have adjusted one-dimensional stable, neutral, and unstable boundary layer wind velocities which are a function of z. The wind velocity is mass conservative analytically. The wind velocity remains mass conservative in the numerical sense in the base mesh since the unstructured base mesh is regular, but may not be numerically mass conservative once the grid is refined (derefined). A representative one-dimensional neutral boundary layer velocity is shown in Figure 11.11(b). The velocity field is adjusted in the refinement region, but away from the refinement region, the velocity remains almost unchanged.

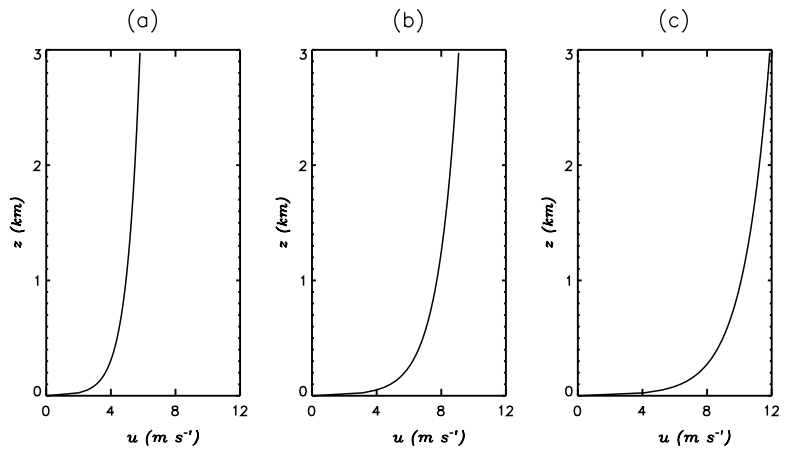


FIGURE 11.11
A representative variation of wind with height for (a) stable, (b) neutral, and (c) unstable boundary layers.

The base grid spacings along the vertical increases upwards. Thus, the grid quality near the ground is worse due to the large aspect ratio of the tetrahedron. The velocity

corrections decrease upwards as the refine region moves upwards. Suppose we have a refine region at 150 m height. The maximum corrections are 12, 14, and 0.06 cm s^{-1}, respectively, for the u, v, and w components. For a refine region at 600 m height the corresponding components are 11, 11, and 0.03 cm s^{-1}. And finally, the corresponding corrections decreases to 0.3, 0.2, and 0.0002 cm s^{-1} at 1.8 km height. The neutral boundary layer velocity increases from 0 to 9 m s^{-1} as z increases from 0 to 3 km and so the velocity corrections are small.

11.11.3 Advection Scheme

The discretization of the term

$$\int_{\Omega_0} \left[\frac{\partial(u\,c)}{\partial x} + \frac{\partial(v\,c)}{\partial y} + \frac{\partial(w\,c)}{\partial z} \right] d\Omega \equiv \int_{\Omega_0} \nabla . F \, d\Omega \, ,$$

where

$$F = \left(u\hat{i} + v\hat{j} + w\hat{k} \right) c \, ,$$

is done by using an algorithm based on that of Barth and Jespersen [8] and uses Equation (11.30) to rewrite the equation above as an edge-based computation:

$$\sum_i \left[u_p \, (A_i)_x + v_p \, (A_i)_y + w_p \, (A_i)_z \right] (c)_p = \sum_i \left(\underline{F}_p . \underline{A}_i \right) (c)_p \, , \qquad (11.32)$$

where $\underline{A_i}$ is called the edge-normal associated with the edge i and the sum is over all the edges sharing the node 0 with control volume Ω_0. Evaluation of this expression is by using the upwind limited approach of Barth and Jesperson [8]. The values of limiter functions and gradient at the nodes are not calculated on a node-by-node basis (which is CPU intensive), instead they are calculated in an edge-based operation [8, 9]. The time step for the advection scheme is chosen so that it satisfies the CFL condition [37]. The minimum of the time steps over all the vertices constitutes the time step for the advection scheme. Again this computation can be cast into an edge-based operation.

11.11.4 Diffusion Scheme

The diffusion term is discretized using the standard linear finite-element method or the equivalent cell-vertex method described by Barth. Again the key feature is that the calculation of the diffusion terms is reordered so that it involves edge gradient terms, see [9]. The disadvantage of the standard approach is that it does not preserve positivity of the solution for certain meshes, see [9]. Very recent work has provided methods that begin to address this issue [23].

11.12 Mesh Adaptation

The cell-vertex scheme approach is hierarchical in nature [10, 28], and is applicable to meshes constructed from tetrahedral shaped elements. The basic mesh objects of *nodes*, *edges*, *faces,* and *elements,* that together form the computational domain, map onto the data objects within the adaptation algorithm tree data structure. The data objects contain all flow and connectivity information sufficient to adapt the mesh structure and flow solution by either local *refinement* or *derefinement* procedures. The mesh-adaptation strategy assumes that there exists a "good quality" initial unstructured mesh covering the computational domain. The refinement process adds nodes to this base level mesh by edge, face, and element subdivision, with each change to the mesh being tracked within the code data structure by the construction of a data hierarchy. The derefinement is the inverse of refinement, where nodes, faces, and elements are removed from the mesh by working back through the local mesh refinement hierarchy.

The main adaptation is driven by refining and derefining element edges. Thus, if an edge is refined by the addition of a node along its length, then all the elements that share the (parent) edge under refinement must be refined. In the case of derefinement all the elements that share the node being removed must be derefined. Numerical criteria derived from the flow field will mark an edge to either refine, derefine, or remain unchanged. It is necessary to make sure the edges targeted for refinement and derefinement pass various conditions prior to their adaptation. These conditions effectively decouple the regions of mesh refinement from those of derefinement, meaning that, for example, an element is not both derefined and refined in the same adaptation step.

For reasons of both tetrahedral quality control and algorithm simplicity only two types of element subdivisions are used [28]. The first type of subdivision is called *regular subdivision,* where a new node bisects each edge of the parent element resulting in eight new elements. The second type of dissection, *green subdivision,* introduces an extra node into parent tetrahedron, which is subsequently connected to all the parent vertices and any additional nodes that bisect the parent edges. The green refinement inconsistently removes connected or "hanging" nodes without the introduction of additional edge refinement. The green elements may be of poorer quality in terms of aspect ratio and so the green element may not be further refined. Figure 11.12 demonstrates regular and green refinement for a tetrahedron. The five possible refinement possibilities (if all the edges are refined then the parent element is regularly refined) give rise to between 6 and 14 child green elements.

The choice of adaptation criteria is very important since it can produce either a large or small number of nodes depending on the condition used to flag an edge for the adaptation. Also, when there are a large number of species, the choice of a given criteria might result in high resolution for some species but low resolution for the other species. Let 0 and i be the nodes for a given edge $e(0, i)$. We calculate $tolg$

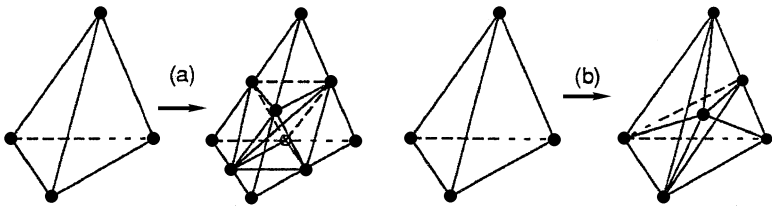


FIGURE 11.12
(a) Regular refinement based on the subdivision of tetrahedron by dissection of interior diagonal (1:8) and (b) "green" refinement by addition of an interior node (1:6).

and $tolc$ by

$$tolg = \frac{|(c)_0 - (c)_i|}{\text{dist}} \quad \text{and} \quad tolc = \frac{(c)_0 + (c)_i}{2},$$

where dist is the length of the edge $e(0, i)$. We refine the edge $e(0, i)$ if $tolg$ and $tolc$ exceed some tolerances, otherwise it is derefined. Also a maximum level of refinement is specified at the beginning so that if an edge is targeted for refinement but it is in the maximum level, then it is kept unchanged.

Suppose we have two edges with $tolg = 100$ and 200. If we take the tolerance parameter, T_g say, for $tolg$ equal to 150, then only the second edge is refined to maximum level. On the other hand, if $T_g = 50$, then both edges are refined to maximum level. We expect that the solution error for edge with $tolg = 200$ is greater than the error in the edge with $tolg = 100$. It might be advantageous to use two sets of $T_g = 50$ and 150. If $tolg > 150$, than we refine an edge to maximum level and if $50 < tolg < 150$, then we refine an edge to the level just lower than the maximum levels. Thus, the idea is to refine to the maximum level in the steepest gradient regions but to lower levels in the regions of less steep gradients.

11.13 Time Integration for 3D Problems

Although in two space dimensional calculations we have used sophisticated space-time error control techniques [7, 32], the need to preserve positivity, to reduce computational cost, and to take into account the different time-scales needed for the integration of advection and chemistry has led us to use an operator-splitting technique. In this approach, the chemistry is decoupled from the transport. The main reason for the use of this is that it is much easier to ensure positivity of the solution components. The nonlinear chemistry part gives rise to stiff ordinary differential equations. We solve the chemistry part using the SPRINT time integration methods [7] and have also used the Gauss–Seidel iteration of Verwer [35]. The transport step is considered first. If $\underline{c}^n$ denotes the species concentration at time level t_n, then the species concentration

at the next time step is given by

$$\underline{c}^{n+1} = \underline{c}^{n} + \triangle t\, \underline{g}(c) + \triangle t\, \underline{f}(c) + \underline{S}\,, \tag{11.33}$$

where $\triangle t$ is the time step, $\underline{g}(c)$ is the advection operator, and $\underline{f}(c)$ is the diffusion operator. In a fully explicit scheme, $\underline{f}$ and $\underline{g}$ are evaluated using values at the time level n. However, the time restriction for stability due to vertical diffusion is severe since the grid spacings along the vertical can be small. Hence, we use an implicit-explicit formulation for Equation (11.33), where the advection is evaluated explicitly and the diffusion is calculated implicitly. Again let us consider node i and let N(i) be the set of nodes sharing the node i. The discretized form of the advection-diffusion equation for c at node i is given by

$$\left(\frac{1}{\triangle t} + a_i\right)(\underline{c}^{n+1})_i = \sum_{j \in \mathrm{N(i)},\, j \neq i} a_j \left(\underline{c}^{n+1}\right)_j + Q_i^n\,, \tag{11.34}$$

where i is varied over all the nodes and

$$Q_i = \left[\frac{\underline{c}^{n}}{\triangle t} + \underline{g}(\underline{c}^{n}) + \underline{S}\right]_i\,.$$

The time step $\triangle t$ is chosen to be equal to the time step due to advection only. The value of time step mainly depends on the wind speed and the vertical mesh spacings near the source. For the base mesh (described in the next section) used in the test examples, $\triangle t$ is $\approx$ 35 s for the stable atmospheric boundary layer but decreases to $\approx$ 18 s for the unstable atmospheric boundary layer. Thus, the time step is smaller for higher wind speed and vice versa. The system of equations given by Equation (11.34) is solved using the Gauss–Seidel iteration technique with over-relaxation and the iteration is stopped when the relative error is less than some prescribed tolerance. The advantage of this method is its computational efficiency. The disadvantage is that we are introducing an extra time integration and splitting error which is not easily quantified. In future work we will revisit this issue of a standard method of lines approach vs. the operator-splitting approach used here.

11.14 Three-Dimensional Test Examples

The advection scheme has been tested by advecting a puff of NO around a horizontal circle without any diffusion [33]. The results showed that the peak almost remains constant suggesting that very little artificial diffusion has taken place for refined meshes. Here we consider the solution of the combined advection-diffusion problem with a source term that relates to the long-range transport of a passive species from an elevated point source.

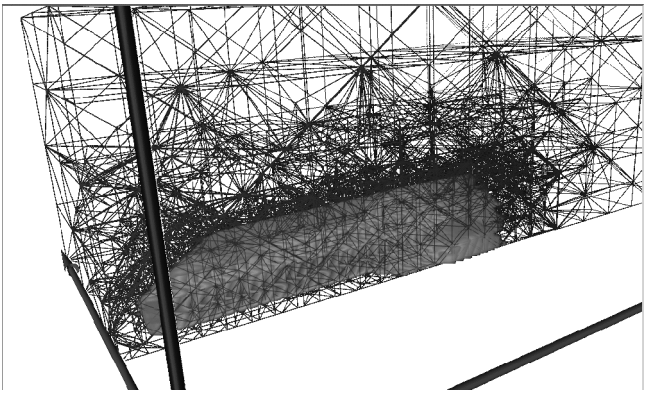


FIGURE 11.13
A representative mesh for the 3D atmospheric dispersion problem.

The background concentration of NO is 7.5×10^{10} molecules/cm^3. The horizontal dimensions of the domain are 96 km and 48 km along the x and y axis, respectively. The vertical height of the domain is 3 km. We consider a point source at (6, 24, 0.24) km location with an NO emission rate of 1.98×10^{24} molecules s^{-1}. For simplicity, we consider a constant wind direction along the x-axis. We consider three different wind velocity and vertical diffusion profiles which are representative of stable, neutral, and unstable boundary layers. The corresponding velocities and vertical diffusions are shown in Figures 11.11 and 11.14 from Seinfeld [25].

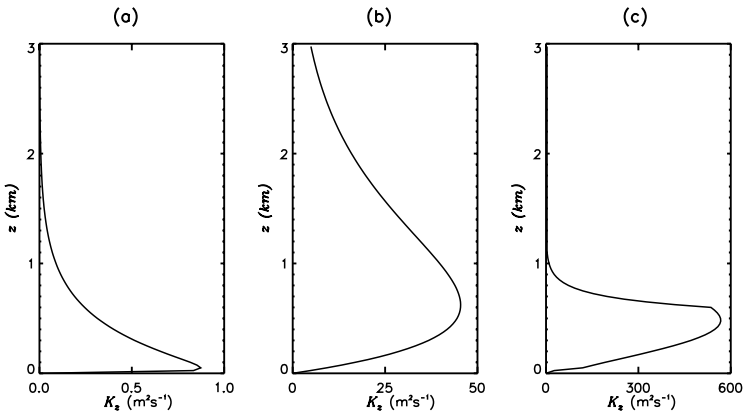


FIGURE 11.14
A representative variation of vertical diffusion with height for (a) stable, (b) neutral, and (c) unstable boundary layers.

The horizontal diffusion coefficients K_x and K_y are kept constant and equal to 50 m^2, s^{-1}. The initial tetrahedral mesh is generated by dividing the whole region

into cuboids and then subdividing a cuboid into 6 tetrahedral elements. The cuboids are 4 km and 4 km along the x and y axis, respectively. The vertical height is divided into nine layers and the layers are placed at 0, 0.206, 0.460, 0.767, 1.13, 1.54, 2.0, 2.45, and 3 km heights, respectively.

We compute the solutions on the adaptive grid and also check the accuracy against a reference solution. The reference solution is obtained on a fixed grid generated from the base mesh by refining all the edges (to level 3) which lie inside a box lying along the x-axis through the source. We also compute the solution on a telescopic grid with refinement around the source and compare the solution with the adaptive and reference solution. The vertical turbulent diffusivity coefficient is small and confined very near to the ground level for the stable boundary layer. Thus, the concentration does not mix much above the source height. The height of the reference box is $1/2$ km and the width is 10 km for the stable boundary layer. On the other hand, the pollutant becomes well mixed above the source height for the neutral and unstable boundary layers. Thus, a box of width 10 km and height 1 km is chosen for the neutral and unstable boundary layers. The total number of nodes in the reference grid is $114{,}705$ for the stable layer and $142{,}247$ for the neutral and unstable boundary layers. The initial grid for the adaptive solution is generated by refining a region around the point source. The refinement region lies horizontally within a 3-km circle with the point source as the center and it lies vertically within 300 m from the source. The initial number of nodes is $6{,}442$ for all three boundary layers. The number of nodes for the telescopic method remains $6{,}442$ throughout the simulation period. On the other hand, the adaptive grid is refined/derefined as the solution advances. The time step $\triangle t$ for the implicit-explicit scheme is small (usually less than 1 min) due to small vertical spacings near the ground level which affect the CFL condition. Instead of carrying out the adaptation after every time step (which is CPU intensive), the adaptation is carried out approximately every 20 min. This prevents large amounts of computational effort being used to perhaps refine very few tetrahedra each time step and does not significantly affect solution accuracy.

11.14.1 Grid Adaptation

Three sets of tolerance parameters are chosen for the adaptive grid method for each boundary-layer profile as described below. Let $TOLg$ be the maximum values of $tolg$ outside the source region. The refinement criteria of the edges are

(a) Refine edges to level 3 if $tolc > 9 \times 10^{10}$ and $tolg > 0.002 \times TOLg$

(b) Refine edges to level 2 if $tolc > 9 \times 10^{10}$ and $tolg > 0.00002 \times TOLg$

(c) Refine edges to level 1 if $tolc > 9 \times 10^{10}$ and $tolg > 0.000001 \times TOLg$

for the stable boundary layer.

The corresponding criteria for the neutral and unstable boundary layers are

(a) Refine edges to level 3 if $tolc > 10^{11}$ and $tolg > 0.01 \times TOLg$

(b) Refine edges to level 2 if $tolc > 10^{11}$ and $tolg > 0.0005 \times TOLg$

(c) Refine edges to level 1 if $tolc > 10^{11}$ and $tolg > 0.00005 \times TOLg$

The total number of nodes generated by the adaptive grid method are 60,000, 51,000, and 52,000 for the stable, neutral, and unstable boundary layers, respectively. The adaptive grid refinement in the vertical plane downwind along the plume centerline is shown in Figure 11.15. The concentration is confined near the ground level due

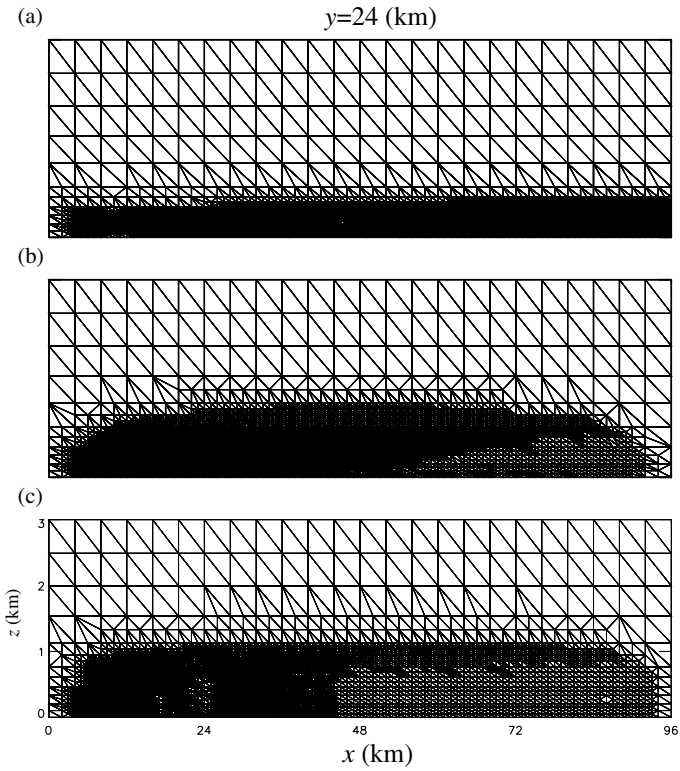


FIGURE 11.15
Grid refinement in the vertical plane through the source along the downwind direction for the (a) stable, (b) neutral, and (c) unstable boundary layers.

to small vertical diffusion for the stable case. This produces high spatial gradients within this region and grid refinement is highest near the ground. Since the vertical diffusion for the other two cases is larger compared to the stable boundary layer, the grid refinement extends to almost 1 km from ground level. It is also interesting to note that at large distances downwind from the source, the adaptive technique places more mesh points at the top of the boundary layer domain. This reflects the steep gradients found here due to a significant drop in the vertical diffusion coefficient K_z. This result may have significance for models attempting to represent boundary-layer

transport and mixing since the usual approach to vertical meshing is to place a greater number of mesh points close to the ground and not the top of the boundary layer. For the unstable boundary layer [see Figure 11.15(c)], the concentration becomes uniformly mixed below the inversion layer but very little diffusion is taking place above the inversion layer. The gradient is high near the inversion layer compared to the gradient near the ground. Thus, the edges near the inversion layer refine to a higher level than the edges near the ground.

The adaptive grid refinement at three different locations in the cross-wind direction is shown by Ghorai et al. [12]. The concentration gradients remain high for the stable case but low for the neutral and unstable cases far downwind from the source. Thus, the edges for the stable boundary layer, far downwind from the source, are refined to higher level than for the neutral and unstable cases. The gradients are high near the source for all the three cases and the edges are refined to the maximum level for all of them.

