

An ODE/MOL PDE Template For Soil Physics

Author

Lee, Hock Seng

Published

2003

Thesis Type

Thesis (PhD Doctorate)

School

Australian School of Environmental Studies

DOI

[10.25904/1912/740](https://doi.org/10.25904/1912/740)

Rights statement

The author owns the copyright in this thesis, unless stated otherwise.

Downloaded from

<http://hdl.handle.net/10072/365588>

Griffith Research Online

<https://research-repository.griffith.edu.au>

**SCHOOL OF ENVIRONMENTAL ENGINEERING
GRIFFITH UNIVERSITY, BRISBANE**

**AN
ODE/MOL
PDE TEMPLATE
FOR
SOIL PHYSICS
A NUMERICAL STUDY**

by

**Hock S. Lee, B.Sc. (Honours)
(Griffith University)**

30th November 2001

**A dissertation submitted in fulfillment of the requirements for the degree of
Doctor of Philosophy in the Faculty of Australian Environmental Sciences.**

STATEMENT OF ORIGINALITY

To the best of my knowledge and belief, this work has never previously been submitted for a degree or diploma in any University and contains no material previously published or written by another person except where due reference is made in the thesis itself.

Hock Seng Lee

ACKNOWLEDGEMENTS

I would like to thank the following gentlemen whose advice and guidance resulted in the completion of this dissertation:

Dr. Graham Sander, an expert on water flow phenomena in soil and my minor supervisor, who raised many question critical to the progress of my thesis.

And, **Dr. Roger Braddock**, my main supervisor, an expert in numerical modelling who had raised many objective criticisms that contributed greatly to the improved quality of the thesis. Also not forgetting, his support, encouragement, patience, time and guidance, without which this thesis would not materialize.

DEDICATION

*Dedicated to my other half, my beloved partner, **Siau Lin Lih**. Without her support and love, this thesis would not materialize. Most of all, I would like to thank her for her patience and understanding toward my quest for the honorific 'Dr'.*

ABSTRACT

The aim of the thesis is to find a method, in conjunction with the ordinary differential equation (ODE) based method of lines (MOL) solution of Richards' equation, to model the steep wetting front infiltration in very dry soils, accurately and efficiently. Due to the steep pressure head or steep water volumetric content gradients, highly nonlinear soil hydraulic properties and the rapid movement of the wetting front, accurate solutions for infiltration into a dry soil are usually difficult to obtain. Additionally, such problems often require very small time steps and large computation times.

As an enhancement to the used ODE/MOL approach, Higher Order Finite Differencing, Varying Order Finite Differencing, Vertical Scaling, Adaptive Schemes and Non-uniform Stretching Techniques have been implemented and tested in this thesis. Success has been found in the ability of Vertical Scaling to simulate very steep moving front solution for the Burgers' equation. Unfortunately, the results also show that Vertical Scaling needs significant research and improvement before their full potential in routine applications for difficult nonlinear problems, such as Richard's equation with very steep moving front solution, can be realized. However, we have also shown that the use of the composed form of RE and a 2nd order finite differencing for the first order derivative approximation is conducive for modelling steep moving front problem in a very dry soil. Additionally, with the combination of an optimal influx value at the edges of the inlet, the ODE/MOL approach is able to model a 2-D infiltration in very dry soils, effectively and accurately.

Furthermore, one of the strengths of this thesis is the use of a MATLAB PDE template. Implementing the ODE/MOL approach via a MATLAB PDE template has shown to be most suitable for modelling of partial differential equations. The plug and play mode of modifying the PDE template for solving time-dependent partial

differential equations is user-friendly and easy, as compared to more conventional approaches using Pascal, Fortran, C or C++. The template offers greater modularity, flexibility, versatility, and efficiency for solving PDE problems in both 1-D and 2-D spatial dimensions. Moreover, the 2-D PDE template has been extended for irregular shaped domains.

Table of Contents

STATEMENT OF ORIGINALITY	2
ACKNOWLEDGEMENTS	3
DEDICATION	4
ABSTRACT	5
TABLE OF CONTENTS	7
LIST OF FIGURES	10
LIST OF TABLES	12

CHAPTER 1. INTRODUCTION

1.1	Origins of the Present Work.	16
1.2	Mathematical Modelling of Fingering	17
1.3	Brief Summary of Each Chapter	21

CHAPTER 2. LITERATURE REVIEW

2.1	Richards' Equation	23
2.2	Hydraulic Conductivity Model	25
2.3	Hysteresis Model	26
2.4	Current Numerical Methods in Soil Physics	29
2.5	Software libraries, MATLAB and Template in Numerical Computation	42
2.6	Conclusions	45

CHAPTER 3. MODELS USED

3.1	Introduction	47
3.2	Burgers' Model of Advective-Convective Equation	47
3.3	Analytical Solution of 1-D Constant Flux Infiltration using Richards' equation.	49
3.4	Model of 2-D Infiltration in Very Dry Soil	52
3.5	Performance Indicators	56