11.14.2 Downwind Concentration

The solutions downwind along the plume center-line in the ground level are shown in Figure 11.16. The maximum relative errors with respect to reference solutions are 16%, 20%, and 20% approximately for the stable, neutral, and unstable boundary layers, respectively. The maximum errors for the neutral and unstable cases occur far downwind from the source where the magnitude of the concentrations are small. The solution on the telescopic grid is accurate near the source region only due to the refinement in this region. Far downwind from the source, the solution on the telescopic grid differs widely from the reference solution. The programs have been run serially on an Origin2000 computer. For the neutral boundary layer, the total CPU times are approximately 1, 7, and 25 h for the telescopic, adaptive, and reference grids, respectively. Thus, the adaptive method is efficient compared to the other methods and achieves greater accuracy in a reasonable time.

11.15 Discussions and Conclusions

In this chapter we have described an MOL approach to the solution of transient reacting-flow problems. In particular, the atmospheric diffusion equation was solved by using unstructured, adaptive meshes with the MOL in two space dimensions. However, because of efficiency and positivity considerations, the three space dimensional case was solved by using operator splitting. The single most important conclusion is that there are key features of plume characteristics which cannot be represented by the coarse meshes generally used in regional scale models.

The test cases have demonstrated that adaptive methods can give much improved accuracy when compared to telescopic refinement methods particularly at large dis-

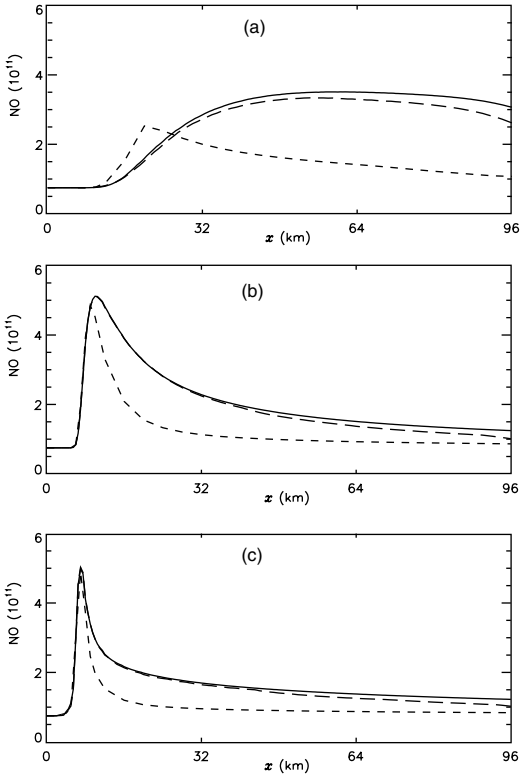


FIGURE 11.16
Comparison of the solution along the plume center-line in the ground level for the (a) stable, (b) neutral, and (c) unstable boundary layers. The solid, dotted, and dashed lines correspond to the solutions in the reference, telescopic, and adaptive grids.

tances from the source. The adaptive mesh methods may also use fewer mesh points than using fixed refined meshes since they are able to place mesh points where the solution requires them rather than in pre-defined locations where they may not be necessary for solution accuracy. However, there is an extra cost with the adaptive codes, that of periodically refining/coarsening the mesh. In particular, the test cases have demonstrated some important consequences of vertical mesh resolution for boundary-layer pollutant dispersion.

It is usual in tropospheric dispersion models to stretch the mesh in the vertical domain and place more solution points near to the ground. Close to ground-level sources, this often makes sense since it gives a better resolution of the initial stages of vertical mixing and of deposition to the ground. However, at large distance from their sources pollutants can become well mixed close to the ground and the important feature is their escape from the boundary layer to higher levels of the troposphere.

The results here demonstrate that for neutral and unstable boundary layers solution accuracy requires refined meshes not close to the ground but close to the inversion height where steep gradients can occur. The use of coarse meshes in this region could have a significant affect on the prediction of pollutants mixing out of the boundary layer for these conditions and may be a source of error in regional-scale, pollution-dispersion models. In a realistic boundary-layer model, vertical mixing profiles will change during the diurnal cycle making the *a priori* choice of vertical mesh structure difficult. Adaptive refinement would seem to be the simplest method for resolving such phenomena since the choice of mesh is made naturally according to the solution structure resulting from different stability conditions.

Our general conclusion is that the adaptive MOL approach works well for two space dimensional problems and in those cases it is possible to use standard codes providing that it is possible to make use of sophisticated linear algebra methods that are tailored to the problem. In the case of three-dimensional problems, however, it seems more necessary to use tailor-made codes either based on the MOL as in [17] or using the operator-splitting approach described here. Very recent work by Verwer and others has suggested that the approach we used in two dimensions should also be used in three space dimensions rather than introducing an operator-splitting error. The challenge now is to implement this in a sufficiently efficient way to make the MOL competitive with operator splitting in terms of efficiency.

Acknowledgments

This research has been supported by grants from NERC and from the Pakistan Government for one of us (IA). The funding for the SPRINT2D and 3D software has come from Shell Global Solutions. These calculations have been carried out on an Origin2000 machine with support from a JREI grant. We would also like to thank our many colleagues who helped on this project such as G. Hart, J. Smith, M. Pilling, and many others.

References

[1] I. Ahmad and M. Berzins, MOL solvers for hyperbolic PDEs with source terms, submitted to special issue of *Mathematics and Computers in Simulation.*

[2] I. Ahmad and M. Berzins, An algorithm for ODEs from atmospheric dispersion problems, *Applied Numerical Mathematics,* **25**, (1997), 137–149.

[3] M. Berzins, Temporal error control for convection-dominated equations in two space dimensions, *SIAM Journal on Scientific and Statistical Computing,* **16**, (1995), 558–580.

[4] M. Berzins and J.M. Ware, Solving convection and convection reaction problems using the MOL, *Applied Numerical Mathematics,* **20**, (1996), 83–99.

[5] M. Berzins, P.M. Dew, and R.M. Furzeland, Developing software for time-dependent problems using the method of lines and differential algebraic integrators, *Applied Numerical Mathematics,* **5**, (1989), 375–390.

[6] M. Berzins and J.M. Ware, Positive cell-centered finite volume discretization methods for hyperbolic equations on irregular meshes, *Applied Numerical Mathematics,* **16**, (1995), 417–438.

[7] M. Berzins, R. Fairlie, S.V. Pennington, J.M. Ware, and L.E. Scales, SPRINT2D: Adaptive Software for PDEs, *ACM Transactions on Mathematical Software,* **24**, (1998), 475–499.

[8] T.J. Barth and D.C. Jesperson, The design and application of upwind schemes on unstructured meshes, *AIAA-89-0366,* (1989), 9–12.

[9] T.J. Barth, Numerical aspects of computing viscous high reynolds number flows on unstructured meshes, AIAA Paper 91-0721, 29th Aerospace Sciences Meeting, January 7–10, (1991), Reno Nevada.

[10] R. Biswas and R.C. Strawn, A new procedure for dynamic adaption of 3D unstructured grids, *Applied Numerical Mathematics,* **13**, (1994), 437–452.

[11] D.P. Chock, A comparison of numerical methods for solving the advection equation III, *Atmos. Env.,* **25A**, (1991), 553–571.

[12] S. Ghorai, A.S. Tomlin, and M. Berzins, Resolution of pollutant concentrations in the boundary layer using a fully 3D adaptive gridding technique, *Atmospheric Environment,* **34**, 18, (2000), 2851–2863.

[13] G. Hart, A.S. Tomlin, J. Smith, and M. Berzins, Multi-scale atmospheric dispersion modelling by use of adaptive gridding techniques, *Environmental Monitoring and Assessment,* **52**, (1998), 225–238.

[14] Ø. Hov, Z. Zlatev, R. Berkowicz, A. Eliassen, and L.P. Prahm, Comparisons of numerical techniques for use in air pollution models with non-linear chemical reactions, *Atmospheric Environment,* **23**, (1989), 967–983.

[15] H.J. Jacobs, H. Feldman, H. Kass, and M. Messesheimer, The use of nested models for air pollution studies: an application of the EURAD model to a SANA episode, *Journal of Applied Meteorology,* **34**, (1995), 1301–1319.

[16] B. Joe and R.B. Simpson, Triangular meshes for regions of complicated shape, *Int. J. Numer. Meth. Eng.,* **23**, (1991), 987–997.

[17] C.R. Johnson, M. Berzins, L. Zhukov, and R. Coffey, SCIRun: Application to atmospheric dispersion problems using unstructured meshes, pp. 111–122 in *Numerical Methods for Fluid Dynamics VI,* M.J. Baines, ed., ICFD, Wolfson Building, Parks Road, Oxford. ISBN 0 9524929 11, (1998).

[18] R.J. Leveque and H.C. Yee, A study of numerical methods for hyperbolic conservation laws with stiff source terms, *Journal of Computational Physics,* **86**, (1990), 187–210.

[19] S.C. Liu, M. Trainer, F.C. Fehsenfeld, D.D. Parrish, E.J. Williams, D.W. Fahey, G. Hubler, and P.C. Murphey, Ozone production in the rural troposphere and the implications for regional and global ozone distributions, *J. Geophys. Res.,* **92**, (1987), 4191–4207.

[20] R. Mathur and L.K. Peters, Adjustment of wind fields for application in air pollution modelling, *Atmospheric Environment,* **24A**, (1990), 1095–1106.

[21] S.V. Pennington and M. Berzins, New NAG library software for first-order partial differential equations, *ACM Transactions on Mathematical Software,* **20**, (1994), 63–99.

[22] L.K. Peters, C.M. Berkovitz, G.R. Carmichael, R.C. Easter, G. Fairweather, S.J. Ghan, J.M. Hales, L.R. Leung, W.R. Pennell, F.A. Potra, R.D. Saylor, and T.T. Tsang, The current and future direction of Eulerian models in simulating the tropospheric chemistry and transport of trace species: a review, *Atmospheric Environment,* **29**, (1995), 189–222.

[23] M. Putti and C. Cordes, Finite element approximation of the diffusion operator on tetrahedra, *SIAM Journal on Scientific Computing,* **19**, (1998), 1154–1168.

[24] T. Rajaona, M.C. Ciccoli, and A. Coppalle, Local refinement around a pollution source using a domain decomposition method, in *Proc. Int. Conf. on Air Pollution Modelling and Simulation,* Paris, France (1998).

[25] J.H. Seinfeld, (1986) *Air Pollution,* Wiley, New York.

[26] S. Sillman, J.A. Logan, S.C. Wofsy, *J. Geophys. Res.,* **95**, (1990), 1837.

[27] W. Skamarock, J. Oliger, and R.L. Street, Adaptive grid refinement for numerical weather prediction, *Journal of Computational Physics,* **80**, (1989), 27–60.

[28] W. Speares and M. Berzins, A 3D unstructured mesh adaptation algorithm for time-dependent shock-dominated problems, *International Journal for Numerical Methods in Fluids,* **25**, (1997), 81–104.

[29] S. Spekreijse, Multigrid solution of monotone second order discretizations of hyperbolic conservation laws, *Math. Comp.,* **47**, 179, (1987), 135–155.

[30] S. Sunderam, J.A. Logan, and S.C. Wofsy, A regional scale model for ozone in the united states with subgrid representation of urban and power plant plumes, *Journal of Geophysical Research,* **95**, (1990), 5731–5748.

[31] O. Talat, A quantitative analysis of numerical diffusion introduced by advection algorithms in air quality models, *Atmospheric Environment,* **31**, (1997), 1933–1940.

[32] A.S. Tomlin, M. Berzins, J. Ware, J. Smith, and M.J. Pilling, On the use of adaptive gridding methods for modelling chemical transport from multi-scale sources, *Atmospheric Environment,* **31**, (1997), 2945–2959.

[33] A.S. Tomlin, S. Ghorai, G. Hart, and M. Berzins, The use of 3D adaptive unstructured meshes in pollution modelling, in *Large-Scale Computations in Air Pollution Modelling,* Z. Zlatev et al., ed., (1999) Kluwer Academic Publishers.

[34] M. VanLoon, Numerical methods in smog prediction, Ph.D. Thesis, (1996) CWI Amsterdam.

[35] I. Verwer, Gauss–Seidel iteration for stiff ODEs from chemical kinetics, *SIAM Journal on Scientific Computing,* **15**, (1994), 1243–1250.

[36] C.B. Vreugdenhil and B. Koren, Numerical methods for advection-diffusion problems, *Notes on Numerical Fluid Mechanics,* **45**, (1993), Vieweg, Braunschweig/Weisbaden, ISBN 3-528-07645-3.

[37] M. Wierse, A new theoretically motivated higher order upwind scheme on unstructured grids of simplices, *Advances in Computational Mathematics,* **7**, (1997), 303–335.

Chapter 12

Two-Dimensional Model of a Reaction-Bonded Aluminum Oxide Cylinder

M.J. Watson, H.S. Caram, H.M. Chan, M.P. Harmer, Ph. Saucez, A. Vande Wouwer, and W.E. Schiesser

12.1 Introduction

The reaction-bonded aluminum oxide (RBAO) process has been shown to have many advantages over conventional ceramic processing [1, 2, 3, 4]. The RBAO process starts with intensely milled aluminum and Al_2O_3 powders which can be compacted into a variety of shapes and sizes. The main advantage of the process is that the compacted shapes are strong enough to be machined prior to firing. The porous, compacted samples are then heat treated in air, to oxidize the aluminum, producing Al_2O_3-based ceramics.

The oxidation of aluminum is highly exothermic and, as a consequence, an ignition front has been observed passing over the sample's surface during firing. An ignition wavefront is undesirable for the RBAO process because both thermal and chemical stresses are developed. The thermal stresses are transitory and are caused by the large temperature difference between the hot reaction zone and the cooler unreacted zone. The chemical stresses are caused by the 28% volumetric expansion associated with the oxidation of aluminum. The chemical stresses are not transitory, but remain in the wake of the ignition wave-front, as is evident from the steep composition gradients in the radial direction, shown in Figure 12.1. The depth of the reacted shell is restricted by the radial diffusion of oxygen through the pores of the rod. Sample failure is usually observed soon after ignition.

In this chapter a two-dimensional, transient model of the reaction behavior of a RBAO rod is developed. The model can be used to investigate the reaction behavior under a variety of experimental conditions, including variations in initial aluminum concentration, furnace heating cycles, furnace atmosphere, and furnace temperature gradients. Specifically, the model is used here to describe the reaction behavior during ignition. The results from the model, which gives the oxygen concentration,

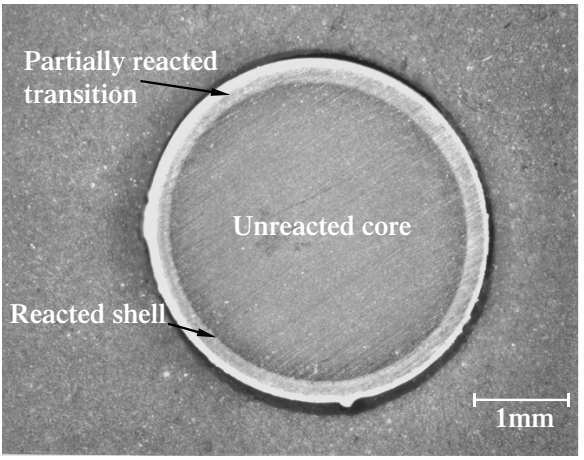


FIGURE 12.1
Light optical micrograph of the cross-section of an RBAO rod after an ignition front has passed.

aluminum concentration, and temperature as a function of position and time, can then be used to evaluate the stress distribution within the sample.

In this study we use the method of lines (MOL) to solve the set of three two-dimensional simultaneous partial differential equations (PDEs), developed in the next section. Three variations, based on three different configurations of the two-dimensional spatial mesh, are used to calculate the spatial derivatives:

1. fixed, equally spaced nodes in the axial and radial directions

2. fixed, equally spaced nodes in the axial direction, and fixed, non-equally spaced nodes in the radial direction

3. time-adapted nodes in the axial direction, and fixed, non-equally spaced nodes in the radial direction

In Section 12.2 the three PDEs that are used to describe the distribution of temperature, aluminum concentration, and oxygen concentration are developed. They are presented in dimensionless form with the appropriate initial and boundary conditions. The equations are solved using the numerical method of lines. The three variations listed above are used to evaluate the spatial derivatives. In Section 12.3 the numerical solutions are presented, and the different mesh configurations are compared. The results are discussed in Section 12.4 and possibilities for improvement are given.

12.2 Model Development

The following is the development of a continuum model that describes the reaction behavior within a cylinder of RBAO. The model describes changes in oxygen concentration, aluminum concentration, and sample temperature with time along the axial and radial directions of the cylinder.

12.2.1 Model Assumptions

1. The bulk (furnace) concentration of oxygen is given by the ideal gas law:

$$C_{O_2,\infty} = \frac{(0.21)P}{R_g T_\infty} ,$$

where T_∞ is the furnace temperature, R_g is the gas constant, P is the furnace pressure (atmospheric), and 0.21 is the mol fraction of O_2 in air.

2. For the reaction

$$2\mathrm{Al} + \frac{3}{2}\mathrm{O_2} \longrightarrow \mathrm{Al_2O_3} ,$$

the reaction rate, $\Re$, is described by

$$\Re = k(T) C_{O_2} C_{Al}^2 , \tag{12.1}$$

where C_{Al} is the concentration of aluminum, C_{O_2} is the concentration of oxygen, and $k(T)$ is the reaction rate constant. The rate constant, $k(T)$, is described by Arrhenius temperature dependence given by:

$$k(T) = k_0 \exp\left(\frac{-E_a}{R_g T}\right) , \tag{12.2}$$

where k_0 is the pre-exponential factor, E_a is the activation energy, and T is absolute temperature. The expression for the reaction rate is empirical and is the same as that used in a similar study [5].

3. There is no variation in the θ-direction. Only variations in the radial direction, r, and the axial direction, z, are considered.

4. For simplicity, properties such as specific heat, density, diffusivity, and conductivity are kept constant.

12.2.2 Continuum Model Equations

The aluminum in the rod reacts to form Al_2O_3. The general mass balance for aluminum is given by:

$$\frac{\partial C_{Al}}{\partial t} = -2\Re , \tag{12.3}$$

where t is time and the coefficient of 2 is required for stoichiometry.

The oxygen balance is given by:

$$\frac{\partial C_{O_2}}{\partial t} = D\nabla^2 C_{O_2} - \frac{3}{2}\Re \,, \tag{12.4}$$

where D is the effective diffusivity of oxygen in the pore structure, ∇^2 is the Laplacian operator, and the coefficient of 3/2 is required for stoichiometry. Ignoring any variation in the θ-direction, Equation 12.4 becomes:

$$\frac{\partial C_{O_2}}{\partial t} = D\left(\frac{\partial^2 C_{O_2}}{\partial z^2} + \frac{1}{r}\frac{\partial}{\partial r}r\frac{\partial C_{O_2}}{\partial r}\right) - \frac{3}{2}\Re \,. \tag{12.5}$$

The energy balance is given by:

$$\rho c_p \frac{\partial T}{\partial t} = \lambda\nabla^2 T + (-\Delta H)\Re \,, \tag{12.6}$$

where ρ is the density, c_p is the specific heat, λ is the thermal conductivity, and $(-\Delta H)$ is the heat of reaction. Ignoring any variation in the θ-direction, Equation 12.6 becomes:

$$\frac{\partial T}{\partial t} = \alpha\left(\frac{\partial^2 T}{\partial z^2} + \frac{1}{r}\frac{\partial}{\partial r}r\frac{\partial T}{\partial r}\right) + \frac{(-\Delta H)}{\rho c_p}\Re \,, \tag{12.7}$$

where $\alpha = \frac{k}{\rho c_p}$ is the thermal diffusivity.

12.2.3 Initial and Boundary Conditions

The initial conditions, at $t = 0$, for the cylinder are

$$\begin{aligned} C_{Al}(0, z, r) &= C_{Al,0} \\ T(0, z, r) &= T_0 \\ C_{O_2}(0, z, r) &= C_{O_2,0} = \frac{0.21P}{R_g T_0} \,, \end{aligned} \tag{12.8}$$

where the initial concentration of aluminum and initial temperature are constants and independent of r and z. The initial oxygen concentration is found from the ideal gas law.

Heat is transferred to or from the furnace to the outer surface of the cylinder through convective and radiative heat transfer. A flux balance on the left- and right-hand sides of the cylinder, corresponding to $z = 0$ and $z = L$, respectively, gives the boundary conditions for the energy balance:

$$\begin{aligned} \lambda\frac{\partial T}{\partial z}(t, 0, r) &= h\left(T(t, 0, r) - T_\infty\right) + \sigma\epsilon\left(T^4(t, 0, r) - T_\infty^4\right) \\ \lambda\frac{\partial T}{\partial z}(t, L, r) &= -h\left(T(t, L, r) - T_\infty\right) - \sigma\epsilon\left(T^4(t, L, r) - T_\infty^4\right) \,, \end{aligned} \tag{12.9}$$

where $T_\infty = T_\infty(t, z, r)$ is the furnace temperature, h is the heat-transfer coefficient, σ is the Stefan–Boltzman constant, and ϵ is the emmisivity. A heat-flux balance on the curved surface of the cylinder, corresponding to $r = R$, gives:

$$\lambda \frac{\partial T}{\partial r}(t, z, R) = -h\left(T(t, z, R) - T_\infty\right) - \sigma\epsilon\left(T^4(t, z, R) - T_\infty^4\right) , \qquad (12.10)$$

while the symmetry condition at the center of the cylinder, $r = 0$, gives:

$$\frac{\partial T}{\partial r}(t, z, 0) = 0 . \qquad (12.11)$$

Balancing the mass flux of oxygen at the left, $z = 0$, and right, $z = L$, boundaries of the cylinder yields:

$$\begin{aligned} D\frac{\partial C_{O_2}}{\partial z}(t, 0, r) &= k_m\left(C_{O_2}(t, 0, r) - C_{O_2,\infty}\right) \\ D\frac{\partial C_{O_2}}{\partial z}(t, L, r) &= -k_m\left(C_{O_2}(t, L, r) - C_{O_2,\infty}\right) , \end{aligned} \qquad (12.12)$$

where k_m is the mass transfer coefficient and C_∞ is the concentration of oxygen in the furnace. Balancing the mass flux on the curved surface of the cylinder, corresponding to $r = R$, gives:

$$D\frac{\partial C_{O_2}}{\partial r}(t, z, R) = -k_m\left(C_{O_2}(t, z, R) - C_{O_2,\infty}\right) , \qquad (12.13)$$

while the symmetry condition at the center of the cylinder, $r = 0$, gives:

$$\frac{\partial C_{O_2}}{\partial r}(t, z, 0) = 0 . \qquad (12.14)$$

12.2.4 Parameters

The model parameters that are used are listed in Table 12.1. The kinetic parameters, E_a and k_0, are chosen to agree with an existing model that describes the RBAO reaction behavior of a flat slab [5]. The other parameters are either measured or calculated from correlations found in the literature.

12.2.5 Dimensionless Equations

The set of PDEs can be rendered dimensionless by using the following variables:

$$u = \frac{C_{Al}}{C_{Al,0}}, \quad v = \frac{C_{O_2}}{C_{O_2,0}}, \quad w = \frac{T}{T_{ad}} ,$$

$$\xi = \frac{r}{R}, \quad \eta = \frac{z}{L}, \quad \tau = \left[k_0 \exp(-\gamma) C_{Al,0} C_{O_2,0}\right] t ,$$

Table 12.1 Model Parameters

Property	Symbol	Value	Unit
Length	L	0.1	m
Radius	R	0.002	m
Density	ρ	2460	kg/m^3
Specific heat	c_p	1000	J/(kg K)
Conductivity	λ	1.4	W/(m K)
Heat of formation	$(-\Delta H)$	1669792	J/mol
Gas constant	R_g	8.314	J/(mol K)
Activation energy	E_a	170000	J/mol
Arrhenius factor	k_0	300	m^6/(mol^{2}s)
Heat-transfer coefficient	h	15	W/(m^{2}K)
Stefan-Boltzman constant	σ	5.67e-8	W/(m^{2}K^4)
Emissivity	ϵ	0.1	
Diffusivity	D	1.5×10^{-6}	m^2/s
Mass-transfer coefficient	k_m	0.03	m/s
Initial Al concentration	$C_{Al,0}$	20000	mol/m^3
Initial temperature	T_0	750	K

where

$$T_{ad} = T_0 + \frac{(-\Delta H)C_{Al,0}}{2\rho C_p}, \quad \gamma = \frac{E_a}{R_g T_{ad}}, \tag{12.15}$$

and T_{ad} represents the adiabatic temperature rise. The PDEs become:

$$\frac{\partial u}{\partial \tau} = -2\exp\left[\gamma\left(1-\frac{1}{w}\right)\right] vu^2 \tag{12.16}$$

$$\frac{\partial v}{\partial \tau} = -\psi\exp\left[\gamma\left(1-\frac{1}{w}\right)\right] vu^2 + \chi_R\left(\frac{1}{\xi}\frac{\partial}{\partial\xi}\xi\frac{\partial v}{\partial\xi}\right) + \chi_L\frac{\partial^2 v}{\partial\eta^2} \tag{12.17}$$

$$\frac{\partial w}{\partial \tau} = +\beta\exp\left[\gamma\left(1-\frac{1}{w}\right)\right] vu^2 + \phi_R\left(\frac{1}{\xi}\frac{\partial}{\partial\xi}\xi\frac{\partial w}{\partial\xi}\right) + \phi_L\frac{\partial^2 w}{\partial\eta^2} \tag{12.18}$$

where

$$\phi_L = \frac{\alpha\exp(\gamma)}{L^2 k_0 C_{Al,0} C_{O_2,0}}, \quad \phi_R = \frac{\alpha\exp(\gamma)}{R^2 k_0 C_{Al,0} C_{O_2,0}}, \quad \beta = \frac{(-\Delta H)C_{Al,0}}{\rho C_p T_{ad}},$$

$$\chi_L = \frac{D\exp(\gamma)}{L^2 k_0 C_{Al,0} C_{O_2,0}}, \quad \chi_R = \frac{D\exp(\gamma)}{R^2 k_0 C_{Al,0} C_{O_2,0}}, \quad \psi = \frac{3C_{Al,0}}{2C_{O_2,0}}.$$

In dimensionless variables, the initial conditions become:

$$\begin{aligned} u(0,\eta,\xi) &= 1 \\ v(0,\eta,\xi) &= 1 \\ w(0,\eta,\xi) &= 1-\frac{\beta}{2}. \end{aligned} \tag{12.19}$$

The thermal boundary conditions become:

$$\begin{aligned}
\frac{\partial w}{\partial \eta}(\tau, 0, \xi) &= \kappa_L(w - w_\infty) + \Upsilon_L\left(w^4 - w_\infty^4\right) \\
\frac{\partial w}{\partial \eta}(\tau, 1, \xi) &= -\kappa_L(w - w_\infty) - \Upsilon_L\left(w^4 - w_\infty^4\right) \\
\frac{\partial w}{\partial \xi}(\tau, \eta, 1) &= -\kappa_R(w - w_\infty) - \Upsilon_R\left(w^4 - w_\infty^4\right) \\
\frac{\partial w}{\partial \xi}(\tau, \eta, 0) &= 0\,, \qquad (12.20)
\end{aligned}$$

where

$$\kappa_L = \frac{Lh}{\lambda}, \quad \Upsilon_L = \frac{\sigma \epsilon L T_{ad}^3}{\lambda}, \quad \kappa_R = \frac{Rh}{\lambda}, \quad \Upsilon_R = \frac{\sigma \epsilon R T_{ad}^3}{\lambda}\,,$$

and the boundary conditions for oxygen mass transfer become:

$$\begin{aligned}
\frac{\partial v}{\partial \eta}(\tau, 0, \xi) &= \Xi_L(v - v_\infty) \\
\frac{\partial v}{\partial \eta}(\tau, 1, \xi) &= -\Xi_L(v - v_\infty) \\
\frac{\partial v}{\partial \xi}(\tau, \eta, 1) &= -\Xi_R(v - v_\infty) \\
\frac{\partial v}{\partial \xi}(\tau, \eta, 0) &= 0\,, \qquad (12.21)
\end{aligned}$$

where

$$\Xi_L = \frac{Lk_m}{D}, \quad \Xi_R = \frac{Rk_m}{D}\,.$$

12.2.6 Method of Solution

By dividing the axial coordinate, η, into NL spatial points, and the radial coordinate, ξ into NR spatial points, and using discrete approximations to describe the spatial derivatives, the three PDEs are approximated by $3 \times NL \times NR$ ordinary differential equations (ODEs). This is known as the numerical method of lines [6] and is used to solve the simultaneous mass and energy balances with the appropriate boundary conditions. The set of ODEs is solved using the `LSODES` integrator [7, 8, 9] for stiff ODEs with a sparse Jacobian matrix. The integrator uses an implicit method, based on backward differentiation formulas, of order 5 with step-size control. The maximum absolute and relative integration errors are set to 10^{-5}. The parameters that are used to solve the equations are listed in Table 12.2.

Three variations are used to calculate the distribution of the nodes and the spatial derivatives.

Table 12.2 Solution Parameters

Parameter	Symbol	Value
Axial nodes	NL	26
Radial nodes	NR	11
Tolerance		10^{-5}
Adaptation parameter	α	1
Adaptation parameter	β	100

Method 1. The first variation uses fixed, equally spaced nodes in the radial and axial directions. The second-order spatial derivatives are calculated using subroutine `DSS044` and the first-order spatial derivatives in the radial direction are calculated using subroutine `DSS034` from the `DSS/2` package [10].

Method 2. The second variation uses fixed, equally spaced nodes in the axial direction, and fixed, non-equally spaced nodes in the radial direction, using subroutine `DSS044` to calculate the second-order spatial derivatives in the axial direction, and subroutine `DSS032` to calculate the first- and second-order spatial derivatives in the radial direction.

Method 3. The third variation uses fixed, non-equally spaced nodes in the radial direction, and time-adapted nodes in the axial direction. Subroutine `AGE` [11] is used to adapt the axial nodes. Finite differences, as implemented in subroutine `WEIGHTS` [12] are used to calculate the second-order spatial derivative in the axial direction. Subroutine `DSS032` is used to calculate the first- and second-order spatial derivatives in the radial direction.

These three methods will be referred to as Methods 1, 2, and 3, respectively, throughout.

The spatial remeshing algorithm, `AGE`, has been applied to a variety of one-dimensional transient problems [11]. Here it is applied to the two-dimensional problem [Equations (12.16) through (12.21)] by basing the grid adaptation in the axial direction on the solution at $r = R$, or $\xi = 1$. The spatial remeshing algorithm is based on the second derivative:

$$\int_{\eta_{i-1}^{k+1}}^{\eta_i^{k+1}} \left(\alpha + \left\| \frac{\partial^2 U}{\partial \eta^2}(t_{k+1}, \eta) \right\|_\infty \right) d\eta \approx \text{ constant} , \tag{12.22}$$

where $U(t, \eta) = \{u(t, \eta, 1), v(t, \eta, 1), w(t, \eta, 1)\}$. A parameter β is used to avoid excessive grid distortion, i.e., the value of the derivatives that exceed β are reduced to β. Values of α and β are given in Table 12.2. The grid adaptation is performed at each print time interval (i.e., in contrast with the static gridding methods presented in Chapter 1, grid adaptation occurs at regular time intervals rather than after a certain number of variable time steps).

12.3 Results

In this section the numerical solutions of Equations (12.16) through (12.21) are presented. The three different configurations of the two-dimensional spatial mesh are used to evaluate the spatial derivatives and the different numerical solutions are compared.

12.3.1 Furnace Conditions

To simulate the firing of a sample in a furnace, the sample is heated at 5 K/min and a linear temperature gradient of 40 K is assumed from the left end of the sample to the right end. The initial furnace temperature is 750 K ($= T_0$), thus:

$$T_\infty(t, z) = T_0 + \frac{5}{60}t - 40\frac{z}{L}\,, \tag{12.23}$$

and in dimensionless units:

$$w_\infty(\tau, \eta) = 1 - \frac{\beta}{2} + \frac{5\tau}{60T_{ad}k_0 \exp(-\gamma)C_{Al,0}C_{O_2,0}} - \frac{40\eta}{T_{ad}}\,. \tag{12.24}$$

Integration continues for 14 min, until the furnace temperature reaches 820 K. The temperature and concentrations are recorded 100 times within this period.

12.3.2 Numerical Solutions

Figures 12.2, 12.3, and 12.4 show the time progression of the temperature and concentration distributions of the ignited rod. The shading of the solution has been interpolated for clarity. The high temperature ignition front is seen propagating from the hot end on the left to the right of the sample in Figure 12.2. Similarly in Figure 12.3, a region of reacted aluminum (black) follows the ignition front. The depth of the reacted zone is limited by the rate of diffusion of oxygen into the sample relative to the rate of reaction. This is shown in Figure 12.4. All of the oxygen within the porous structure of the sample is consumed by the oxidation reaction. As fresh oxygen from the furnace diffuses into the rod, it is consumed near the surface before it is allowed to diffuse to the center. The limited depth of the reaction zone agrees qualitatively with the micrograph of Figure 12.1.

A comparison of the node placement of methods 1, 2, and 3, at $t = 210$ sec, is shown in Figure 12.5. The shading of the space between the nodes is based on the value of the upper left node. Figure 12.5(a) shows the solution with equally spaced nodes in the radial and axial direction. Figures 12.5(b) and (c) have radial grid spacing based on concentric cylinders of equal volume, ΔV:

$$\Delta V = \pi r_i^2 - \pi r_{i-1}^2 = \frac{\pi R^2}{NR - 1}\,. \tag{12.25}$$

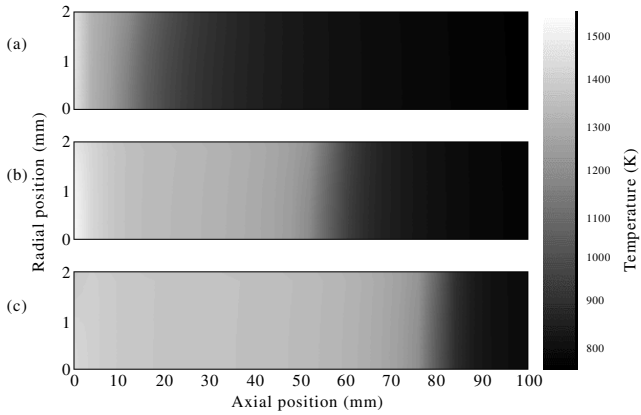


FIGURE 12.2
Temperature distribution within a RBAO rod: (a) 160 sec; (b) 210 sec; (c) 260 sec.

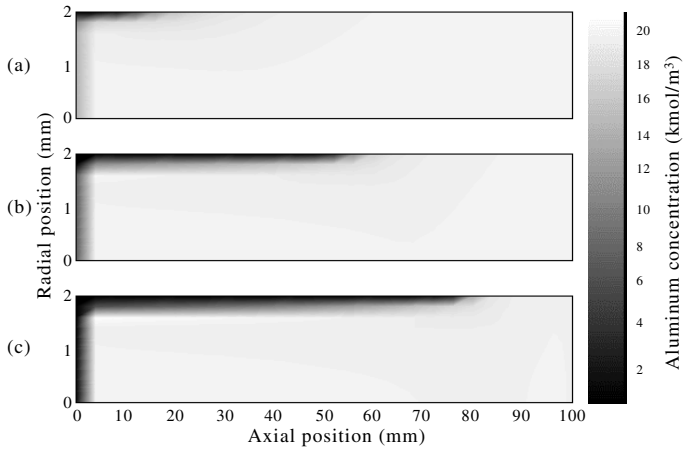


FIGURE 12.3
Aluminum concentration distribution within a RBAO rod: (a) 160 sec; (b) 210 sec; (c) 260 sec.

This allows for greater spatial resolution and accuracy close to the surface of the cylinder, where the majority of the oxidation reaction occurs during ignition. Figure 12.5(c) shows the solution when axial grid adaptation is used. The grid points are spaced closer together near the left end, and at the ignition front near the center.

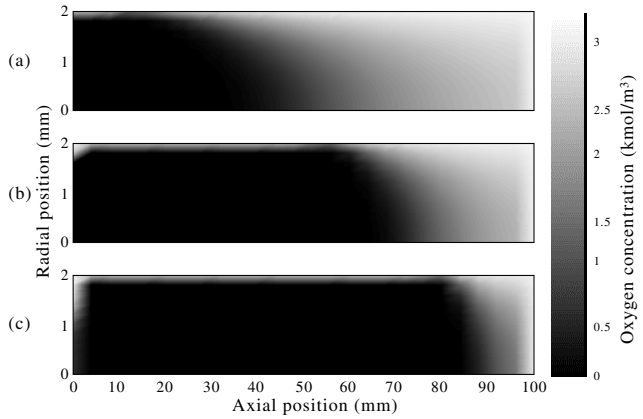


FIGURE 12.4
Oxygen concentration distribution within a RBAO rod: (a) 160 sec; (b) 210 sec; (c) 260 sec.

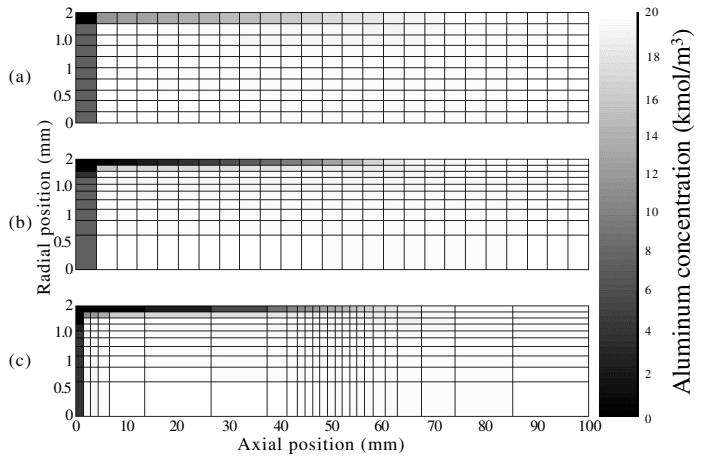


FIGURE 12.5
Aluminum concentration distribution at $t = 210$ sec showing a comparison of the node placements. The nodes are distributed according to: (a) Method 1; (b) Method 2; (c) Method 3.

The close spacing of the grid points follows the regions where concentration and temperature gradients are steep.

Figures 12.6 and 12.7 show the temperature and concentration distributions, respectively, on the outer surface of the rod, corresponding to $r = R$. The nodes are distributed evenly in the axial direction in Figures 12.6(a) and (b) and 12.7(a) and

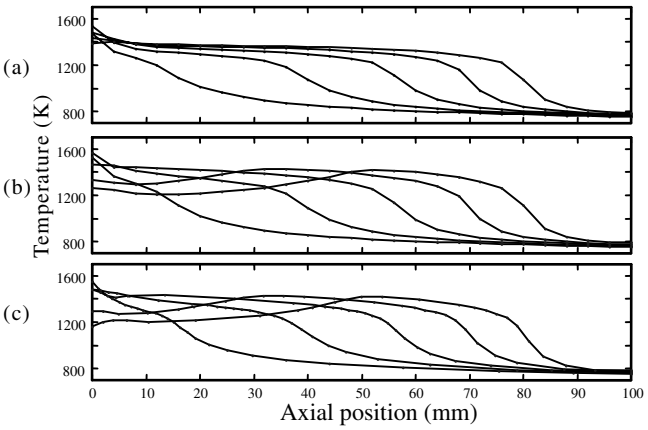


FIGURE 12.6
Axial temperature distribution at $r = R$ at $t = 160$, 185, 210, 225, and 260 sec. The nodes are distributed according to: (a) Method 1; (b) Method 2; (c) Method 3.

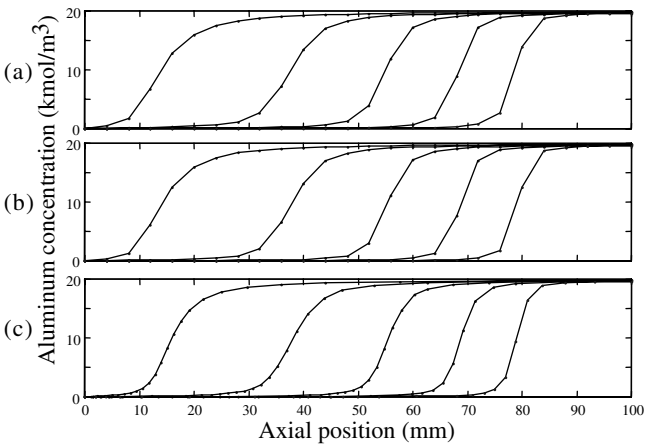


FIGURE 12.7
Axial concentration distribution at $r = R$ at $t = 160$, 185, 210, 225, and 260 sec. The nodes are distributed according to: (a) Method 1; (b) Method 2; (c) Method 3.