CHAPTER 4. PDE TEMPLATE

4.1	Introduction	59
4.2	One Dimensional Heat Equation	60
4.3	Template for Solving Various 1-D PDE	69
4.4	Template for Solving 2-D Problem on a Rectangular Domain	75
4.5	Modification of Template for 2-D Problem with Irregular Shaped Domain	78
4.6	Extensions	84

CHAPTER 5. FORMS OF RICHARDS EQUATION

5.1	Introduction	85
5.2	The ODE/MOL Solution of Richards Equation	87
5.3	Numerical Experiments	89
5.4	Results and Discussion	89
5.5	Conclusions	93

CHAPTER 6. HIGHER ORDER SCHEMES

6.1	Introduction	95
6.2	Numerical Differentiation	96
6.3	First Order Differentiation Matrix (FODM)	98
6.4	Numerical Experiments	101
6.5	Results and Discussion	101
6.6	Conclusions	108

CHAPTER 7. VARYING ORDER IN FODM OVER 1-D DOMAIN

7.1	Introduction	109
7.2	Stage-wise Differentiation Verses Direct Differentiation	110
7.3	Method of Varying the Order of the FODM	111
7.4	Manual Varying Order FODM	112
7.5	Numerical Experiments	113
7.6	Results and Discussion	115
7.7	Conclusions	124

CHAPTER 8. ADAPTIVE SCHEME 1-D

8.1	Introduction	126
8.2	Adaptive Scheme 1-D (CDGA)	127
8.3	Numerical Experiments	134
8.4	Results and Discussion	135
8.5	Conclusions	141

CHAPTER 9. NON-UNIFORM GRID STRETCHING

9.1	Introduction	143
9.2	Non-uniform Grid Stretching in ODE/MOL	144
9.3	Numerical Experiments	146
9.4	Results and Discussion	147
9.5	Conclusions	149

CHAPTER 10. VERTICAL SCALING 1-D

10.1	Introduction	151
10.2	Maximized Time Stepping for Highly Viscous BE	152
10.3	Numerical Experiments	155
10.4	Results and Discussion	155
10.5	Conclusions	159

CHAPTER 11. 2-D INFILTRATION MODELS

11.1	Introduction	161
11.2	Numerical Experiments	162
11.3	Results and Discussion	164
11.4	Conclusions	170

CHAPTER 12. CONCLUSIONS AND RECOMMENDATIONS

12.1	Conclusions	172
12.2	Future Research	175

APPENDICE A. SUBROUTINES FOR 1-D PDE'S TEMPLATE	179
--	-----

APPENDICE B. SUBROUTINES FOR 2-D PDE'S TEMPLATE	181
--	-----

APPENDICE C. CONTENT IN CD-ROM	183
---------------------------------------	-----

Noted that the programs in the attached CD-ROM are for your perusal and improvement; with the condition that it is not used for commercial purposes and due reference is made.

REFERENCES	185
-------------------	-----

LIST OF FIGURES

- FIGURE 4.1 Discretization and Numbering
- FIGURE 4.2 The discretized domain, G matrix, k vector and the boundary points for a rectangle domain.
- FIGURE 4.3 The discretized domain of an irregularly shaped domain.
- FIGURE 5.1 \mathbf{q} and Ψ profile with respect to z from 0 to 0.3625 min ($\mathbf{u}=0.85$).
- FIGURE 5.2 \mathbf{q} and Ψ profile with respect to z from 0 to 36.25 ($\mathbf{u}=0.85$).
- FIGURE 5.3 \mathbf{q} and Ψ profile with respect to z from 0 to 0.3625 ($\mathbf{u}=0.99995$).
- FIGURE 5.4 \mathbf{q} and Ψ profile with respect to z from 0 to 36.25 ($\mathbf{u}=0.99995$).
- FIGURE 6.1 Even Order FODM Template
- FIGURE 6.2 Odd Order FODM Template
- FIGURE 6.3 Solution Profile in Ψ for BE, $\mathbf{n} = 0.01$.
- FIGURE 6.4 Solution Profile in Ψ for BE, $\mathbf{n} = 0.001$.
- FIGURE 7.1 The Graphical Interface (GUI).
- FIGURE 7.2 The Template.
- FIGURE 7.3 Solution at $t = 0.1$ for 2nd order FD.
- FIGURE 7.4 Solution at $t = 0.1$ for 6th order FD.
- FIGURE 7.5 Solution at $t = 0.1$ for 8th order FD.
- FIGURE 7.6 Solution at $t = 0.1$ for 16th order FD.
- FIGURE 7.7 GUI at the first successful step.
- FIGURE 7.8 Clicks of the ‘Soln’ and ‘hz’ button at the first successful step.
- FIGURE 7.9 Clicks of the ‘hzz’ and ‘Curve’ button at the first successful step.
- FIGURE 7.10 Click of the ‘vFODM’ button and the assigned stencil for each node at the first successful step.
- FIGURE 7.11 Clicks of the ‘vFODM’ button.
- FIGURE 7.12 Solution at the final successful step.