(b). As a consequence the solution is somewhat jagged at the wavefront. Interestingly, the method used to incorporate the boundary condition, Equation 12.9, into the calculation of the spatial derivative at $z = 0$ has an effect on the solution for the temperature distribution. The second derivative in Figure 12.6(a) and (b) is evaluated in subroutine `DSS044` using the fourth-order approximation:

$$\frac{\partial^2 w_1}{\partial \eta^2} \approx \frac{1}{12(\Delta\eta)^2}\left(-\frac{415}{6}w_1 + 96w_2 - 36w_3 + \frac{32}{3}w_4 - \frac{3}{2}w_5 - 50\frac{\partial w_1}{\partial \eta}\Delta\eta\right) ,$$

whereas in Figure 12.6(c), it is evaluated using the second-order approximation:

$$\frac{\partial^2 w_1}{\partial \eta^2} \approx \frac{2}{(\Delta\eta_1)^2}\left(\frac{\partial w_1}{\partial \eta}\Delta\eta_1 - w_1\right) .$$

However, the lower-order method that is used in the axial grid adaptation should not cause significant inaccuracy because of the closer spacing of the nodes at the boundary.

The solutions found from the three different methods agree, despite the small differences at the boundaries. The advantage of the axial grid adaptation is greater accuracy and spatial resolution at the wavefront.

Figure 12.8 compares the aluminum concentration distributions in the radial direc-

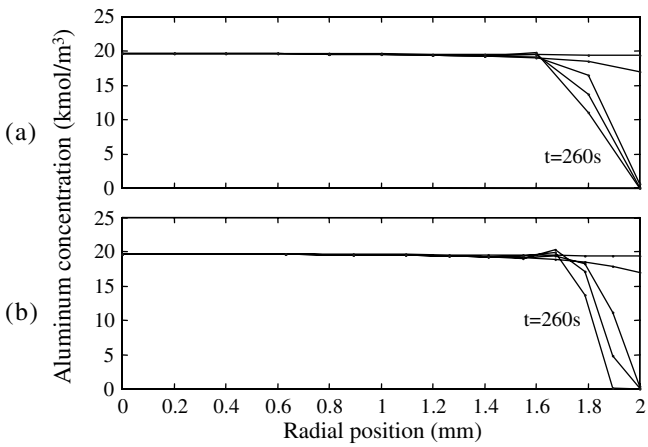


FIGURE 12.8
Radial concentration distribution at $z = 44$ mm at $t = 160, 185, 210, 225$, and 260 sec. The nodes are distributed according to: (a) Method 1; (b) Method 2.

tion when the nodes are equally spaced, Figure 12.8(a), and spaced on equal volumes according to Equation (12.25), Figure 12.8(b). The greater spatial resolution near the outer surface of the cylinder is required to track the inward-moving reaction front. The most dramatic illustration of this is found in the curve corresponding to $t = 260$ sec. On the equally spaced grid, the solution goes from 0 kmol/m^3 at the outer-most node

to 11 kmol/m^3 at the next node in. On the grid spaced on equal volumes, the zone of complete oxidation, corresponding to a concentration of 0 kmol/m^3 has penetrated 0.1 mm into the sample.

Both solutions show a small region where the concentration goes beyond the initial concentration. This suggests that more grid points are required in the radial direction to improve the accuracy of the solution.

12.4 Discussion

A banded diagonal structure of the ODEs, and hence the Jacobian matrix, cannot be maintained for this two-dimensional problem. The large size of the Jacobian, caused by the large number of ODEs, excludes the use of full Jacobian matrix evaluation. The integrator `LSODES` was chosen because of the efficient nature of the sparse Jacobian matrix calculation. This allowed for fast computation of the solution to the problem when the grid was fixed, as in Methods 1 and 2, where the calculation took 2.8 min and 3.2 min, respectively, on a 333 MHz PC. The solution took considerably longer (14.5 min) to calculate when Method 3 was used. This is because `LSODES` uses variable order backward differentiation formulas to perform each integration step. Every time the grid was adapted the integrator had to re-initialize, which is computationally expensive. Ideally, a single-step solver, such as the implicit Runge–Kutta integratator `RADAU5` [13], should be used.

Subroutine `AGE` successfully tracked the ignition wavefront and adapted the grid points in the region of the wavefront. The spatial grid was adapted at specified discrete time intervals, determined by the printing time interval. Perhaps greater success could be achieved by using the adaptation algorithm after a fixed number of integrator time steps. This would allow the integrator, which has time step-size control, to adapt more frequently when the solution to the problem is changing faster.

For comparison purposes, results were obtained with the adaptive mesh refinement algorithm `AGEREG` discussed in Chapter 2, i.e., using a time-varying number of nodes adapted periodically after a certain number of steps, and the `RK` solver `RADAU5`. The dimensionless oxygen concentration distribution is shown at 14 equally spaced time intervals at $r = R$ in Figure 12.9. The results were compared with 111 fixed, equally spaced nodes, shown in Figure 12.10. The number of fixed nodes was chosen so as to have the same CPU time as with a variable number of nodes computed by `AGEREG`. The computational statistics are summarized in Table 12.3, which compares the number of axial nodes NL, the number of function evaluations FNS, the number of Jacobian evaluations JACS, and the number of computed steps STEPS. The evolution of the number of grid points used by `AGEREG` was 101, 48, 49, 50, 52, 87, 93, 102, 95, 81, 67, 67, 55, 51, 52, 51 for the 14 time intervals shown in Figure 12.9. The results give similar global accuracy, but improved resolution of small-scale solution features near the boundaries at $z = 0$ and 1 when adaptive grids are used.

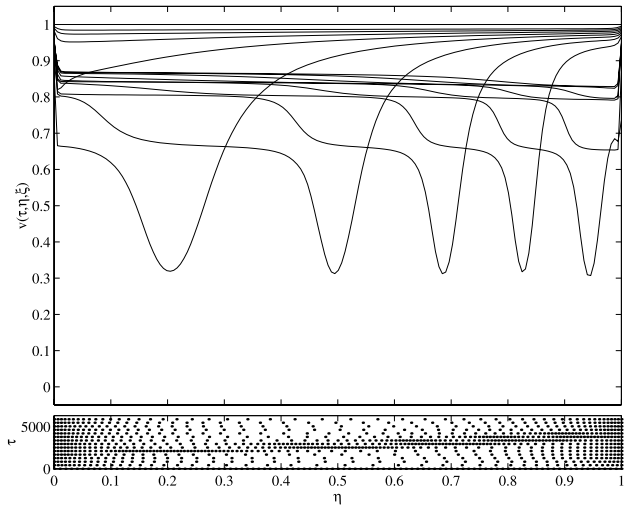


FIGURE 12.9
Dimensionless oxygen concentration distribution at $\xi = 1$ using a variable number of nodes.

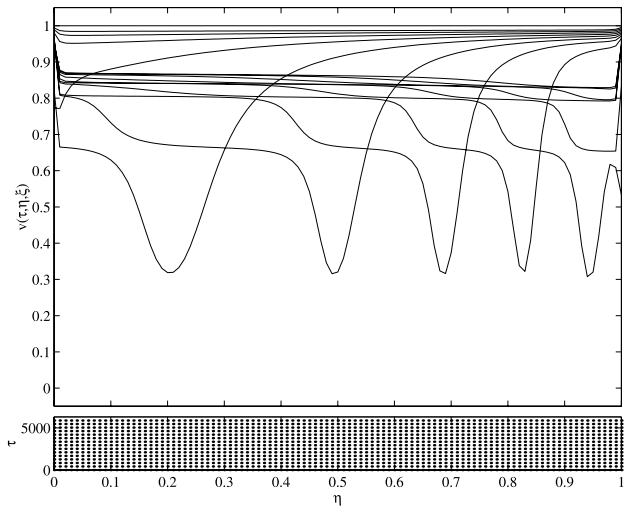


FIGURE 12.10
Dimensionless oxygen concentration distribution at $\xi = 1$ using 111 fixed, equally spaced nodes.

Table 12.3 Computational Statistics

Grid	NL	FNS	JACS	STEPS
AGEREG	48 – 101	18229	887	1829
uniform	111	13637	662	1333

The results obtained thus far suggest that further improvements could probably be achieved by using 2D adaptation techniques (in the radial direction as well as the axial direction) such as the one reported by Steinebach and Rentrop in Chapter 6.

12.5 Summary

A method to use a one-dimensional spatial remeshing algorithm on a two-dimensional problem, in cylindrical coordinates, was presented. The axial grid adaptation was based on the condition of the second derivative at the outer surface of the cylinder. The solution displayed a moving ignition wave-front. The algorithm was able to focus the axial nodes in the region of the ignition front, thereby improving the spatial resolution and accuracy in that region. By appropriately choosing the numerical integrator, similar global accuracy was obtained using a fixed, equally spaced grid, but better local accuracy was obtained on an adapted grid with a variable number of nodes, without compromising computational efficiency.

Acknowledgment

I gratefully acknowledge the kind financial support of the U.S. Office of Naval Research (grant N0014-96-1-0426) under the monitoring of Dr. S. Fishman.

References

[1] N. Claussen, R. Janssen, and D. Holz, The Reaction Bonding of Aluminum Oxide, *Journal of the Ceramic Society of Japan,* **103**(8), (1995), 1–10.

[2] N. Claussen, S. Wu, and D. Holz, Reaction Bonding of Aluminum Oxide (RBAO) Composites: Processing, Reaction Mechanisms and Properties, *Journal of the European Ceramic Society,* **14**, (1994), 97–109.

[3] D. Holz, S. Wu, S. Scheppokat, and N. Claussen, Effect of Processing Parameters on Phase and Microstructure Evolution in RBAO Ceramics, *Journal of the American Ceramic Society,* **77**(10), (1994), 2509–2517,

[4] S. Wu, D. Holz, and N. Claussen, Mechanisms and Kinetics of Reaction–Bonded Aluminum Oxide Ceramics, *Journal of the American Ceramic Society,* **76**(4), April 1993, 970–980.

[5] S.P. Gaus, M.P. Harmer, H.M. Chan, and H.S. Caram, Controlled Firing of Reaction-Bonded Aluminum Oxide (RBAO) Ceramics: Part I, Continuum Model Predictions, *Journal of the American Ceramic Society,* **82**(4), April 1999, 897–908,

[6] W.E. Schiesser, *The Numerical Method of Lines: Integration of Partial Differential Equations,* Academic Press, San Diego, 1991.

[7] A.C. Hindmarsh, `LSODE` and `LSODI`, Two Initial Value Ordinary Differential Equation Solvers, *ACM — Signum Newsletter,* **15**(5), (1980), 10–11.

[8] S.C. Eisenstat, M.C. Gursky, M.H. Schultz, and A.H. Sherman, Yale Sparse Matrix Package: I. The Symmetric Codes, Technical Report 112, Department of Computer Sciences, Yale University, 1977.

[9] S.C. Eisenstat, M.C. Gurskey, M.H. Schultz, and A.H. Sherman, Yale Sparse Matrix Package: II. The Nonsymmetric Codes, Technical Report 114, Department of Computer Sciences, Yale University, 1977.

[10] W.E. Schiesser, *DSS/2 (Differential Systems Simulator, version 2.) An Introduction to the Numerical Method of Lines Integration of Partial Differential Equations,* Lehigh University, Bethlehem, PA, 1977.

[11] P. Saucez, A. Vande Wouwer, and W.E. Schiesser, Some Observations on a Static Spatial Remeshing Method Based on Equidistribution Principles, *Journal of Computation Physics,* **128**, (1996), 274–288.

[12] B. Fornberg, Generation of Finite Difference Formulas on Arbitrarily Spaced Grid, *Mathematics of Computation,* **51**(184), (1988), 699–706.

[13] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems,* Springer Series in Computation Mathematics 14, Springer-Verlag, Berlin, 1991.

Chapter 13

Method of Lines within the Simulation Environment DIVA *for Chemical Processes*

R. Köhler, K.D. Mohl, H. Schramm, M. Zeitz,
A. Kienle, M. Mangold, E. Stein, and E.D. Gilles

13.1 Introduction

In chemical engineering, detailed process modeling, simulation, nonlinear analysis, and optimization of single process units as well as integrated production plants are issues of growing importance. As a software tool addressing these issues, the simulation environment DIVA [17, 26] is introduced. DIVA comprises tools for process modeling, preprocessing, and code generation of simulation models. Furthermore, its simulation kernel contains several advanced methods for simulation, parameter continuation, and dynamic optimization which can be applied to the same process model. These numerical methods require a model description in a form of differential algebraic equations (DAE). However, many models of chemical processes derived from first principles lead to partial differential equations (PDE) for distributed parameter systems, to integro partial differential equations (IPDE) for population balances of dispersed phases, and to DAE for lumped parameter systems.

In order to transform the PDE and IPDE models into the required DAE model formulation, DIVA employs the "method-of-lines" (MOL) approach for one space coordinate. The wide variety of distributed parameter models of chemical processes requires on one hand conventional discretization methods like, e.g., finite-difference schemes, and on the other hand more sophisticated methods to obtain reliable results in an acceptable computation time. One common feature of all advanced methods is the use of some type of adaptive strategy. In moving-grid methods, the adaptation concerns the positions of the grid points in the discretized spatial domain. Another approach of so-called high-resolution methods uses adaptive approximation polynomials. High-resolution methods are developed for hyperbolic conservation laws with steep moving fronts. Examples are essentially non-oscillatory (ENO) schemes [33] or the robust upwind κ-interpolation scheme [16].

In order to support the user of the simulation environment DIVA, the symbolic preprocessing tool SYPPROT for MOL discretization is developed by means of the

computer-algebra-system MATHEMATICA. This tool provides different discretization schemes to transform PDE, IPDE, and the related boundary conditions on 1-dimensional spatial domains into DAE. Discretization options concern fixed spatial grids and moving grids based on an equidistribution principle [43]. The toolbox architecture of SYPPROT allows fast testing of various discretization schemes without redefinition of the model equations. This is regarded as an important feature to minimize the overall effort for modeling and simulation. A further characteristic that is key to the easy exchange of discretization schemes is the separation between model equations and MOL parameter definitions by means of the MATHEMATICA data structure (MDS). The resulting overall DAE are written in symbolic form as an input file for the DIVA code generator [15, 31], which automatically generates the DIVA simulation files representing a process unit model in the model library.

In the following section, the architecture of the simulation environment DIVA is presented. This architecture consists of four layers, i.e., the DIVA simulation kernel, the code generator, the symbolic preprocessing tool SYPPROT, and the process modeling tool PROMOT. The MOL discretization of PDE and IPDE as the main preprocessing feature is the focus of Section 13.3. Standard discretization schemes like finite-difference and finite-volume schemes as well as adaptive approaches like high-resolution methods and an equidistribution principle based moving grid method are explained for distributed parameter models with one space coordinate. The application of these discretization methods within the symbolic preprocessing tool SYPPROT follows in Section 13.4, where the PDE and IPDE model representation as well as the implemented MOL discretization capabilities are described. In Section 13.5, the utilization of SYPPROT and DIVA is illustrated by simulations of a circulation-loop-reactor model and a moving-bed chromatographic process model.

13.2 Architecture of the Simulation Environment DIVA

The DIVA architecture (Figure 13.1) comprises four layers that are all accessible for editing and debugging by the user. The first or bottom layer contains the DIVA simulation kernel with the numerical methods and the library of generic process unit models. The simulation kernel as well as the model representation are implemented in FORTRAN77. A process unit model consists of several FORTRAN subroutines collected in the model library and a data file for the parameters and initial values. A more detailed description of the structure of the DIVA simulation kernel, the linear implicit DAE model representation, and its numerical methods follows in the next subsections.

The second layer of DIVA consists of the code generator (CG) and the corresponding CG input file which uses the same standardized DAE representation as the simulation kernel itself. The code generator automatically generates the FORTRAN subroutines for a generic process unit model along with a data file for its parameterization, as required by DIVA.

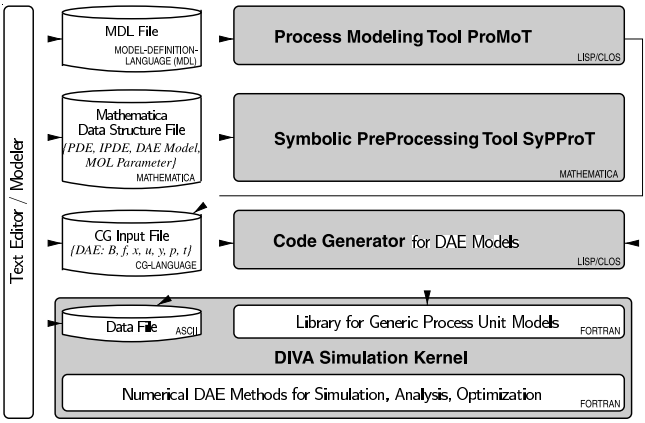


FIGURE 13.1
Architecture of the simulation environment Diva with the process modeling tool ProMoT, the symbolic preprocessing tool SyPProT, the code generator, and the Diva simulation kernel including the model library and the numerical DAE methods [41, 40, 15, 17, 26, 31].

The third layer represents the symbolic preprocessing tool SyPProT as well as the appropriate model definition file. The task of symbolic preprocessing concerns the transformation of models derived from first principles into the DAE representation of Diva. The input format for representation of mixed DAE, PDE, and IPDE models is the Mathematica data structure (MDS).

The top layer of Diva is the process modeling tool ProMoT.

13.2.1 The Diva Simulation Kernel

13.2.1.1 Structure of the Diva Simulation Kernel

The Diva simulation kernel is a numerical tool that has been developed at the University of Stuttgart over the last 15 years [11, 17, 10, 13, 26, 22]. It offers a wide variety of numerical methods for the treatment of both process design and process operation problems. The simulation kernel is organized in a modular structure as depicted in Figure 13.2. Within the kernel, the user has the possibility to switch interactively between numerical methods (second layer) using a command language, which is handled by the command interpreter (top layer). All numerical methods can be applied to the Diva plant model (third layer). This is possible due to the modular structure of the Diva simulation kernel and the uniform representation of the unit models (bottom layer, left). Using the plant flowsheet information, the individual unit models are combined by the plant model processor to form the overall plant model. The model representation will be discussed in more detail in the following section.

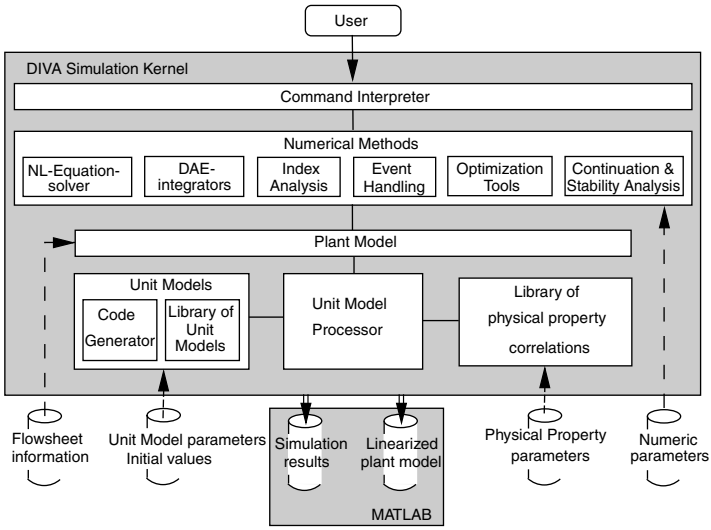


FIGURE 13.2
Structure of the Diva simulation kernel [17, 26].

13.2.1.2 DIVA Model Representation

Within a flowsheet simulation tool, a process plant is represented as a structure of interconnected process unit models. Therefore, each process unit has to be specified by a unit model. The plant model is then generated by connecting the unit models according to the flowsheet of the process. To make connecting unit models easy, all unit models have to be specified in a uniform way. In dynamic flowsheet simulation, the formulation of unit models as systems of ordinary differential and algebraic equations (DAE) is common practice for chemical engineering problems. For the simulation environment DIVA, the formulation as semiimplicit DAE with a differential index of one has been chosen [42]. The model formulation of a process unit k is given as follows:

$$\begin{aligned} B_k(x_k, u_k, p_k, t) \cdot \frac{dx_k}{dt} &= f_k(x_k, u_k, p_k, t) \quad t > t_0, x_k(t_0) = x_k^0 \,, \\ y_k &= H_k \cdot x_k \end{aligned} \tag{13.1}$$

with

process unit model index	k	$\in$	I	left side matrix	B_k	$\in$	$\mathrm{R}^{n\times n}$
states	x_k	$\in$	R^n	function vector	f_k	$\in$	R^n
inputs	u_k	$\in$	R^m	outputs	y_k	$\in$	R^r
parameters	p_k	$\in$	R^p	output matrix	H_k	$\in$	$\mathrm{R}^{r\times n}$
time	t	$\in$	R^1	initial values	x_k^0	$\in$	R^n

The connections among the unit models are specified in a flowsheet information file which contains a coupling matrix $C = \{C_{ij}\}$ that connects the internal outputs y_j of the process unit j to the internal inputs u_i of the process unit i according to

$$u_i = C_{ij} \cdot y_j, \qquad C_{ij} \in R^{m_i \times r_j} . \tag{13.2}$$

The plant model that is obtained by connecting the unit models is also a system of semi-implicit DAE. The numerical DAE methods of DIVA are based on this uniform DAE representation, into which PDE and IPDE have to be transformed by means of the MOL approach.