FIGURE 7.13 Plot of MRE at all final successful step against time for the heat Equation using varying FODM.

FIGURE 7.14 Solution at the final successful step, $2 \rightarrow 6$ FODM.

FIGURE 7.15 Plot of MRE at all final successful step against time for the heat equation using $2 \rightarrow 6$ FODM.

FIGURE 7.16 Plot of MRE at all final successful step against time for the heat equation using $12 \rightarrow 14$ FODM.

FIGURE 8.1 Grid clustering in z with 21 points, for various h values.

FIGURE 8.2 Oscillations developed in system with $h = 21$ and 21 points.

FIGURE 8.3 Solution Profile for $h = 0$.

FIGURE 11.1 Showing node numbering and flux approximation near discontinuity.

FIGURE 11.2 Solution oscillations for $s = 1$ (before modification) and $s = 0.5$ (after modification) of the boundary condition, at 1 minute in time, using a spatial step of 1 cm.

FIGURE 11.3 Oscillation at the influx before and after modification of influx conditions reflected via the pressure contour of the domain at 1 minute in time, using a spatial step of 1cm.

LIST OF TABLES

- TABLE 4.1 The PDE Template for solving the heat equation.
- TABLE 4.2 The right-hand side function for the heat equation.
- TABLE 4.3 Results for the heat equation at $t = 1$.
- TABLE 4.4 The PDE Template for solving the hyperbolic equation.
- TABLE 4.5 The right-hand side function for the hyperbolic equation.
- TABLE 4.6 Results for the hyperbolic equation at $t = 1$.
- TABLE 4.7 The PDE Template solving the 1-D Burgers' equation.
- TABLE 4.8 The right-hand side function for the 1-D Burgers' equation.
- TABLE 4.9 Results for the 1-D Burgers' equation.
- TABLE 4.10 The PDE Template solving the 2 Coupled nonlinear PDE's.
- TABLE 4.11 The right-hand side function for the 2 Coupled nonlinear PDE's.
- TABLE 4.12 Results for the 2 Coupled nonlinear PDE.
- TABLE 4.13 The PDE Template solving the 2-D Burgers' equation.
- TABLE 4.14 The right-hand side function for the 2-D Burgers' equation.
- TABLE 4.15 Results for the 2-D Burgers' equation at $t=1$.
- TABLE 4.16 PDE Template solving the 2-D Laplacian on a irregular shaped domain with spatial step of $dx=0.1143$ and $dz=0.0571$, and $21*21$ points.
- TABLE 4.17 The right-hand side function for the Laplacian equation with irregular shaped domain.
- TABLE 4.18 Relative error in the domain at $t=500$ for an irregular shaped domain.
- TABLE 5.1 Errors and CPU-Time at model time 0.3625 min ($\mathbf{u}=0.85$).
- TABLE 5.2 Errors and CPU-Time at 36.25 min ($\mathbf{u}=0.85$).
- TABLE 5.3 Errors and CPU-Time at 0.3625 min ($\mathbf{u}=0.99995$).
- TABLE 5.4 Errors and CPU-Time at 36.25 min ($\mathbf{u}=0.99995$).
@ implies integration proceeds very slowly at 0.5607 min.
- TABLE 6.1 Example 1, MRE for $\mathbf{n}=1$ where integration is from 0 to 0.5 second for a spatial step of $1/20$ and $tol = 1e-16$.