13.2.1.3 Sparse Numerical Methods within the DIVA Simulation Kernel

Models of process plants typically result in highly nonlinear large sparse systems of arbitrary structure. To keep computation times small, sparse matrix numerical techniques are used within DIVA. If sparse matrix techniques are used for the numerical solution of (13.1), the patterns of the Jacobians $\partial(B_k \cdot \dot{x}_k)/\partial x_k$, $\partial(B_k \cdot \dot{x_k})/\partial u_k$, $\partial f_k/\partial x_k$, $\partial f_k/\partial u_k$ have to be provided.

All numerical methods available in DIVA can be applied to the same plant model. The possibility of processing one plant model with different numerical methods strongly reduces model implementation efforts. This is particularly important because the modeling and model implementation steps are still the most time-consuming steps in computer-aided process engineering.

Besides numerical methods provided by DIVA, the export functions to MATLAB 5.3 (a commercial software tool for matrix-based systems analysis and synthesis) offer additional numerical and graphical capabilities. It is possible to generate a linearized plant model in MATLAB format that can be used to apply control analysis and design methods within MATLAB (Figure 13.2). Furthermore, MATLAB is used for the visualization of simulation results.

Steady-State and Dynamic Simulation

DIVA offers numerical methods for several different simulation tasks. For the solution of initial value problems, several DAE integrators are available: SDASSL [2] and SDASAC [3], using a backward differentiation method, LIMEXS using an extrapolation method [4] and RADAU5S [7], using Runge–Kutta methods. A special feature of SDASAC is the simultaneous calculation of parameter sensitivities (derivatives of model states with respect to model parameters), which are needed, e.g., for the solution of optimization and identification problems.

DIVA also provides different methods for solving systems of nonlinear algebraic equations which are needed for the determination of steady-state solutions and the consistent initialization of DAE integration. The solvers implemented in DIVA are a Levenberg-Marquardt algorithm [9] and a Newton method [28]. For the solution of steady-state equations there is also the possibility to calculate parameter sensitivities.

In addition to these standard numerical methods, DIVA provides routines for event handling. This includes explicit events caused by discrete input functions that are

triggered at given times, and implicit events which are caused, e.g., by bounded outputs of controllers and which depend on model states.

Continuation Methods for Nonlinear Analysis

Nonlinear effects have a strong influence on many chemical engineering processes, especially in reaction engineering. When operating conditions are varied, nonlinearities may cause sudden and often unexpected changes in the qualitative behavior of a process. Examples are transitions from a steady-state set point to a completely different steady state with undesired properties or to oscillatory behavior [39]. The knowledge of operating conditions under which such changes in the qualitative behavior occur are crucial for a safe and efficient operation of many chemical processes. Continuation methods in combination with stability analysis have proven to be efficient tools for determining those boundaries between regions of qualitatively different behavior. In DIVA, continuation algorithms are available for the application to all kinds of plant models. The methods comprise algorithms for the one-parameter continuation of steady states as well as periodic solutions and algorithms for the two-parameter continuation of saddle-node and Hopf bifurcations. The numerics have been tailored to systems of high dynamical order. Within reasonable computation time, they can perform a bifurcation analysis of differential algebraic systems from several hundred to a few thousand equations as they typically result from a spatial discretization by the MOL. In contrast, classical packages for bifurcation analysis are restricted to systems of a few ordinary differential equations or to systems with special structural properties. A detailed description of the numerical methods used in DIVA is beyond the scope of this chapter. The interested reader is referred to [13, 24]. In the following, only a brief idea of the capabilities of the methods will be given. In general, a continuation algorithm can be used to trace the solution curve of an under-determined system of algebraic equations

$$g(\eta) = 0, \quad g \in \mathrm{R}^m, \quad \eta \in \mathrm{R}^{m+1} \tag{13.3}$$

in an $(m + 1)$-dimensional space. For that purpose a predictor-corrector algorithm with local parameterization and step-size control is used in DIVA. This algorithm is able to cope with turning points, where the solution curve changes its direction. A simple application of continuation methods is the computation of steady-state solutions as a function of some distinguished model parameter p. In this case, g is the right-hand side vector f of the model equations, and η consists of the state vector x and the parameter p. The result of such a one-parameter continuation of steady states can be visualized in a bifurcation diagram as shown in Figure 13.3. In this example, the steady-state solutions form an S-shaped curve. A co-existence of three steady states is found for certain values of the bifurcation parameter F. An eigenvalue monitor is used to determine the stability of the computed steady states and to detect singularities, where one or several eigenvalues cross the imaginary axis. In Figure 13.3, two types of steady-state singularities are found. The first type is a so-called saddle-node bifurcation where two steady-state solutions coincide. The second type is a Hopf bifurcation where a steady-state solution loses its stability and

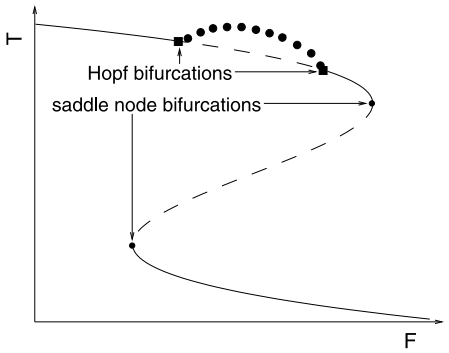


FIGURE 13.3
Bifurcation diagram of a CSTR model: dependence of reactor temperature T on reactant flow F [25].

gives rise to a branch of periodic solutions. In the diagram, the periodic solutions are symbolized by the maximum temperature during a periodic cycle. From bifurcation theory, necessary and sufficient conditions for saddle-node bifurcations and Hopf bifurcation can be derived. Together with the steady-state equations of the model, they form augmented equation systems for the direct computation of the state vector x and the parameter p at the singular points. Those augmented equation systems are generated automatically by DIVA. In the framework of the continuation algorithm, they are used to trace the curves of singular points in two parameters. The resulting curves form the boundary of the regions of qualitatively different behavior in the parameter space. The predictor corrector algorithm in DIVA is not only used for the computation of steady-state solutions but also for the continuation of stable and unstable periodic solutions in one parameter. For that purpose, the continuation algorithm is combined with a shooting method adapted to the special demands of high-order systems. For details, see [24].

Parameter Estimation and Optimization

Every model of chemical engineering processes relies on parameters connecting mathematics with reality. Two different problems arise in this context. The *parameter estimation problem* refers to the determination of unknown model parameters from measurement data. The improvement of process behavior results in the *parameter optimization problem.* Both of these problems can be solved by DIVA. In any case it is crucial to compute *parameter sensitivities* $W_{i,j} = \partial x_i / \partial p_j$, i.e., partial derivatives of state variables x with respect to parameters p. This information is obtained by differentiating the model equations (13.1) with respect to the parameters p yielding the equations of variations

$$B \frac{dW}{dt} = \frac{\partial}{\partial x}\left(f - B\frac{dx}{dt}\right) \cdot W + \frac{\partial}{\partial p}\left(f - B\frac{dx}{dt}\right) . \qquad (13.4)$$

The DAEs (13.1) and (13.4) are solved simultaneously in DIVA with the integrator SDASAC [3].

The parameter estimation problem is formulated as a least-squares (LS) problem where the squares of the differences between measured and calculated output variables at discrete points of time are minimized. In DIVA, this problem is solved by the sequential quadratic programming (SQP) algorithm E04UPF from the NAG library [27] which uses sensitivities calculated according to Equation (13.4). Parameter estimation for a steady-state model is also possible. In this case, the sensitivities $W_{i,j}$ are calculated by finite differences with one of the solvers mentioned above. Normally, only a subset of the desired set of parameters can be estimated at the same time with a defined accuracy. This subset is determined in DIVA by a *parameter analysis* based on the calculation of output sensitivities $\partial y/\partial p$ [22]. Combining this parameter analysis with parameter estimation provides a powerful tool to determine unknown model parameters from measurement data.

The optimization of processes and plants is an important engineering task in the design stage as well as in the retrofit of existing plants. Mathematically, this leads to a *parameter optimization problem* where an objective function has to be minimized

$$\min_{p,u} \Phi = \int_{t_{\text{start}}}^{t_{\text{end}}} g(p, u(t), x(t), t)\, dt + h(p, u(t_{\text{end}}), x(t_{\text{end}}), t_{\text{end}}) \quad (13.5)$$
$$p_{\min} \le p \le p_{\max}$$

with parameters p, input variables u, and state variables x. The criterion g is calculated over the considered time horizon $t \in [t_{\text{start}}, t_{\text{end}}]$, whereas h considers only the final point t_{end}. In the case of a steady-state optimization problem, g equals zero and only h is to be minimized. For a dynamic optimization problem the input trajectories $u(t)$ are discretized in order to obtain a parameter optimization problem where only time-independent values of parameters have to be determined. As an illustrating example, the dynamic optimization of coupled distillation columns is given in [37]. This class of problems is solved in DIVA with the SQP routine E04UCF from the NAG library. The corresponding model equations and sensitivity equations are solved with the integrator SDASAC in the case of dynamic optimization or with one of the nonlinear equation solvers in the steady-state case. The influence of parameters on the optimal solution is quite different. To choose the most promising parameters as optimization variables, a *sensitivity analysis* can be applied. Like in the parameter estimation problem, the combination of analysis and optimization provides a powerful tool to determine parameters and input trajectories in order to improve the process performance.

Due to the utilization of sparse matrix techniques and standard numerical routines like Harwell [9] and BLAS, DIVA is suitable for large-scale systems with as much as 10.000 state variables and with over 100.000 Jacobian entries. Systems of this scale have been realized in DIVA with reasonable computation times.

13.2.2 Code Generation of DIVA Simulation Models

For an efficient and user-friendly implementation of process unit models, the code generator (CG) and a CG language have been developed [15, 31]. The syntax of the CG language is similar to the programming language LISP [36] which is used for the implementation of the CG. As depicted in Figure 13.1, the CG input file serves as an interface for the symbolic preprocessing tool, the process modeling tool, as well as for the user. The CG language provides powerful definition elements for the automatic generation of compact and efficient FORTRAN code by the code generator. The two-step implementation of simulation models by the symbolic model definition with the CG language and its transformation into FORTRAN code has several advantages. On one hand, the implementation of model equations in the CG language is much more convenient to the user than the direct coding in FORTRAN. For example, the coding of the Jacobians' patterns required by the sparse matrix numerics of DIVA is automatized by the code generator. Doing this manually would be a cumbersome and time consuming task. On the other hand, the executable FORTRAN program including the automatically generated simulation models leads to better computational performance than interpretation of model equations during runtime.

13.2.3 Symbolic Preprocessing Tool

The capabilities of the symbolic preprocessing tool SYPPROT allow transformation of process models derived from first principles into linear implicit DAE models required by DIVA. The functionalities concern MOL discretization of PDE and IPDE into DAE, index analysis and reduction of DAE, as well as DAE transformation into the linear implicit form (13.1). SYPPROT is implemented with the computer-algebra-system MATHEMATICA and it is designed as a toolbox that contains several preprocessing modules. The architecture of the preprocessing tool is depicted in Figure 13.4. On the left, it shows two files for symbolic preprocessing model representation using the MATHEMATICA data structure (MDS) and a CG input file, which is the interface to the DIVA simulation kernel.

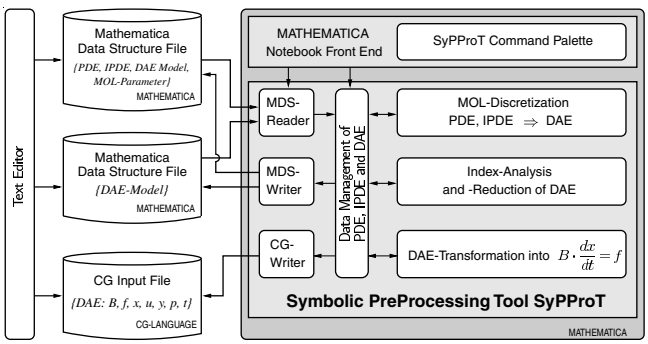


FIGURE 13.4
Architecture of the symbolic preprocessing tool SyPPProT [14].

The symbolic preprocessing tool SYPPROT is depicted on the right as part of the computer-algebra-system MATHEMATICA. SYPPROT contains three preprocessing modules for MOL discretization of PDE and IPDE, for index-analysis and -reduction of DAE, and for DAE transformation into the linear implicit form (13.1). Furthermore, there are interface modules for the interpretation of MDS objects (MDS-Reader) and for file output (MDS-Writer, CG-Writer). The MDS model definition is interpreted by the MDS-Reader. The MDS-Writer generates an MDS file of a selected model definition stored in the data management. The CG-Writer is used to translate the preprocessing result of linear implicit DAE models from MDS into CG-language. The interface and the preprocessing modules are interconnected by the data management. It calls the preprocessing modules and handles the model information of all preprocessing steps.

MDS models can be defined as a text file or interactively within the MATHEMATICA notebook front end which is shown at the top within the MATHEMATICA box. The notebook front end contains as an extension of the MATHEMATICA standard palettes the SYPPROT command palette. This palette allows a convenient generation of MDS objects as well as the execution of the preprocessing commands. By means of the notebook front end, complete preprocessing sessions as well as additional symbolic manipulations using the MATHEMATICA capabilities can be performed and saved.

13.2.4 Computer-Aided Process Modeling

The knowledge-based process modeling tool PROMOT supports modelers in developing DIVA models on the phase level of a process unit [41, 40]. The object-oriented knowledge base of PROMOT contains modeling entities for the representation of the structure of process unit models, the model equations, and the occurring material substances and mixtures. These structural, behavioral, and material model entities are defined with the model definition language MDL of PROMOT. A process model is represented as MDL frames in an MDL-file which can be generated using a text editor or the graphical MDL-editor. PROMOT is implemented in LISP and its object-oriented extension CLOS. This allows PROMOT direct access to the code generator commands. With the CG-Writer of PROMOT, a CG input file can be generated. Future research for PROMOT will include the extension of the language concepts and constructs of MDL for an implementation of PDE and the connection of PROMOT and the preprocessing tool SYPPROT.

13.3 MOL Discretization of PDE and IPDE

Typical equations for distributed parameter models of chemical processes are coupled PDE and IPDE with one space or property coordinate z for spatial distributed models or population models, respectively. The PDE and IPDE are up to second

order in the space or property coordinate z and of first order in time t. The initial conditions (IC) are specified by the profiles $x_0(z)$.

$$A\frac{\partial x}{\partial t} = C\frac{\partial^2 x}{\partial z^2} + \frac{\partial F(x, u, p, z, t)}{\partial z} + S(x, u, p, z, t) \quad t > t_0, \quad z \in (0, l) \tag{13.6}$$

$$x(z, t_0) = x_0(z) \quad z \in [0, l]\,. \tag{13.7}$$

The related boundary conditions (BC) depend on the input vectors $v_{0,l}(t)$:

$$0 = C_{0,l}\frac{\partial x}{\partial z} + F_{0,l}(x, v_{0,l}, p, t) \quad t > t_0, \quad z \in \{0, l\}\,. \tag{13.8}$$

The matrices $A(x, u, p, z, t)$ and $C(x, u, p, z, t)$ as well as the source vector $S(x, u, p, z, t)$ may depend on the state variables $x(z, t) \in \mathrm{R}^n$, the input variables $u(z, t)$, the parameters p, and on z and t. The flux function $F(x, u, p, z, t)$ represents the convective transport with the flow velocity $w(x, u, p, z, t)$ according to:

$$F(x, u, p, z, t) = w(x, u, p, z, t)x(z, t)\,. \tag{13.9}$$

In case of population models [30], the source vector S consists of non-integral terms, collected in the function vector $R(x, u, p, z, t)$ and integral terms concerning the function vector $Q(x, u, p, z, z', t)$.

$$S(x, u, p, z, t) = R(x, u, p, z, t) + \int_{z_{\mathrm{MIN}}}^{z_{\mathrm{MAX}}} Q(x, u, p, z, z', t)\, dz'\,. \tag{13.10}$$

The function vector R represents the sources and sinks only depending on the current value of the population coordinate z. The remote effects of the population also concerning interactions of several particles are described by the integral over the function vector Q. The integral limits z_{MIN} and z_{MAX} depend on the related population effects and are either the population domain boundary values 0 or l, the coordinate z or a function of this coordinate $f(z) \in [0, l]$.

These general forms of the PDE, IPDE, and BCs which describe hyperbolic as well as parabolic problems are handled by the symbolic preprocessing tool SyPPRoT. It is assumed that the model equations are well posed and have a unique solution.

For the discretization of PDE and IPDE models (13.6) through (13.8), the MOL approach discretizes the domain of the independent variable z for the space or property coordinate. The continuous coordinate z is replaced by discrete grid points z_k or $z_k(t)$, $k = 1(1)k_{\max}$ for static or moving grids, respectively, which is illustrated for a static grid in Figure 13.5. The grid partition also includes the segmentation by means of continuous control volumes (CV). In this case, the grid points represent the cell center points of the CVs. On the discretized domain of z, the partial derivatives $\partial^i x(z, t)/\partial z^i$, $i = 1, 2$ are approximated by functions of the state variables $x_k(t)$ at the grid points z_k. For these approximations, SyPPRoT provides configurable finite-difference and

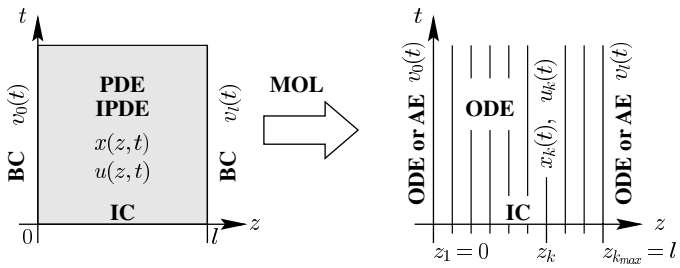


FIGURE 13.5
Method-of-Lines (MOL) approach for the transformation of partial differential equation (PDE), integro partial differential equation (IPDE), and the related boundary conditions (BC) into ordinary differential and algebraic equations (ODE,AE) [32].

finite-volume schemes. The result of the MOL discretization are ODEs at the inner grid points $z_k, k = 2(1)k_{\max} - 1$ and algebraic equations (AE) or ordinary differential equations (ODE) at the boundary grid points z_k, $k = 1, k_{\max}$ for finite-difference or finite-volume schemes, respectively. Furthermore, the ICs $x(z, 0)$ are transformed into $x_k(0)$, $k = 1(1)k_{\max}$, which are consistent with the AEs if the ICs (13.7) and BCs (13.8) are formulated consistently.

In the next subsections, the discretization schemes available in SYPPROT are described. First the standard MOL discretization schemes of finite differences (FD) and finite volumes (FV) are presented. However, these FD and FV schemes have some disadvantages when applied to problems with steep moving fronts. High-order approximations lead to unphysical oscillatory behavior of the solution. Low-order schemes do not show such oscillations but require a very fine grid for sufficiently accurate solutions. Due to the high number of grid points, the computational effort is quite large.

There are two approaches to combine the objectives of high numerical accuracy as well as efficient computation: so-called *high-resolution schemes* [8] and *moving-grid methods* [5, 6]. High-resolution schemes use high-order approximations based on a suitable choice or weighting of the approximation points to get an oscillation-free scheme. The adaptivity concerns the approximation polynomials. Examples are essentially-non-oscillatory (ENO) and flux-limiter schemes [33, 34, 38, 16] which are also described in a following subsection.

Moving grid methods use variable grid nodes to concentrate these nodes in solution sections with steep fronts. The grid nodes are placed where they are most needed to keep the discretization error small. The adaptivity of this approach concerns the flexible distribution of the available number of grid nodes. Furthermore, moving grid methods are distinguished into dynamic and static regridding methods. Dynamic regridding methods operate on continuously moving grid nodes. The number of state variables and equations is increased by the number of moving grid points. Static regridding methods compute the solution for a certain number of time steps on a static

grid. Then the regridding step calculates new positions of grid points, whereas the insertion and elimination of grid points is possible. After each such static regridding step, an interpolation is necessary to transform the solution on the old grid to a solution on the new grid. The interpolation introduces additional numerical errors. Moreover, in order to continue the simulation, consistent initial values must be computed. Of course both regridding techniques can be combined.

In the context of automatic discretization with the symbolic preprocessing tool of the simulation environment DIVA, the application of static regridding methods seems to be less suitable considering the static storage management of the FORTRAN-based DIVA simulation kernel, which prohibits a change in the number of state variables and equations during simulation. A further disadvantage is the computational cost of frequent initialization of the model equations required after each static regridding step, especially for multistep integration procedures. For the symbolic preprocessing tool SYPPROT, we focus on a dynamic regridding method based on the Lagrangian approach [43] which is comparably easy and flexibly implemented within the software architecture of SYPPROT. This method is described in the last part of this section.

13.3.1 Finite-Difference Schemes

The finite-difference (FD) schemes used in the symbolic preprocessing tool are based on Lagrange polynomials and can be applied on uniform or non-uniform grids for the discretized space coordinate. The spatial grid is defined by a grid function $z(k)$ that leads to the grid points z_k, $k = 1(1)k_{\max}$. Then the state variables $x(z,t)$ are approximated by Lagrange polynomials according to

$$x(z,t) \approx L^{[r,s]}(z,t) = \sum_{j=r}^{s} l_j(z) x_j(t) \; . \tag{13.11}$$

This approximation uses the state variables $x_j(t)$, $j = r(1)s$ at the grid points between z_r and z_s (Figure 13.6). The polynomial (13.11) is used to approximately calculate the spatial derivatives of $x(z,t)$ at the center point z_k by differentiating with respect to z:

$$\left.\frac{\partial^i x(z,t)}{\partial z^i}\right|_{z=z_k} \approx \left.\frac{\partial^i L^{[r_i,s_i]}(z,t)}{\partial z^i}\right|_{z=z_k} \; . \tag{13.12}$$

For the approximation of an integral occurring in IPDE problems (13.10), the integral range is split into subranges for each grid point $i = i_{\mathrm{MIN}}(1)i_{\mathrm{MAX}}$ with $z_i^- = z_{\mathrm{MIN}}$ for $i = i_{\mathrm{MIN}}$ and $z_i^+ = z_{\mathrm{MAX}}$ for $i = i_{\mathrm{MAX}}$. For the subrange integrals, the state variables are approximated by Lagrange polynomials according to (13.11):

$$\int_{z_{\mathrm{MIN}}}^{z_{\mathrm{MAX}}} Q(x,u,p,z',z,t)dz' = \sum_{i=i_{\mathrm{MIN}}}^{i_{\mathrm{MAX}}} \int_{z_i^-}^{z_i^+} Q(L^{[r_i,s_i]}(z',t),u,p,z',z,t)\, dz \; . \tag{13.13}$$

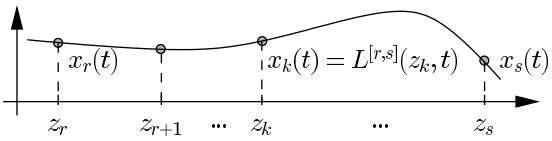


FIGURE 13.6
Lagrange polynomial (13.11) for the approximation of $x(z, t)$ in the range $z_r < z < z_s$.

This leads to the discretization parameters r_i, s_i, and k for the order of the spatial derivative $i = 0, 1, 2$. For the symbolic preprocessing, these discretization parameters can be individually defined for each PDE and IPDE. The approximations at the boundary points require a special treatment. At the boundary points, the approximation polynomials can depend on grid points outside the spatial domain. This is avoided by means of the so-called *sliding differences technique.* Thereby local spatial oscillations possibly appearing in numerical solutions of hyperbolic problems can be prevented by an order reduction of the approximation polynomials.

13.3.2 Finite-Volume Schemes

The finite-volume (FV) method subdivides the spatial domain into a finite number of discrete contiguous control volumes (CV) to which the PDE or IPDE (13.6) are applied. This leads to the integral form of the PDE or IPDE on which the FV method is based [29].

For the CV definition, a uniform or non-uniform grid function $z(k)$, $k = 1(1)k_{\max}$ is used. The grid function can either define the cell center points of the CVs or the cell boundary points. This leads to the two possible grid practices shown in Figure 13.7. For the grid practices named "GridPoints" and "VolumeBounds," the grid functions compute the cell center points and the cell boundaries, respectively.

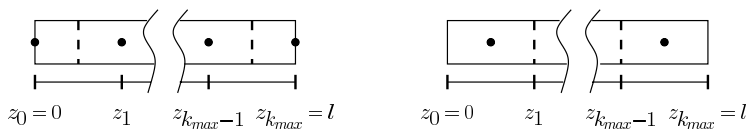


FIGURE 13.7
Definition of the control volumes (CV) for discretization by finite volumes and of the corresponding grid practices "GridPoints" (left) and "VolumeBounds" (right).

The FV discretization of a PDE is performed in two steps: the integration over each CV and the approximation of the resulting cell boundary values. In order to illustrate this procedure, a scalar convection diffusion PDE for $x(z, t)$, $z \in (0, l)$ is used with

the source term $S(x, u, p, z, t)$.

$$\frac{\partial x}{\partial t} = -v\frac{\partial x}{\partial z} + D\frac{\partial^2 x}{\partial z^2} + S \qquad t > 0, \quad z \in (0, l)\ . \tag{13.14}$$

The related BC and IC are not considered. The integration of PDE (13.14) over the kth CV from the cell boundaries z_k^- to z_k^+ (for notations see Figure 13.8) leads to

$$\left(z_k^+ - z_k^-\right)\frac{dx_k}{dt} = -v\left(x_k^+ - x_k^-\right) + D\left(\left.\frac{\partial x}{\partial z}\right|_{z_k^+} - \left.\frac{\partial x}{\partial z}\right|_{z_k^-}\right) + \left(z_k^+ - z_k^-\right)S_k \tag{13.15}$$

with

$$\begin{aligned} S_k = S\left(x_k, u_k, p, z_k, t\right) &= R_k + \sum_{i=i_{\mathrm{MIN}}}^{i_{\mathrm{MAX}}} \int_{z_i^-}^{z_i^+} Q\left(x, u, p, z', z, t\right) dz' \qquad (13.16)\\ &= R_k + \sum_{i=i_{\mathrm{MIN}}}^{i_{\mathrm{MAX}}} \left(z_i^+ - z_i^-\right) Q\left(x_k, u_k, p, z_i, z_k, t\right)\ . \end{aligned}$$

The discretized source S_k is interpreted as a representative mean value for the CV k.

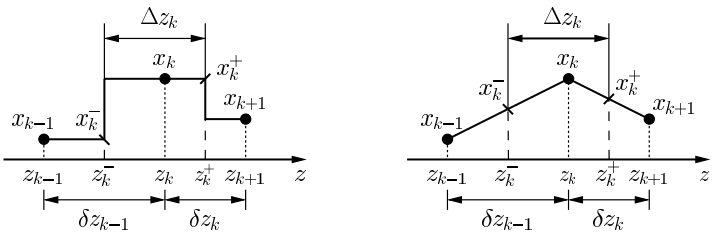


FIGURE 13.8
Profile assumptions of *piecewise constant (upwind)* (left) and *piecewise linear* schemes (right) for the approximation of $x_k^\pm$ and the derivatives $\partial x/\partial z|_{z_k^\pm}$, respectively.

The integral term included in (13.17) is discretized by a sum of subintegrals. These subintegrals represent the involved CVs according to the complete integral range in (13.10) with $z_i^- = z_{\mathrm{MIN}}$ for $i = i_{\mathrm{MIN}}$ and $z_i^+ = z_{\mathrm{MAX}}$ for $i = i_{\mathrm{MAX}}$.

In the integral form of PDE (13.15), the values of $x_k^\pm$ and $\partial x/\partial z|_{z_k^\pm}$ are unknown and must be approximated. For the boundary CVs, the boundary conditions are inserted into (13.15) to eliminate unknown values at the inlet and the outlet of the spatial domain. For the approximation of the remaining unknown values at the interior cell boundaries z_k^- and z_k^+, so-called profile assumptions are used as shown in Figure 13.8. The following ODE for the kth CV is obtained applying these profile assumptions

to (13.15).

$$\frac{dx_k}{dt}\Delta z_k = -v\,(x_k - x_{k-1}) + D\left(\frac{x_{k+1} - x_k}{\delta z_k} - \frac{x_k - x_{k-1}}{\delta z_{k-1}}\right) + S_k \Delta z_k\,. \quad (13.17)$$

Here, the symbols Δz_k, δz_k, and δz_{k-1} are the distances of the CV boundary points and the CV center points, respectively.

For IPDE of population balances, the discretization of an integral within the source term S introduces further difficulties which concern the moment conservation of population distributions. More details about this topic as well as a solution strategy are described in [18].

13.3.3 High-Resolution Schemes

The concept of high-resolution schemes is based on approximation polynomials of high order with variable selection or weighting of the approximation points. This adaptive idea allows combination of high numerical accuracy and stability. High-resolution schemes are derived for hyperbolic conservation laws, which can be written in the following form using the PDE notations (13.6):

$$A\frac{\partial x}{\partial t} = \frac{\partial F(x, u, p, z, t)}{\partial z} + S(x, u, p, z, t) \qquad t > 0, \quad z \in (0, l)\,. \quad (13.18)$$

High-resolution schemes are mostly used in the context of finite-volume discretization which integrates (13.18) over the CV k according to

$$\int_{z_k^-}^{z_k^+} A\frac{dx}{dt}\,dz = \left(F_k^+ - F_k^-\right) + \int_{z_k^-}^{z_k^+} S\,dz \quad (13.19)$$

with the flux functions $F_k^{\pm} = F(x, u, p, z, t)|_{z=z_k^{\pm}}$ at the cell faces of the CV, see Figure 13.8. The focus is directed on the approximation of these flux function values. This approximation can also be interpreted as a more sophisticated profile assumption for the flux function.

Two high-resolution schemes are briefly described: the ENO-Roe scheme [33, 34] and the so-called robust upwind scheme [16].

13.3.3.1 ENO-Roe Scheme

The ENO-Roe scheme [33, 34] uses the so-called ENO interpolation which approximates the flux values using a fixed number of interpolation points on solution-dependent positions. The adaptive aspect of this scheme is the detection of the "optimal" position of the interpolation points. This position is determined successively by the construction of a Newton interpolation formula of increasing order for approximation of the flux. This approximation procedure compares divided differences of the Newton interpolation formula built of adjacent interpolation points. The minimal magnitude of the compared divided differences determines the position of the next

interpolation point. ENO schemes can be extended to any approximation order. But for schemes higher than third order, numerical stability problems increase [44]. An important advantage of the adaptive determination of the approximation points is that a reversal of the flux direction is automatically considered.

13.3.3.2 Robust Upwind Scheme

The robust upwind scheme [16] is a κ-interpolation scheme originated by Van Leer [19] with a modified flux limiter of Sweby [38]. Its accuracy is of 2nd to 3rd order. The robust upwind approximation of F_k^+ uses flux function values $F_j = F(x_j, u_j, p, z_j, t)$ of the upwind located CV centers $j = k, k-1$.

$$F_k^+ = F_k + 0.5\phi\left(r_k^+\right)(F_k - F_{k-1}) \;, \tag{13.20}$$

$$r_k^+ = \frac{F_{k+1} - F_k + \epsilon}{F_k - F_{k-1} + \epsilon} \;, \tag{13.21}$$

$$\phi(r) = \max\left(0, \min\left(2r, \min\left(\frac{1}{3} + \frac{2}{3}r, 2\right)\right)\right) \;. \tag{13.22}$$

The limiter function ϕ depends on the ratio of consecutive solution gradients r_k^+. The parameter ϵ is very small to avoid division by zero in uniform flow regions.

In contrast to the ENO scheme, the approximation points z_k are fixed. The adaptation lies in the solution dependent weighting of the approximation points by the limiter function. The described high-resolution schemes can be easily integrated into the MOL module of SYPPROT, as shown in Section 13.4.2.

13.3.4 Equidistribution Principle Based Moving Grid Method

The considered moving grid method within the symbolic preprocessing tool uses the Lagrangian approach of grid nodes, which are continuously moved along with the solution. The grid consists of moving grid nodes $z_k(t)$, $k = 2(1)k_{\max} - 1$ for the inner spatial domain and fix grid nodes $z_k(t)$, $k = 1, k_{\max}$ at the domain boundaries. This approach reduces the rapid changes of steep front solutions at fixed grid points leading to small time steps during numerical simulation. Thus, the additional effort of the moving grid discretization is partly compensated.

Using the Lagrangian form of the time derivative at the grid node $z_k(t)$

$$\left.\frac{\partial x}{\partial t}\right|_{z_k(t)} = \frac{dx_k}{dt} - \left.\frac{\partial x}{\partial z}\frac{dz}{dt}\right|_{z_k(t)} \;, \tag{13.23}$$

the semi-discrete form of Equation (13.6) at the grid node $z_k(t)$ can be written as

$$A\left(\frac{dx_k}{dt} - \left.\frac{\partial x}{\partial z}\frac{dz}{dt}\right|_{z_k(t)}\right) = C\left.\frac{\partial^2 x}{\partial z^2}\right|_{z_k(t)} + \left.\frac{\partial F(x, u, p, z, t)}{\partial z}\right|_{z_k(t)} + S(x_k, u_k, p, z_k, t) \quad t > 0 \;. \tag{13.24}$$

According to this, no transformation is necessary for the BC (13.8). Additional equations are required for the computation of moving grid nodes $z_k(t), k = 2(1)k_{\max} - 1$. The general form of the moving grid equations reads

$$\tau E \frac{dz}{dt} = G(x, z, \kappa) \tag{13.25}$$

with the temporal regularization parameter τ, the matrix $E(x, z, \kappa)$, and the function vector $G(x, z, \kappa)$. The parameter κ is used to control the grid expansion such that

$$\frac{\kappa}{\kappa + 1} \leq \frac{z_i - z_{i-1}}{z_{i+1} - z_i} \leq \frac{\kappa + 1}{\kappa} . \tag{13.26}$$

The matrix E as well as the function vector G are based on the spatial equidistribution equations

$$\int_{z_k}^{z_{k+1}} M(z, x)dz = \int_{z_{k-1}}^{z_k} M(z, x)dz \qquad k = 2(1)k_{\max} - 1 \tag{13.27}$$

with the arc-length monitor function $M(z, x)$ considering the spatial gradients of the state variables $x \in R^n$

$$M(z, x) = \sqrt{1 + \frac{1}{n} \sum_{i=1}^{n} \left(\frac{\partial x_i}{\partial z} \right)^2} . \tag{13.28}$$

The elements of E and G depend on temporal as well as spatial smoothing techniques. For the considered implementation within the symbolic preprocessing tool, the smoothing procedures following Dorfi and Drury [5] are taken. A more detailed discussion of this smoothing technique can be found in [43].

The discretization of Equations (13.27) and (13.28) depends on the chosen discretization of the spatial domain according to finite differences or finite volumes as well as on stability criteria [21].

13.4 Symbolic Preprocessing for MOL Discretization

The preprocessing procedure of SYPPROT for automatic MOL discretization of distributed parameter models is performed in three steps. The first step is the MOL discretization of PDE and IPDE into DAE by means of a corresponding MATHEMATICA module of the preprocessing tool, see Figure 13.4. The available discretization methods of finite differences and finite volumes as well as the equidistribution principle based moving grid method have been described in the previous section. In a second step, the resulting DAE are transformed by another preprocessing module into the linear implicit model form (13.1) required by DIVA and the CG. Finally, the CG-Writer generates the CG input file (Figure 13.4).

For application of the automatic MOL discretization by the symbolic preprocessing tool, the model as well as required MOL parameters must be defined. The MATHEMATICA data structure (MDS) is a tailor-made input format for this purpose. This section describes the definition of PDE and IPDE models as well as that of MOL parameters by means of the MDS. Furthermore, the preprocessing module for MOL discretization is explained. Therefore, the internal procedure for application of the implemented discretization schemes is presented.

13.4.1 MATHEMATICA Data Structure

For the definition of mixed PDE, IPDE, and DAE models and the related MOL parameters, the symbolic preprocessing requires appropriate MATHEMATICA data structure (MDS) objects containing the complete model information (Figure 13.4). The MDS is based on the computer-algebra-system MATHEMATICA and uses (as MATHEMATICA itself) only a small number of symbolic programming methods. Mainly the methods of *expressions, rules,* and *lists* build the framework for the MDS. Furthermore, the MDS and the according MDS input file are based on the CG language and the CG input file, respectively, to facilitate the final transformation of a preprocessed PDE and IPDE model into a CG input file. The MDS provides several functions for the definition of mixed PDE, IPDE, and DAE models as well as MOL parameters. Each MDS definition function builds a section within the MDS input file, which contains the whole information about the model and the related MOL parameters. The MDS definition functions can also be executed directly in the notebook front end of MATHEMATICA forcing the MDS-Reader to generate the according MDS objects (Figure 13.4).

13.4.1.1 Definition of Mixed PDE, IPDE, and DAE Models

The definition functions for a PDE and IPDE model comprise a general model description and the definition of parameters, variables, and equations. The equation section `SystemEquations[...]` contains PDE, IPDE, and DAE in separate subsections but in a standardized symbolic representation. Special MDS functions are provided for the coupling of sequential and parallel neighboring spatial domains. Sequential domain coupling requires proper inner boundary conditions with consistent discretization schemes according to the PDE in the involved spatial domains. In order to obtain a common discretization scheme for a PDE or IPDE as well as the according boundary and inner boundary conditions, the definitions of the PDE or IPDE, the boundary, and inner boundary conditions are grouped together into a common section. In addition, this common section specifies common MOL parameters.

Coupling of parallel spatial domains requires interpolations between different grid point positions, if different individual spatial grids are used for the involved domains. The MDS provides so-called *shifted grids* for parallel coupled domains. A shifted grid is identical to the related basic grid with the only difference being an offset in the values of the grid points. Thereby, interpolations are not necessary. An additional advantage of a shifted grid is that it does not require additional own state variables and

corresponding new equations if a moving grid method is used. Due to the identical grid node distribution of the basic and the shifted grid, only the basic grid extends the number of state variables and equations. This reduces the complexity increase of the moving grid approach. The use of a shifted grid is demonstrated in Section 13.5.1 for the example of a circulation-loop-reactor model.

13.4.1.2 Definition of MOL Parameter

For the MOL parameters, two definition functions exist: the function `Domains[...]` defines the grid functions of the spatial domains and the function `Discretizations[...]` defines the discretization methods and its parameters.

The MDS definition function `Domains[...]` for a spatial grid contains arguments for the grid name, and the grid function as well as its parameterization. The grid function is a MATHEMATICA expression computing the grid node location for any valid index value. The grid function must be reversable to compute the index value for a given location within the spatial domain. This is required for graphical output or process unit coupling. The moving grid method is activated by the `MovingGrid` attribute. Its value is a list of moving grid parameters. All moving grid parameters are assigned default values which are used if the user defines an empty list for the moving grid parameters. But the user can also redefine these values to tune the moving grid method. The following moving grid parameters exist, see Section 13.3.4:

- `SpaceSmoothing` defines the spatial smoothing parameter κ (default value: $\kappa = 2$),
- `TimeSmoothing` defines the temporal smoothing parameter τ (default value: $\tau = 0$),
- `MonitorStates` collects the state variables of this domain to be considered for the monitor function; the default configuration considers all distributed state variables of the considered domain.

If the moving grid method is activated, the grid function computes only the initial distribution of the grid nodes.

The available discretization methods for the `Discretizations[...]` function are configurable FD and FV schemes, which are defined by the MDS functions `FDMethod[...]` and `FVMethod[...]`, respectively. The FD schemes define individual Lagrange polynomials for approximation of the spatial derivatives $\partial^i x(z,t)/\partial z^i$, $i = 0, 1, 2$. For the configuration of the Lagrange polynomials, the MDS provides three MOL parameters to determine the discretization parameters $r_i, s_i, k, i = 0, 1, 2$ (see Section 13.3.1):

- `PolynomPoints` m_i defines the numbers of interpolation points $m_i = s_i - r_i + 1$,
- `Eccentricity` q_i determines the approximation centers $k = (r_i + s_i)/2 + q_i$,
- `OrderReduction` o_i defines the order reduction of the approximation polynomials in the boundary area.

The FV scheme is defined by the MDS function `FVMethod[...]`, which is configured by the choice of profile assumptions for approximations of the unknown CV boundary values (see Section 13.3.2). The MDS provides thereto the parameter `Profile`.

Extracts of the MDS functions `SystemEquations[...]`, `Domains[...]`, and `Discretizations[...]` of the MDS input files for the circulation-loop-reactor model and the true-moving-bed process unit model are presented in Sections 13.5.1 and 13.5.2.