- TABLE 6.2 Example1, MRE for $n=2$ where integration is from 0 to 0.5 second for a spatial step of 1/20 and $tol=1e-16$.
- TABLE 6.3 Example 1, MRE for $n=0.1$ where integration is from 0 to 0.5 second for a spatial step of 1/20 and $tol=1e-6$.
- TABLE 6.4 Example 1, MAE for $n=0.01$ where integration is from 0 to 0.5 second for a spatial step of 1/200 and $tol=1e-6$.
- TABLE 6.5 Example 1, MAE for $n=0.001$ where integration is from 0 to 0.5 second for a spatial step of 1/200 and $tol=1e-6$.
- TABLE 6.6 Example 2, Model A's results for different orders where integration is from 0 to 36.25 min for a spatial step of 0.75 and $tol=1e-6$ ($u=0.85$).
- TABLE 6.7 Example2, Model A's results for different orders where integration is from 0 to 36.25 min for a spatial step of 0.75 and $tol=1e-16$. ($u=0.85$)
- TABLE 6.8 Example2, Mode B's results for different orders where integration is from 0 to 36.25 min for a spatial step of 0.75 and $tol=1e-6$. ($u=0.99995$)
- TABLE 6.9 Example2, Model B's results for different orders where integration is from 0 to 36.25 min for a spatial step of 0.75 and $tol=1e-10$. ($u=0.99995$)
- TABLE 7.1 Results of comparison between stage-wise differentiation and direct derivatives approximation. The results in the column are MRE over the 13 nodes.
- TABLE 7.2 Results for the heat equation at $t=0.1$.
- TABLE 7.3 Results for the heat equation using varying FODM at $t=0.1$.
- TABLE 7.4 Results for the heat equation using varying FODM at $t=0.1$.
- TABLE 7.5 Results for the heat equation using varying FODM at $t=0.1$.
- TABLE 7.6 Results for the heat equation using varying FODM at $t=0.1$.
- TABLE 7.7 Results for the heat equation using varying FODM at $t=0.1$.
- TABLE 7.8 Results for the heat equation using varying FODM at $t=0.1$.
- TABLE 8.1 $\partial z / \partial t$ function.
- TABLE 8.2 Effect of W_1 and W_2 for $h=1$, $n=1$ and $Tol=10^{-6}$.
- TABLE 8.3 Effect of W_1 and W_2 for $h=1$, $n=10^{-3}$ and $Tol=10^{-6}$.

- TABLE 8.4 Effect of W_1 and W_2 for $h = 30$, $n = 1$ and Tol= 10^{-6} .
- TABLE 8.5 Effect of W_1 and W_2 for $h = 30$, $n = 10^{-3}$ and Tol= 10^{-6} .
- TABLE 8.6 Effect of h and Tol for $n = 10^{-3}$.
- TABLE 8.7 Effect of h and Tol for $n = 10^{-3}$.
- TABLE 8.8 $n = 1$, where integration is from 0 to 0.5 second for a spatial step of 1/20.
- TABLE 8.9 $n = 2$, where integration is from 0 to 0.5 second for a spatial step of 1/20.
- TABLE 8.10 $n = 0.1$, where integration is from 0 to 0.5 second for a spatial step of 1/20.
- TABLE 8.11 $n = 0.01$, where integration is from 0 to 0.5 second for a spatial step of 1/200.
- TABLE 8.12 $n = 0.001$ where integration is from 0 to 0.5 second for a spatial step of 1/200.
- TABLE 8.13 $n = 0.001$ where integration is from 0 to 0.5 second for a spatial step of 1/200.
- TABLE 9.1 Sensitivity Analysis of the Clustering Parameters, Pt and b at time = 0.3625 min.
- TABLE 9.2 Sensitivity Analysis of the Clustering Parameter, Pt at time = 36.25 min.
- TABLE 10.1 The effect of Vertical Scaling on the time step-size for $n = 0.01$.
- TABLE 10.2 The effect of the Vertical Scaling on the time step-size for $n = 1$.
- TABLE 10.3 Effect of the step-size on parabolic dominated Burgers' equation.
- TABLE 10.4 Effect of the step-size on hyperbolic dominated Burgers' equation.
- TABLE 10.5 Effect of Vertical Scaling on the parabolic dominated Burgers' equation.
- TABLE 10.6 Effect of Vertical Scaling on the hyperbolic dominated Burgers' equation.
- TABLE 10.7 Effect of the tolerance, using the optimal Mf , applied on hyperbolic dominated Burgers' equation.
- TABLE 10.8 Effect of the order of the spatial finite differencing using the optimal Mf , applied on the hyperbolic dominated Burgers' equation.

TABLE 11.1 Efficiencies of the numerical algorithm at spatial step of 4 cm (21*21 nodes).

TABLE 11.2 Efficiencies of the numerical algorithm at spatial step of 2 cm (41*41 nodes).

TABLE 11.3 Efficiencies of the numerical algorithm at spatial step of 1 cm (81*81 nodes).

TABLE 11.4 Efficiencies of the numerical algorithm at spatial step of 4 cm (21*21 nodes), with an optimal $\mathbf{s} = 0.5$ value.

TABLE 11.5 Efficiencies of the numerical algorithm at spatial step of 1.5 cm (1600nodes), with an optimal $\mathbf{s} = 0.7$ value.

TABLE 11.6 Efficiencies of the numerical algorithm at spatial step of 1 cm (81*81 nodes), with an optimal $\mathbf{s} = 0.7$ value.

Introduction

Simplicity is the most difficult thing to secure in this world; it is the last limit of experience, and the last effort of genius.

George Sand

1.1 ORIGINS OF THE PRESENT WORK

In 1980, approximately a thousand wells in eastern Long Island were contaminated with Aldicarb, a pesticide used to fight the Colorado potato beetle. The local community was stunned because they did not expect the pesticide to get into the groundwater so rapidly, nor in such high concentration. Even the local specialists were equally shocked, obviously their conception of contaminated flow in soil is inadequate. (Steenhuis and Parlange, 1990).

Modern agriculture depends on a broad range of fertilisers and pesticides to assure reliable crop yields. Although integrated pest management may reduce the amount of chemical needed, a total ban is not currently feasible. Thus the contamination of groundwater with agricultural chemicals seems to be a long-term hazard: once the chemicals get into groundwaters, they may remain there for hundreds of years. (Steenhuis and Parlange, 1990).

In the Netherlands, as well as in many other parts of the world, water repellent soils are widespread and they often show irregular moisture patterns that lead to accelerated transport of water and solutes to the groundwater and surface water. Water repellent soils may lead to the formation of preferential flow paths, or fingered flows,

which provide avenues for the relatively rapid movement of water down through the soil (Ritsema et al., 1998). These preferential flow paths are induced by an unstable wetting front formation. Roberts and Carbon (1972) demonstrated that films of organic material on the sand grains caused water repellency. In water repellent sand, loam, clay, and peat soils with grasscover, water moves downward through narrow channels, tongues or fingers. Thus preferential flow or fingering may be more common than is presently thought.

Surprisingly, only during the last two decades has preferential flow attracted the attention of soil physicists (Hillel and Baker, 1988). Obtaining a quantitative description of this flow through a soil profile is an integral step in the formulation of policies pertinent to the management of the environment. The development of mathematical and numerical models usually serve this purpose, where in most cases analytical solutions are unavailable.

1.2 MATHEMATICAL MODELLING OF FINGERING

Mathematical and numerical models using averaged transport parameters are widely used for predicting water and solute movement through the unsaturated soil (Gee et al., 1991; Van Genuchten, 1991). Disagreement between model results and actual field measurements often occur (Jury and Fluhler, 1992). In particular, many studies have found, contrary to model prediction, high concentrations of pesticides in shallow groundwater or agricultural tile lines shortly after the application of the pesticide (Kladvko et al., 1991; Smith, 1990) – these are probably evidence of preferential flow. These averaged models have been developed at the macroscale.

At present, theoretical and experimental models of gravitational fingering abound in the literature, developed primarily through mathematical analysis and theory based on laboratory experiments. Generally linear stability theory, sometimes with

dimensional analysis, was used to predict the growth of a dominant wavelength that may serve as an estimator of finger width and velocity (e.g. Saffman and Taylor, 1958; Glass et al., 1989a; Glass et al., 1991). Recent work on the prediction of finger width using a hysteresis theory was also carried out by Ritsema et al. (1998). Other numerical methods used in research of fingering include that of Mandelbrot (1982), who put forward the concept of a fractal to describe a complex geometry relating self-similarity and scale of observation. A more recent application of fractal theory to describe fingering structure and to estimate the effective surface tension at the wetting front during infiltration was carried out by Chang et al. (1994). These theories have been developed at the micro scale, but have yet to provide any useful links to the medium to macroscale phenomena.

The finger flow phenomena occupies a middle scale between the pore, or micro scale, and the large scale where averaged models are available. However, conventional numerical simulations of gravity-driven fingering are few in the literature. Nieber (1996) has used a finite element approach to build models of fingered flow. Eliassi and Glass (2001) have built finite difference models of fingered flow and have been led to question the results by Nieber (1996) and the use of upwinding to control the numerical errors. Eliassi and Glass (2001) also considered the propagation of numerical oscillations. Their work may be flawed by the neglect of the temporal truncation errors. Eliassi and Glass (2001) also concluded that Richards' equation may not contain all the essential physics required to adequately explain fingering, but they were unable to suggest how the model should be improved. As yet, no numerical studies using conventional numerical techniques and incorporating hysteresis, have been conducted for two dimensional gravitational fingering that we could declare good enough to model the physics of gravity-driven fingering realistically and efficiently. Results obtained via these numerical methods are not as accurate as it should be as compared to solving a

non-convective dominated flow (Nieber, 1996). The main reasons for these difficulties are that the very steep wetting fronts in the dry soil, and the highly non-linear transport equation, cause convergence difficulties for the numerical method unless very small time-steps are used (Pan and Wierenga, 1997). Often such problems require very small time steps and large computation times.

In recent years, many developments in numerical analysis have successfully conquered various aspects of solving a convective dominated flow. Methods include the multigrid techniques, the finite element or volume methods, variable step/order ordinary differential equation solvers, adaptive mesh-refinements, spline approximations, spectral methods, iterative algorithms, and domain decomposition methods. Unfortunately the degree of experience and expertise needed to successfully implement these numerical methods are formidable.

The Method of Lines (MOL) has been used to handle both diffusion dominated and convection dominated flows (Schiesser, 1991). The Method of Lines approach to tackling a Partial Differential Equation, is to discretise the spatial coordinates to generate a set of ordinary differential equations (ODEs). This resulting set of ODEs may then be solved using the suites of ODE solvers, which are located in mathematical software such as in MATLAB. This provides another means of investigating the problems of fingering in soil physics, and will contribute to the current lively debate on fingering.