13.4.2 Procedure of the MOL Discretization

For a DIVA simulation of a PDE or IPDE model, three preprocessing steps have to be performed (see SYPPROT box in Figure 13.4 for corresponding preprocessing and interface modules): the MOL discretization of PDE and IPDE into DAE (top right), the transformation of the resulting DAE model into linear implicit form (13.1) (bottom right), and the generation of a CG input file of this DAE model (bottom left).

The user can execute these steps by two commands within a MATHEMATICA session. The command `AutomaticDiscretization[...]` performs the loading of an MDS input file and the execution of the discretization procedure. The result is a DAE model in MDS. This DAE model is transformed into form (13.1) and then written into a CG input file by the command `MDS2CG[...]`.

The internal procedure of the MOL discretization in the symbolic preprocessing tool is described in Figure 13.9. The gray boxes with solid lines mark different MATHEMATICA modules. The module for discretization management collects the information required for the discretization of a PDE or IPDE and the related BC as defined in the MDS input file. In particular, information on the grid function of the according spatial domain and the selected discretization method is supplied.

From the symbolic manipulation point of view, the moving grid method described in Section 13.3.4 can be interpreted as a preliminary model transformation before the actual discretization. This allows a comparably simple integration of this method in the discretization procedure of the preprocessing tool. The moving grid module applies Equation (13.23) to transform PDE and IPDE (13.6) into a form according to Equation (13.24). For the computation of the moving grid nodes, the additional moving grid equations (13.25) are required. These equations are automatically generated considering the grid expansion constraint (13.26), the spatial equidistribution principle (13.27) based on the arclength monitor (13.28), and temporal grid smoothing techniques. Then the transformed equations are discretized on the moving grid by FD or FV schemes, which are implemented in separate modules.

The FD and FV modules generate specific auxiliary variables for the generation of shorter and more efficient simulation models. Such auxiliary variables are used to compute, e.g., distances of grid nodes or lengths of CVs. The FD module approximates the state variable functions and their spatial derivatives by Lagrange polynomials (13.11) and their spatial derivatives (13.12), respectively. The FV module integrates the PDE or IPDE over each CV under consideration of the boundary con-

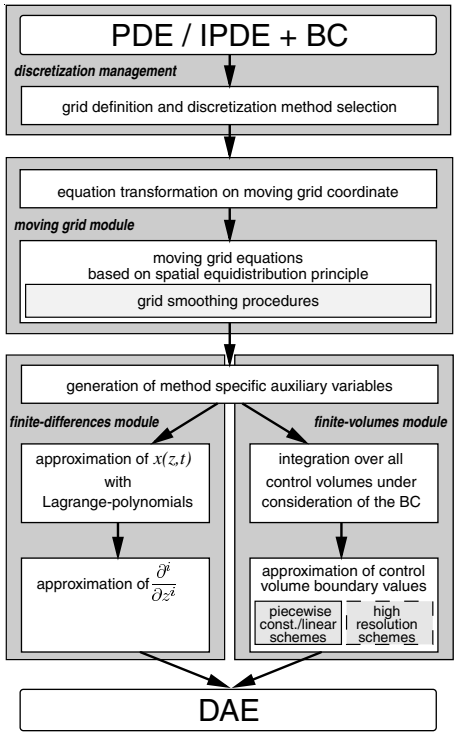


FIGURE 13.9
Procedure for use of the MOL discretization of partial differential equations (PDE), integro partial differential equations (IPDE), and boundary conditions (BC) in SyPProT.

ditions and approximates the resulting unknown values at the CV boundaries by piecewise constant or piecewise linear profile assumptions (Figure 13.8). The high-resolution schemes described in Section 13.3.3 could also be used to approximate the CV boundary values of the flux functions, see Section 13.3.3. These schemes are represented by the gray box with dashed lines in Figure 13.9, as they are not yet implemented in the finite-volume discretization module.

13.5 Application Examples

For illustration of the MOL capabilities of the symbolic preprocessing tool and the numerical methods in DIVA, two application examples from chemical engineering are presented. The first example is a circulation-loop reactor. The reactor consists of se-

quential and parallel coupled spatial domains. The model is described by PDE, which are coupled by inner boundary conditions for sequential neighboring domains and by the heat exchange between the parallel adjacent domains. The MOL discretization is performed with FD schemes applied on a moving grid. The second example is a moving-bed chromatographic process. The PDE describes the counter-current flow of two phases. FV schemes are used for automatic MOL discretization.

13.5.1 Circulation-Loop-Reactor Model

The circulation-loop reactor (CLR) [13, 23] represents a complex example for the application of the MOL. A scheme of the reactor is shown in Figure 13.10. A special feature of the reactor is the geometrical construction. The reactor consists of three spatial domains. Each domain is described by one dynamic PDE for the energy balance and two PDE for the material balances based on a quasisteady-state assumption. Besides the boundary conditions at the inlet and the outlet of the reactor, there are inner boundary conditions for the coupling of sequential neighboring domains. The dynamic behavior of the reactor shows moving fronts of different steepness, which require discretization schemes of adequate accuracy. In the following, the CLR model equations, extracts of the MDS input file for symbolic preprocessing of the PDE model, and the simulation results of DIVA are presented.

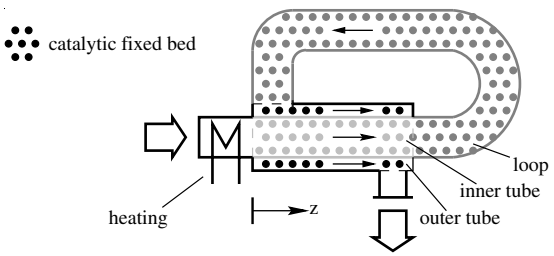


FIGURE 13.10
Circulation-loop reactor consisting of an inner tube, the reactor loop, and an outer tube; the inner and outer tube are built as a co-current heat exchanger [13, 23].

13.5.1.1 Model Equations

The CLR consists of three spatial domains comprising the inner tube, the loop, and the outer tube. A common coordinate z describes the 1-dimensional spatial domains:

$$\begin{aligned} &\text{Inner tube } (I) &&: z \in [0, l_{\text{tube}}] \\ &\text{Reactor loop } (L) &&: z \in [l_{\text{tube}}, l_{\text{loop}} + l_{\text{tube}}] \\ &\text{Outer tube } (O) &&: z \in [l_{\text{tube}}, l_{\text{loop}} + 2l_{\text{tube}}] \,. \end{aligned} \tag{13.29}$$

The reactor is modeled as a single-process unit. Its parts are structured as sequential and parallel adjacent spatial domains. The reactor modules are shown in Figure 13.11.

The loop domain is coupled with the tube sections by inner boundary conditions at $z = l_{\text{tube}}$ and $z = l_{\text{loop}} + l_{\text{tube}}$, respectively. The inner and outer tube domains are parallel adjacent domains and coupled by the heat flux over their interface. For the outer tube domain, a so-called *shifted grid* is used (see Section 13.4.1). This grid is shifted and uses the grid of the inner tube domain as a basic grid but with an offset of $z = l_{\text{loop}} + l_{\text{tube}}$.

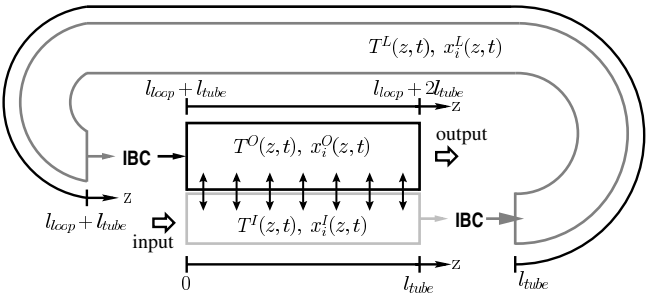


FIGURE 13.11
Scheme of the circulation-loop reactor with three spatial domains: inner tube (I), reactor loop (L), and outer tube (O). The domains are coupled by inner boundary conditions (IBC) and the heat exchange (↕).

The model equations of the CLR are formulated for every spatial domain $j \in \{I, L, O\}$ according to (13.29). The PDE for the temperatures $T^j(z,t)$ and the component mole fractions ${x_i}^j(z,t)$ are derived from energy and material balances as:

$$\begin{aligned}\left(\rho c_p\right)_s \frac{\partial T^j}{\partial t} &= -\left(\rho c_p\right)_g v^j \frac{\partial T^j}{\partial z} \\ &\quad + \lambda \frac{\partial^2 T^j}{\partial z^2} + \dot{Q}_{ex}^j + \sum_{i=1}^{2} \left(-\Delta h_{R,i}\right) r_i \left(x_i^j, T^j\right) \quad (13.30)\end{aligned}$$

$$\begin{aligned}0 &= -v^j \frac{\partial x_i^j}{\partial z} - \frac{M_g}{\rho_g} r_i \left(x_i^j, T^j\right) \\ &\qquad i = 1, 2\,, \quad j \in \{I, L, O\}\,. \quad (13.31)\end{aligned}$$

The expressions for the heat exchange rates $\dot{Q}_{ex}^j(z,t)$, the velocities v^j, and the reaction rates $r_i(x_i^j, T^j)$ can be found in [23]. The corresponding boundary conditions at the inlet at $z = 0$ are given by

$$\left(\rho c_p\right)_g v^I \left(T_{in} - T^I\Big|_{z=0}\right) + \lambda \left.\frac{\partial T^I}{\partial z}\right|_{z=0} = 0\,, \qquad x_{i,in} - x_i^I\Big|_{z=0} = 0 \quad (13.32)$$

and at the outlet of the reactor at $z = l_{\text{loop}} + 2l_{\text{tube}}$ by

$$\left.\frac{\partial T^O}{\partial z}\right|_{z=l_{\text{loop}}+2l_{\text{tube}}} = 0\,. \quad (13.33)$$

The inner boundary conditions at the transitions of the sequential neighboring domains ensure thermal and material equilibrium and the equality of the fluxes over the inner boundaries of the reactor model. Here, only the inner boundary conditions at $z = l_{\text{tube}}$ are presented:

$$\left(\rho c_p\right)_g v^I \left.T^I\right|_{z=l_{\text{tube}}} - \lambda \left.\frac{\partial T^I}{\partial z}\right|_{z=l_{\text{tube}}} = \left(\rho c_p\right)_g v^L \left.T^L\right|_{z=l_{\text{tube}}} - \lambda \left.\frac{\partial T^L}{\partial z}\right|_{z=l_{\text{tube}}} \tag{13.34}$$

$$\left.T^I\right|_{z=l_{\text{tube}}} = \left.T^L\right|_{z=l_{\text{tube}}}, \qquad \left.x_i^I\right|_{z=l_{\text{tube}}} = \left.x_i^L\right|_{z=l_{\text{tube}}}. \tag{13.35}$$

The model is completed by appropriate initial conditions $T^j(0, z)$ and $x_i^j(0, z)$.

13.5.1.2 Definition of Model Equations for Symbolic Preprocessing

For the definition of model equations for symbolic preprocessing, the MDS function `SystemEquations[...]` is used. The following extract shows the definition of the energy balance (13.30), the related boundary condition (13.32), and part of inner boundary condition (13.34).

```
1 SystemEquations[
2  Distributed[
3   Domain  -> "Tube[z]",
4   Comment -> "reaction zone of the inner and outer tubes"
5   Scope   -> { ...
6    Scalar[ rhocps D[Ti[z,t],t] == - rhocpg vi D[Ti[z,t],z]
7            + lam D[Ti[z,t],{z,2}] + Qpex[z,t]
8            - Summands[dhr[j] rreak[j][z,t],{j,1,NC}],
9     LowerBound     ->
10     0 == rhocpg vi (Tin[t] - Ti[z,t]) + lam D[Ti[z,t],z],
11    UpperBound     ->
12     Flux[TiOut[t]] == rhocpg vi Ti[z,t] - lam D[Ti[z,t],z],
13    Name           -> "PDEInnerTube",
14    Comment        -> "energy balance inner tube",
15    Discretization -> "FDOrder24"]
16   ... } ] ...];
```

Within the MDS expression `Distributed[...]` in lines 2–16, the equations related to a spatial distributed domain are defined. The domain itself is specified by the attribute `Domain` in line 3. The value of `Domain` is the name of a user defined spatial grid, here `"Tube[z]"`. The grid `"Tube[z]"` defines the grid points of the inner tube and also serves as the basic grid for the shifted grid of the outer tube. Therefore, one can simply add the equations of the outer tube to the attribute `Scope` in lines 5–16 of this `Distributed` object. This is indicated by the dots in line 16. The `Scalar[...]` expression in lines 6–15 represents a single PDE with attributes for relationships at the boundaries (`Lowerbound`, `UpperBound`) of the considered domain and the name of a MOL parameter definition (`Discretization`) in lines 9, 11, and 15, respectively. The first argument of a `Scalar[...]` expression is

the equation itself. In particular, in lines 6–8 the energy balance (13.30) for the inner tube is shown. Therein, the MATHEMATICA operator `D[...]` is used for temporal and spatial derivatives, e.g., `D[Ti[z,t],{z,2}]` which is the 2nd order spatial derivative of the state variable `Ti[z,t]`, that represents the temperature $T^I(z,t)$.

The attribute `UpperBound` defines the left part of the inner boundary condition (13.34). The MDS expression `Flux[...]` is used to define fluxes at the boundary of sequential neighboring domains. The object `Flux[TiOut[t]]` in line 12 defines the flux at the outlet of the inner tube which is identical with the flux at the inlet of the loop domain, the right part of Equation (13.34). For the inlet flux of the loop, another `Flux[...]` object is defined, which will be equated to `Flux[TiOut[t]]` in a separate MDS expression for coupling of sequential neighboring domains. The value of the attribute `Discretization` refers to MOL parameters named `"FDOrder24"`. These MOL parameters are used to discretize the PDE defined above as well as the relations defined in the attributes `LowerBound` and `UpperBound`, which are grouped together in a `Scalar[...]` expression.

13.5.1.3 Definition of MOL Parameters for Symbolic Preprocessing

The MOL parameters for symbolic preprocessing are defined by two MDS functions. The function `Domains[...]` defines the grid function of a spatial domain. The following extract of the MDS input file for the CLR model shows the definition of the grid for the inner tube domain of the reactor:

```
1 Domains[
2   Grid[ Tube[z],
3     Computation  -> Function[{k},(k-1)*LTube/(kmaxTube-1)],
4     Granularity  -> "kmaxTube",
5     MovingGrid   -> { SpaceSmoothing -> 4.0,
6                       TimeSmoothing  -> 0.0,
7                       MonitorStates  -> {Ti[z,t],To[z,t]} }
8   ],  ... ];
```

The spatial domain is defined by the function `Grid[...]`. The first argument of `Grid[...]` is its name, here `Tube[z]`. The grid function is specified within the `Computation` attribute in line 3. The number of grid points or CV, respectively, is defined by the attribute `Granularity` in line 4. The moving grid is activated by the `MovingGrid` attribute, which defines the moving grid parameters `SpaceSmoothing`, `TimeSmoothing`, and `MonitorStates`, see Section 13.3.4. The monitor function for this moving grid considers only the state variables `Ti[z,t]` and `To[z,t]` representing the temperatures in the inner and outer tubes.

The discretization method and its parameters are defined in MDS within the expression `Discretizations[...]`. The following extract of the CLR example shows the definition of an FD method by the MDS expression `FDMethod[...]`.

```
1 Discretizations[
2   FDMethod[ FDOrder24,
3             PolynomPoints  -> {1,3,5},        (* m *)
4             Eccentricity   -> {0,1,0},        (* q *)
5             OrderReduction -> {0,1,1}]        (* o *)
6 ];
```

In line 2, the FD method is named `FDOrder24`. This name was already used in the `SystemEquations` function presented above. The FD method is specified by the attribute values of `PolynomPoints`, `Eccentricity`, and `OrderReduction`, see Section 13.4.1. Each attribute value is a list comprising three values for the spatial derivative of 0th, 1st, and 2nd order. Here the Lagrange polynomials for approximation of the spatial derivative of 1st and 2nd order are an upwind-biased scheme of 2nd order and a central-difference scheme of 4th order, respectively.

13.5.1.4 Simulation Results

The CLR model (13.30) through (13.35) is discretized with the FD discretization method described in the previous subsection. In particular, a moving grid with 40 grid points for the tube domain and 50 grid points for the loop domain are used for spatial discretization. The PDE model comprises 9 PDE, 3 BC, and 6 inner BC. These PDE are transformed to 390 DAE by the symbolic preprocessing tool. The moving grid method adds 90 ODE or AE for $\tau \neq 0$ or $\tau = 0$, respectively. The preprocessed CLR model is transformed by the code generator to a DIVA simulation model.

The simulation results in Figure 13.12 obtained by DIVA show the temperature in the top and the mole fraction x_2 in the middle diagrams vs. the space coordinate z at different time steps. The solutions depict the autonomous periodic operation characterized by traveling waves with fronts of different steepness [23]. In the left diagrams, the second reaction front moves from the inner tube domain toward the end of the loop domain. In the right diagrams, the front moves back against the flow direction of the fluid.

The grid movement is also shown in Figure 13.12. It clearly shows the grid points following synchronously the temperature front as well as the fixed grid points of the domain transitions at $z = 0.45m$ and $z = 1.034m$. The horizontal lines mark the times of the spatial solution profiles in the temperature and mole fraction diagrams above. The simulation has been performed with moving grid parameters $\kappa = 4.0$ and $\tau = 0$.

A more detailed analysis of the nonlinear behavior of the CLR was performed using the continuation methods presented in Section 13.2.1.3 [13, 23, 24]. Furthermore, the Matlab interface was used for observer and controller design for non-autothermal operation [12].

13.5.2 Moving-Bed Chromatographic Process

The second chemical engineering example for MOL application is a moving-bed chromatographic separation process (MBC). For the continuous isolation of the pure

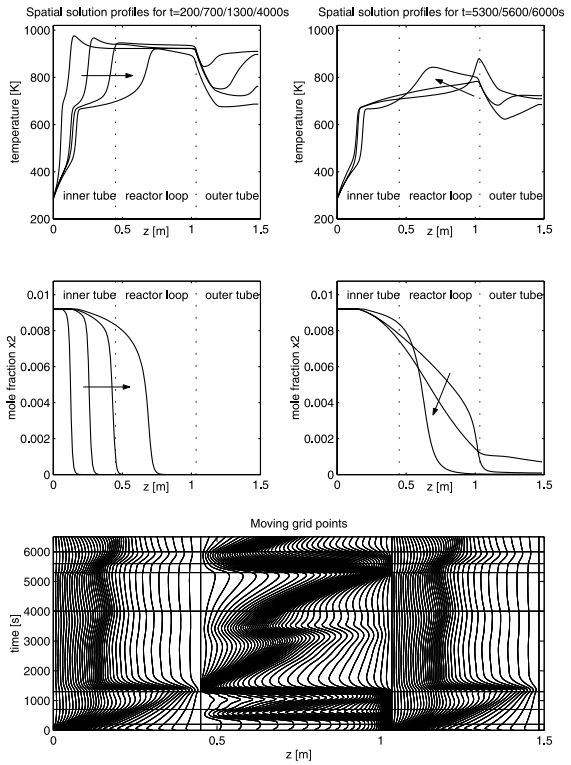


FIGURE 13.12
Simulation spatial profiles of temperature and mole fractions over the three domains of the circulation-loop reactor (CLR) as well as the grid-point movement.

components, the process operates with counter-current flow of a liquid and a solid adsorbent phase. For stable MOL discretization, the reverse direction of the convective transport in both phases has to be taken into account. This is demonstrated by the application of FV schemes with upwind and downwind profile assumptions.

With reference to Figure 13.13, the MBC unit consists of four sections bordered by two inlet nodes (feed and solvent) and two outlet nodes (raffinate and extract). For suitable operating conditions, the stronger adsorbed components can be withdrawn in the extract, the remaining weaker adsorbed components appear in the raffinate.

This MBC process is defined as a flowsheet of several process units on the plant level of the DIVA simulation kernel. The MOL discretization concerns only one generic MBC zone which is used four times within the flowsheet. In contrast to the previous CLR example, the coupling of the adjacent domains is defined by interconnection of process units (13.2) instead of inner boundary conditions.