The use of MATLAB as a programming tool also provides vectorisation properties which are extremely useful in implementing the Method of Lines (Lee 1996, 1998). This leads to the development of templates, which can be used to automate the problem description for the application of Richards' Equation to various geometries, boundary conditions and initial conditions. The resulting template will also find application to other diffusive and convective systems.

The Method of Lines has not yet been applied to the study of water flow in soils. Other techniques such as Finite Elements (Nieber, 1996) and Finite Differences (Eliassi and Glass, 2001) have been used and none is yet proving to be totally satisfactory in soil physics. This thesis will investigate the MOL in application to soil water problems, and add a new tool to this area of science. This new tool will be tested and verified on problems from the literature.

The aims and objectives of this thesis are to:

- Develop a Method of Lines solution to Richards' Equation, on a template approach using vectorised properties of MATLAB.
- Test and verify the method using examples selected from the literature.
- Apply the solution process to flow of water through soils, and preferential flows.

The thesis will proceed through the development of a modelling template based on the vectorisation properties of MATLAB. Modelling is an essential part of science and engineering and the template will facilitate the application of the MOL to environmental problems. This will contribute to the modelling field by allowing the modeller to concentrate on essential physics as input, and the direct causes in the output. Other packages involve the modeller directly in the implementation difficulties of sophisticated numerical algorithms, rather than considering the modelling per se.

This thesis is significant in seeking to develop a simple numerical method that is capable of modelling a fingered flow incorporating hysteresis. Only with a full numerical model, is it possible for soil scientists to study in a detailed and systematic manner the important effects of initial moisture distribution, infiltration and drainage cycles, soil properties on finger development, travel times and finger fluxes over a range of practical applications in both the agricultural and mining industries. A better

understanding of preferential flow will let us predict more accurately the likelihood that groundwater will be affected by agricultural chemical, landfills, and industrial wastes.

1.3 BRIEF SUMMARY OF EACH CHAPTER

The thesis will cover the following material:

- Chapter 2: A review of methods that researchers had offered for the solution to the above challenges. The review concentrates on methods that are simple and have the ability to combat very steep gradient problems. In particular, we use the Richard's Equation to model water infiltration into very dry soil.
- Chapter 3: The numerical models and all measures of performance used in the thesis are discussed in this chapter. The problems are chosen for their characteristic steep moving fronts and in most cases have an analytical solution. In this way, the numerical algorithms can be verified and demonstrate their capability to handle very highly nonlinear problems with steep moving fronts, especially in a very dry soil.
- Chapter 4: A PDE template for solving Partial Differential Equations (PDE) is created using the concept of a differentiation matrix and the matrix based approach to scientific computing that was introduced in the MATLAB (Matrix Laboratory) software package. Finite differences are used to create the differentiation matrix. The method of lines (MOL) is used to convert the PDE system to a system of ordinary differential equations (ODE) or differential algebraic equations (DAE) that can be solved by an ODE/DAE integrator.
- Chapter 5: Numerical solutions based on different forms of the Richard equation can lead to significantly different results. Here, the pressure and mixed form of Richard's equation (RE) and its decomposed form are investigated for modelling variably saturated flow through soil.

- Chapter 6: Higher-order finite difference schemes (2-16 orders) are used in the method of lines (MOL) to solve 1-dimensional hyperbolic and parabolic partial differential equations.
- Chapter 7: A ‘p-refinement’ adaptive scheme is implemented where we adapt the different orders of the finite difference stencil of the first derivative approximation at these areas of large variation (the mesh remains fixed), instead of the conventional way of adapting the nodes. The variation of the order of finite difference approximation in the FODM is correlated with the profile of the error indicators.
- Chapter 8: The Continuous Dynamic Grid Adaptation (CDGA) of Dietachmayer and Droegemeier [1992], is investigated and implemented using MATLAB and its built-in libraries. The CDGA is basically comprised of a moving grid and grid stretching methods.
- Chapter 9: Non-uniform grid stretching in conjunction with the Method of Lines (MOL) is applied to a difficult problem in soil physics, e.g., the numerical tracking of a steep wetting front infiltrating through an initially very dry soil.
- Chapter 10: Vertical Scaling and an explicit integrator is used in the MOL to overcome the time stepping constraint for the integration of the highly viscous Burgers’ equation.
- Chapter 11: Numerical testing of 2-dimensional models of water infiltration into a very dry soil is carried out.
- Chapter 12: In the final chapter, conclusions are made about the template and the numerical methods used in this thesis for modelling water infiltration into very dry soils. Suggestions for future research are also given.

Literature Review

Only those who attempt the absurd will achieve the impossible. I think. I think it's in my basement. Let me go upstairs and check.

Escher

2.1 RICHARDS' EQUATION

The movement of water at the macroscopic scale, through a porous medium is governed by Richards' Equation (RE) [Richards, 1931]. This equation may be derived from two classical laws of soil physics – Darcy's Law and Conservation of Mass or the Continuity Equation.

In 1856, Darcy gave the earliest quantitative description of the processes governing soil water movement which can be represented mathematically by

$$q = -K\nabla\Psi, \quad (2.1)$$

where q is the flux of water = Q/A (units: length/time), Ψ is the soil potential (units: length), K is the soil water conductivity (units: length/time) and $\nabla\Psi$ is the potential gradient (units : dimensionless). K is generally considered a constant for a saturated soil. In unsaturated soils, K is strongly dependent on the water content of the soil. [Hillel, 1971].

The Continuity equation is given by

$$\frac{\partial \mathbf{q}}{\partial t} = -\nabla q, \quad (2.2)$$

where \mathbf{q} is the soil water content [Hillel, 1980].

Richard's equation can be derived by substituting from Darcy's law, equation (2.1), for the water flux model, into equation (2.2) which then becomes

$$\frac{\partial \mathbf{q}}{\partial t} = -\nabla \cdot (-K(\mathbf{q}) \nabla \Psi). \quad (2.3)$$

On using Cartesian coordinates with x, y horizontal and z vertically down, then

$$\begin{aligned} \frac{\partial \mathbf{q}}{\partial t} &= -\nabla \cdot (K(\mathbf{q}) \nabla \Psi) \\ &= \frac{\partial}{\partial x} \left(K \frac{\partial \Psi}{\partial x} \right) + \frac{\partial}{\partial y} \left(K \frac{\partial \Psi}{\partial y} \right) + \frac{\partial}{\partial z} \left(K \frac{\partial \Psi}{\partial z} \right) - \frac{\partial K}{\partial z}, \end{aligned} \quad (2.4)$$

which is the mixed form of Richard's Equation: mixed in the sense that it involves both

\mathbf{q} and Ψ . Let $C(\Psi) = \frac{d\mathbf{q}}{d\Psi}$, which is the water capacity, and assuming that the

relationship between \mathbf{q} and Ψ is $\mathbf{q} = f(\Psi)$, the matric head, or potential Ψ -based form, is given by

$$C(\Psi) \frac{\partial \Psi}{\partial t} = \frac{\partial}{\partial x} \left(K \frac{\partial \Psi}{\partial x} \right) + \frac{\partial}{\partial y} \left(K \frac{\partial \Psi}{\partial y} \right) + \frac{\partial}{\partial z} \left(K \frac{\partial \Psi}{\partial z} \right) - \frac{\partial K}{\partial z}. \quad (2.5)$$

Defining the hydraulic diffusivity D as

$$D = K \frac{d\Psi}{d\mathbf{q}}, \quad (2.6)$$

then equation (2.4) can also be written with \mathbf{q} as the dependent variable as

$$\frac{\partial \mathbf{q}}{\partial t} = \frac{\partial}{\partial x} \left(D \frac{\partial \mathbf{q}}{\partial x} \right) + \frac{\partial}{\partial y} \left(D \frac{\partial \mathbf{q}}{\partial y} \right) + \frac{\partial}{\partial z} \left(D \frac{\partial \mathbf{q}}{\partial z} \right) - \frac{\partial K}{\partial z}, \quad (2.7)$$

which is known as the \mathbf{q} -based form of RE. This diffusive form was advocated by Hillel [1971] for the purpose of creating an analogy to the equation of diffusion of heat by conduction, for which some solutions are already available [Carslaw and Jaeger, 1959; Crank 1956].

In all these three forms of RE, the second order derivative terms on the right hand side represent the diffusive effect due to capillarity while the first order space

derivative term represents the convective term due to gravity. Whether the diffusive or convective term predominates, depends on the initial and boundary conditions and on the stage of the process considered.

Although RE is widely used in simulation of soil water phenomena, its limitations must be carefully considered before any simulation is to be carried out. These limitations are manifested through its simplifying assumptions:

1. The soil is isotropic, homogeneous and rigid.
2. The water is incompressible.
3. The flow is isothermal, so conservation of energy can be ignored.
4. The effect of the air phase on the water flow is negligible.
5. Darcy's law is equivalent to the Navier-Stokes equations when the inertia term is negligible [Phillip 1970], implying that the flow velocity and pore size must be sufficiently small. The flow would be laminar under these conditions.
6. The theory applies only to nonswelling soils [Philip, 1969].

RE is an important equation, although it is well known that significant theoretical questions remain unresolved about its adequacy for describing unsaturated flow [Gray and Hassanizadeh, 1991a, b; Miller et al., 1998a]. Currently, it is the most frequently used equation for simulating water flow problems through soils.