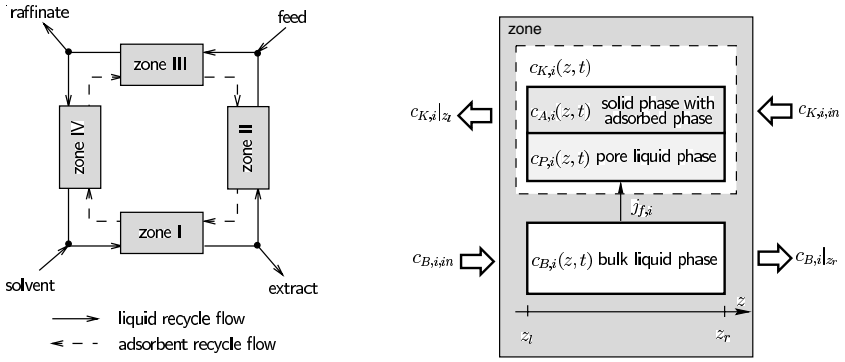


FIGURE 13.13
Scheme of a moving-bed (MBC) process (left) and of a single MBC zone (right) with the bulk (B), pore (P), and adsorbent (A) phases as well as the pseudo-homogeneous phase (K) which contains A and P. The input ($c_{K,i,in}, c_{B,i,in}$) and output ($c_{K,i}\big|_{z_l}$, $c_{B,i}\big|_{z_r}$) concentrations are also shown in the right diagram.

13.5.2.1 Model Equations

The MBC zones are modeled by a 1-dimensional spatial domain consisting of a homogenous bed of spherical porous particles (radius R_p) with a constant bed porosity ψ. The mass balances for the components $i = 1(1)n_c - 1$ lead to the following set of PDE for the concentrations $c_{j,i}(z,t)$, $j \in \{B, K\}$. The index B represents the bulk phase (Figure 13.13). The adsorbent and the pore fluid are combined to a pseudo-homogeneous phase with index K in Figure 13.13. No mass balances are required for the solvent concentrations $c_{j,n_c}(z,t)$, $j \in \{B, K\}$. A detailed description of the modeling and the model assumptions of the considered chromatographic separation can be found in [35]. The PDE and BC are for the bulk concentrations $c_{B,i}$, $i = 1(1)n_c - 1$

$$\frac{\partial c_{B,i}}{\partial t} = -v\frac{\partial c_{B,i}}{\partial z} + E\frac{\partial^2 c_{B,i}}{\partial z^2} - \frac{3(1-\psi)}{R_P\psi}\underbrace{\beta^*(c_{B,i} - c_{P,i})}_{j_{f,i}} \quad z \in (z_l, z_r)\ , \tag{13.36}$$

$$0 = E\left.\frac{\partial c_{B,i}}{\partial z}\right|_{z_l} - v(c_{B,i}\big|_{z_l} - c_{B,i,\text{in}}), \qquad 0 = \left.\frac{\partial c_{B,i}}{\partial z}\right|_{z_r} \tag{13.37}$$

and for the concentrations $c_{K,i}$, $i = 1(1)n_c - 1$

$$\frac{\partial c_{K,i}}{\partial t} = u\frac{\partial c_{K,i}}{\partial z} + \frac{3}{R_p(\epsilon_P + \epsilon_A)} \cdot \underbrace{\beta^*\left(c_{B,i} - c_{P,i}\right)}_{j_{f,i}}\ , \quad z \in [z_l, z_r) \tag{13.38}$$

$$c_{K,i}\big|_{z_r} = c_{K,i,in}\ . \tag{13.39}$$

Again, the model is completed by appropriate initial conditions. The mass transfer through the liquid phase from the bulk to the pore (Figure 13.13) is denoted by $j_{f,i}$. Parameter β^* marks the mass transfer coefficient and E is the axial dispersion coefficient. The symbol v is the interstitial velocity of bulk-fluid and u is the linear velocity of the solid phase in the counter-current system. Since adsorbent flow is assumed to be purely convective, only an inlet boundary condition (13.39) exists for each zone.

The concentrations $c_{K,i}(z,t)$ are defined as

$$c_{K,i} = \varepsilon_P c_{P,i} + (1-\varepsilon_P)\, c_{A,i}\ , \quad i = 1(1)n_c - 1\ . \tag{13.40}$$

The concentrations $c_{A,i}(z,t)$ and $c_{P,i}(z,t)$ are related by the equilibrium isotherm ($\alpha_{i,j}$ is the separation factor of two components)

$$c_{A,i} = \frac{\alpha_{i,n_c} c_{P,i}}{1 + \sum_{j=1}^{n_c-1} \left(\alpha_{j,n_c} - 1\right) c_{P,j}}\ , \quad i = 1(1)n_c - 1\ . \tag{13.41}$$

The complete model for one MBC zone with state variables $c_{j,i}(z,t)$, $j \in \{B, K, P\}$, $i = 1(1)n_c - 1$ comprises $2(n_c - 1)$ PDE, $3(n_c - 1)$ BC, and $n_c - 1$ algebraic equations.

13.5.2.2 Definition of Model Equations for Symbolic Preprocessing

The following MDS extract shows the material balances (13.36) and (13.38) within the `SystemEquations` function.

```
1 SystemEquations[
2  Distributed[
3   Domain -> "Grid1[z]",
4   Comment-> "the counter-current liquid and solid phases",
5   Scope  -> {...
6    Tensor[
7      {Scalar[ D[cB[i][z,t],t] == -v D[cB[i][z,t],z]
8          + E D[cB[i][z,t],{z,2}] +
9        3 (1-psi) beta (cB[i][z,t] - cP[i][z,t])/(Rp psi),
10        LowerBound     -> v cBin[i][t] ==
11                          v cB[i][z,t] - E D[cB[i][z,t],z],
12        UpperBound     -> 0 == D[cB[i][z,t],z],
13        Discretization -> "FVMup", ... ],
14      Scalar[ D[cK[i][z,t],t] == u D[cK[i][z,t],z] +
15         3 beta (cB[i][z,t] - cP[i][z,t])/(Rp (epsP+epsA)),
16        UpperBound     ->  cK[i][z,t] == cKin[i][t],
17        Discretization -> "FVMdown", ... ],
18      ... },
19    {{i,1,NC-1}},
20    Comment -> "loop for all components"
21   ],... }]];
```

Both equations are defined on the common spatial domain `Grid1[z]`, see line 3. The PDE definitions in lines 7-18 are part of the `Tensor[...]` object in lines 6-21. A `Tensor` object represents loops over equations. The index variable and the range for this loop over components are defined in line 19. For a stable MOL discretization, the reverse flow directions of the liquid and the solid phases have to be taken into account. Therefore, we apply two different FV schemes named `FVMup` (line 13) and `FVMdown` (line 17) to discretize the two PDE. These MOL parameter definitions are explained in the next subsection.

13.5.2.3 Definition of MOL Parameters for Symbolic Preprocessing

The MOL parameters are defined by the MDS functions `Domains[...]` and `Discretizations[...]`. In the `Domains` function of the counter-current MBC model, only one spatial grid is used for both phases. The reverse flow directions are considered by two discretization schemes which refer to the common grid function. This is shown in the `Discretizations` extract of the MDS definition.

The following MDS example for the grid function of the spatial domain used in the MBC model is very similar to the definition of the CLR example. Only the attribute `GridPractice` is introduced here in line 5. It is an optional attribute for use of FV methods to define the reference of the grid function (see Section 13.3.2).

```
1 Domains[
2  Grid[ Grid1[z],
3    Computation  -> Function[{k},zl+(k-1)*(zr-zl)/(NE-1)],
4    Granularity  -> "NE",
5    GridPractice -> "GridPoints" ]];
```

The `Discretizations` object contains two FV schemes defined by the MDS expression `FVMethod[...]`. The FV method requires only the choice of profile assumptions for the 1st and 2nd order spatial derivatives. Therefore, the MDS provides the attribute `Profile`.

```
1 Discretizations[
2   FVMethod[ FVMup,
3       Profile -> {"upwind","piecewise-linear"}],
4   FVMethod[ FVMdown,
5       Profile -> {"downwind","piecewise-linear"} ]];
```

The first `FVMethod[...]` is called `FVMup`. For the 1st order spatial derivative, the `upwind` profile assumption is defined in line 3. The second `FVMethod[...]` shows a `downwind` profile assumption due to the reverse flow direction of the solid phase.

13.5.2.4 Simulation Results

The FV schemes presented above have been applied on 150 uniform CVs for MOL discretization. Each PDE model of a MBC unit with 3 components plus solvent

consists of 6 PDE, 9 BC, and 3 algebraic equations. They are transformed into 1350 DAE by the symbolic preprocessing tool. Simulation results are shown in Figure 13.14 for the separation of a three component mixture of C_8 aromatics. Figure 13.14 shows typical stationary concentration profiles of the bulk concentrations $c_{B,i}$, $i = 1(1)3$. Due to the strong nonlinear behavior of the MBC process, steady-state simulation and optimization plays an important role for the design of moving-bed chromatographic processes. Furthermore, dynamic simulation is an important tool for developing new strategies for process operation and control. DIVA is used as an integrated tool to study these various aspects of moving-bed chromatographic processes.

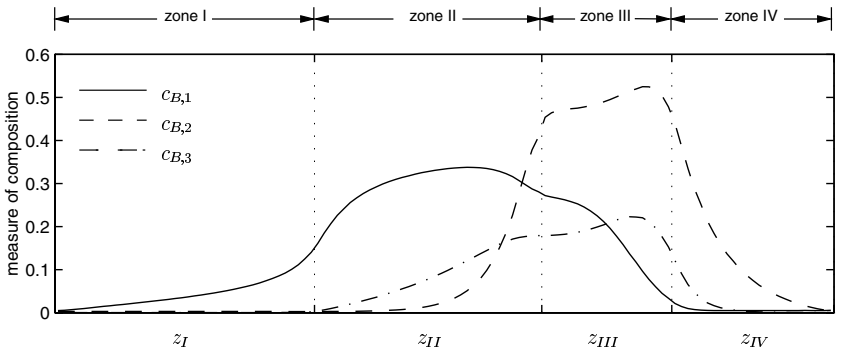


FIGURE 13.14
Simulated concentration profiles $c_{B,i}$ of a C_8 aromatics separation along the axial coordinates $z_j \in [z_{l,j}, z_{r,j}]$, $j = I, II, III, IV$.

13.6 Conclusions and Perspectives

The automatic MOL discretization of the symbolic preprocessing tool SYPPROT within the simulation environment DIVA allows to simulate, analyze, and optimize PDE and IPDE models with the powerful DAE numerical methods provided by DIVA. The symbolic preprocessing tool comprises a MATHEMATICA data structure for symbolic definition of coupled second order PDE and IPDE as well as MATHEMATICA modules for their MOL discretization. The configurable discretization methods of finite-difference and finite-volume schemes are available on uniform, non-uniform, and continuously moving 1-dimensional grids in order to apply the MOL approach to PDE and IPDE. The translation of the discretized DAE model into a code-generator input file allows us to use the simulation environment DIVA and its DAE numerical methods.

In many other numerical tools, the MOL approach is implemented without separation of model equations and discretization schemes. The model equations are part of a discretization routine for specific structured problems or they are directly written

in the discretized form. The application of different discretization schemes leads to a comparably large effort because the model equations must be reimplemented. The choice of the appropriate discretization method grows to a major decision within the complete modeling process. The uniform model representations for symbolic preprocessing as well as for the generic process unit models of the DIVA simulation kernel reduce the overall effort of modeling and simulation. Model redefinitions are not necessary either for the application of different MOL discretization schemes to PDE and IPDE by the symbolic preprocessing tool SYPPROT or by using the various DAE numerical methods of DIVA.

Future work in DIVA concerning MOL discretization is the implementation of high-resolution schemes for hyperbolic partial integro differential equations as used for modeling of population systems. A promising scheme is the robust upwind scheme [1]. Very interesting is the coupling of high-resolution schemes with the already implemented moving grid method. Encouraging results obtained by this approach are described in [20].

Acknowledgments

Main parts of the development of the simulation environment DIVA and the preprocessing tool SYPPROT have been funded by Deutsche Forschungsgemeinschaft (SFB 412). Moreover, many former colleagues and students have been engaged in this project which is also acknowledged by the authors. Especially the unpublished student thesis of J. Rieber is emphasized, in the frame of which the moving grid module has been successfully implemented in SYPPROT.

References

[1] M. Brahmadatta, R. Köhler, A. Mitrović, E.D. Gilles, and M. Zeitz, Symbolic discretization of population models for process simulation, in S. Pierucci, ed., *European Symposium on Computer Aided Process Engineering - 10,* Computer-Aided Chemical Engineering 8, pages 547–552. Elsevier, May 2000.

[2] K. E. Brenan, S.L. Campbell, and L.R. Petzold, *Numerical Solution of Initial Value Problems in Differential-Algebraic Equations,* North Holland, Elsevier Science Publishing Co. Inc., 1989.

[3] M. Caracotsios and W.E. Stewart, Sensitivity Analysis of Initial Value Problems with Mixed ODEs and Algebraic Equations, *Comp. Chem. Engng.,* **9**, (1985), 359–365.

[4] P. Deuflhard, E. Hairer, and J. Zugck, One-step and Extrapolation Methods for Differential-Algebraic Systems, *Numer. Math.,* **51**, (1987), 501–516.

[5] E.A. Dorfi and L.O'C. Drury, Simple adaptive grids for 1-D initial value problems, *J. Comp. Phys.,* **69**, (1987), 175–195.

[6] R.M. Furzeland, J.G. Verwer, and P.A. Zegeling, A numerical study of three moving grid methods for one dimensional partial differential equations which are based on the method of lines, *J. Comp. Phys.,* **89**, (1990), 349–388.

[7] E. Hairer, S.P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations; I: Non-stiff Problems, II: Stiff and Differential-Algebraic Problems,* Springer, Berlin, 1987.

[8] A. Harten, High resolution schemes for hyperbolic conservation laws, *J. Comp. Phys.,* **49**, (1983), 357–393.

[9] Harwell Laboratories, Oxford, England, *Harwell Library,* 1996.

[10] A. Helget and E.D. Gilles, Dynamische Prozeß — und Anlagensimulation, in H. Schuler, ed., *Prozeßsimulation,* pages 109–148, VCH, 1994.

[11] P. Holl, W. Marquardt, and E.D. Gilles, DIVA — A powerful tool for dynamic process simulation, *Comp. Chem. Engng.,* **12**, (1988), 421–425.

[12] X. Hua, M. Mangold, A. Kienle, and E.D. Gilles, State profile estimation of an autothermal periodic fixed-bed reactor, *Chem. Engng. Sci.,* **53**, (1998), 47–58.

[13] A. Kienle, G. Lauschke, V. Gehrke, and E.D. Gilles, On the dynamics of the circulation loop reactor — Numerical methods and analysis, *Chem. Engng. Sci.,* **50**, (1995), 2361–2375.

[14] R. Köhler, A. Gerstlauer, and M. Zeitz, Symbolic preprocessing for simulation of PDE models of chemical processes, *Journal of Math. and Comp. in Sim., special issue "Method of Lines,"* 2000 (submitted).

[15] R. Köhler, S. Räumschüssel, and M. Zeitz, Code generator for implementing differential algebraic models used in the process simulation tool DIVA, in A. Sydow, ed., *15th IMACS World Congress,* volume 3, pages 621–626, Berlin, Germany, 1997. Wissenschaft und Technik Verlag.

[16] B. Koren, A robust upwind discretization method for advection, diffusion and source terms, in C.B. Vreugdenhiland B. Koren, ed., *Numerical Methods for Advection-Diffusion Problems,* pages 117–138. Vieweg, 1993.

[17] A. Kröner, P. Holl, W. Marquardt, and E.D. Gilles, DIVA — An open architecture for dynamic simulation, *Comp. Chem. Engng.,* **14**, (1990), 1289–1295.

[18] S. Kumar and D. Ramkrishna, On the solution of population balance equations by discretization-i. a fixed pivot technique, *Chem. Engng. Sci.,* **51**, (1996), 1311–1332.

[19] B. Van Leer, Upwind-difference methods for aerodynamic problems governed by the euler equations, *Lectures in Applied Mathematics,* **22**, (1985), 327–336.

[20] S. Li and L. Petzold, Moving mesh method with upwinding schemes for time-dependent PDEs, *J. Comp. Phys.,* **131**, (1997), 368–377.

[21] S. Li, L. Petzold, and R. Yuhe, Stability of moving mesh systems of partial differential equations, *SIAM J. Sci. Comput.*, **20**, (1998), 719–738.

[22] C. Majer, *Parameterschätzung, Versuchsplanung und Trajektorenoptimierung für verfahrenstechnische Prozesse,* Dissertation, Universität Stuttgart, Düsseldorf, 1998.

[23] M. Mangold, A. Kienle, E.D. Gilles, M. Richter, and E. Roschka, Coupled reaction zones in a circulation loop reactor, *Chem. Engng. Sci.,* **54**, (1999), 2597–2607.

[24] M. Mangold, A. Kienle, K.D. Mohl, and E.D. Gilles, Nonlinear computation in DIVA — methods and applications, *Chem. Engng. Sci.,* **55**, (2000), 441–454.

[25] M. Mangold, K.D. Mohl, A. Kienle, and E.D. Gilles, Analyse nichtlinearer Phänomene bei verfahrenstechnischen Prozessen, *Chemie Ingenieur Technik,* **70**, (1998), 371–381.

[26] K.D. Mohl, A. Spieker, R. Köhler, E.D. Gilles, and M. Zeitz, DIVA — A simulation environment for chemical engineering applications, in *Informatics, Cybernetics and Computer Science (ICCS-97), Collected Volume of Scientific Papers,* pages 8–15. Donetsk State Technical University, Donetsk, Ukraine, 1997.

[27] NAG Ltd., Oxford, England, UK, *NAG Fortran Library Manual,* 1993.

[28] U. Nowak and L. Weimann, A family of newton codes for systems of highly nonlinear equations, Technical Report TR 91-10, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 1991.

[29] S.V. Patankar, *Numerical Heat Transfer and Fluid Flow,* McGraw-Hill, New York, 1980.

[30] D. Ramkrishna, The status of population balances, *Chem. Eng. Communic.,* **3**, (1985), 49–95.

[31] S. Räumschüssel, *Rechnerunterstützte Vorverarbeitung und Codierung verfahrenstechnischer Modelle für die Simulationsumgebung DIVA,* VDI Fortschritt–Berichte Nr. 20/270, VDI-Verlag, Düsseldorf, 1998.

[32] W.E. Schiesser, The numerical method of lines: Integration of partial differential equations, San Diego, CA, 1991.

[33] C.W. Shu and S. Osher, Efficient implementation of essentially non-oscillatory shock-capturing schemes, *J. Comp. Phys.,* **77**, (1988), 439–471.

[34] C.W. Shu and S. Osher, Efficient implementation of essentially non-oscillatory shock-capturing schemes ii, *J. Comp. Phys.,* **83**, (1989), 32–78.

[35] A. Spieker, E. Kloppenburg, and E.D. Gilles, Computer modeling of chromatographic bioseparation, in G. Subramanian, ed., *Bioseparation & Bioprocessing,* chapter 13, pages 329–362. Wiley-VCH, Weinheim, 1998.

[36] L. Guy, Steele Jr., *Common Lisp: The Language,* Digital Press, Bedford, MA, 2nd ed., 1990.

[37] E. Stein, A. Kienle, and E.D. Gilles, Dynamic Optimization of Multicomponent Distillation Processes, in *Scientific Computing in Chemical Engineering II, Part II,* Springer-Verlag, Berlin, 1999.

[38] P.K. Sweby, High resolution schemes using flux limiters for hyperbolic conservation laws, *SIAM J. Numer. Anal.,* **21**, (1984), 995–1011.

[39] J. Thompson and H.B. Stewart, *Nonlinear Dynamics and Chaos,* John Wiley & Sons, Chichester, 1986.

[40] F. Tränkle, A. Gerstlauer, M. Zeitz, and E.D. Gilles, Application of the Modeling and Simulation Environment PROMOT/DIVA to the Modeling of Distillation Processes, PSE'97, ESCAPE–7, Trondheim, Norway, May 26–29, 1997.

[41] F. Tränkle, *Rechnerunterstützte Modellierung verfahrenstechnischer Prozesse für die Simulationsumgebung DIVA,* VDI Fortschritt–Berichte Nr. 20/309, VDI-Verlag, Düsseldorf, 2000.

[42] J. Unger, A. Kröner, and W. Marquardt, Structural analysis of differential-algebraic equation systems — Theory and applications, *Computers Chem. Engng.,* **19**, (1995), 867–882.

[43] L.G. Verwer, J.G. Blom, R.M. Furzeland, and P.A. Zegeling, A moving grid method for one-dimensional pdes based on the method of lines, in J.E. Flaherty, P.J. Paslow, M.S. Shepard, and J.D. Vasilakis, eds., *Adaptive Methods for Partial Differential Equations,* pages 160–175. SIAM, Philadelphia, 1989.

[44] F.H. Walsteijn, Essentially non-oscillatory schemes, in C.B. Vreugdenhiland B. Koren, ed., *Numerical Methods for Advection-Diffusion Problems,* pages 139–170. Vieweg, 1993.