2.2 HYDRAULIC CONDUCTIVITY MODEL

Ross [1992] stressed that the simulation of unsaturated flow in porous media is critically dependent on an adequate representation of the hydraulic properties, i.e. $K(\Psi)$ and $\mathbf{q} = f(\Psi)$. The relationship between \mathbf{q} and Ψ displays strong hysteresis effects, with marked differences between the wetting and drying properties. The hysteresis between Ψ and \mathbf{q} is still a major concern, although there may also be some hysteresis effect between K , the conductivity, and Ψ . The hydraulic conductivity varies

with factors such as soil porosity, pore size, and water temperature. [Hillel, 1971]. Various empirical equations have been proposed for the nonhysteretic relationship $K(q)$. One of the simplest functional forms for the hydraulic conductivity is

$$K(q) = K_s \left(\frac{q}{q_s} \right)^n, \quad (2.8)$$

where K is the hydraulic conductivity of the water content,
 K_s is the saturated conductivity,
 q_s is the saturated volumetric water content,
 n is the power law constant or index, an empirical constant.

Brooks and Corey [1964] and van Genuchten [1978] proposed somewhat more elaborate relationships that more adequately fit retention curves for a wide variety of soils. Between the two models, the Van Genuchten hydraulic conductivity model is the most commonly used by soil physicists and will be fully described in chapter 3. More complex functions or sums of the usual functions have been used for aggregated soils, soils with macropores and, in general, for material whose pore size distributions do not have a simple form [Ross, 1992].

In all these empirical equations, the most important parameter is the power law index, which controls the steepness with which the conductivity increases with decreasing suction or with increasing water content.

2.3 HYSTERESIS MODEL

Hysteresis is crucial in soil physics, as it consolidates the influence of structure, porosity, pore-size distribution, and adsorption on the state of soil water. The state of soil water and its variations in the soil profile, in turn, induce the direction and influence the rate of soil-moisture movement and thus the uptake of water by plants. [Hillel, 1971].

Research (both experimental and numerical) has shown that hysteresis is of fundamental importance in the modelling transient flows in saturated-unsaturated porous media [Hoa et al., 1977; Dane and Wierenga, 1975; and Tzimas 1979]. Hysteresis can be defined as an irreversible cycle, which cannot be reversed along the same path if the direction of wetting (or drying) is changed, for example a change in suction pressure due to a change in soil wetness [Everett and Whitton, 1952].

The hysteretic water content pressure head relationship for a porous material is required in the solution of the water flow equation in order to predict behaviour of a given flow system. In the past, this phenomenon was generally ignored in the theory and practice of soil physics. In the case of monotonic wetting (e.g. infiltration) or drying processes, this is justifiable due to its monotonic nature. However, in the case of composite processes in which both wetting and drying occur simultaneously or sequentially in various parts of the soil profile (e.g. redistribution, drainage and fingering), hysteresis may be important in changing the dynamic and static properties of the soil. [Poulovassilis 1969, Dane and Wierenga, 1975; Watson et al., 1995; Si and Kachanoski, 2000; and Nieber et al., 2000]. In practical applications, hysteresis is usually ignored, as it is cumbersome and CPU intensive to implement hysteresis effects in soil-water models.

However, Braddock et al. [2001] have developed a general formula for applying the Parlange [1976] hysteresis model to any soil water characteristics function, e.g. the van Genuchten model [van Genuchten et al., 1980]. The Parlange [1976] hysteresis model requires knowledge of only one scanning curve. Consider the repeated wetting and drying of a soil. There will be switch points or values of Θ and Ψ , for each spatial point in the soil, where a wetting phase switches to a drying phase, and vice versa. The main wetting and main drying curves represent an envelope within which this switching

occurs. The hysteresis model of Parlange [1976] for this repeated wetting and drying can be generalized to the form

$$\Theta_{d,p} = \Theta_{w,p} - (\Psi - \Psi_{d,p}) \frac{d\Theta_{w,p}}{d\Psi}, \quad (2.9)$$

and

$$\Theta_{w,p} = \Theta_{w,p-1} - (\Psi - \Psi_{d,p-1}) \left[\frac{d\Theta_{w,p-1}}{d\Psi} \right]_{\Psi = \Psi_{w,p}}, \quad (2.10)$$

where $\Theta_{w,p}(\Psi)$ is the p^{th} wetting curve starting at the switch point $\Psi_{w,p}$ and ending at the switch point $\Psi_{d,p}$, i.e., the curve is defined for $\Psi_{w,p} \geq \Psi \geq \Psi_{d,p}$. Now $\Theta_{d,p}(\Psi)$ is the p^{th} drying curve starting at the switch point $\Psi_{d,p}$, and ending at the switch point $\Psi_{d,p+1}$, i.e., $\Psi_{w,p+1} \geq \Psi \geq \Psi_{d,p}$. The switch points are denoted by $\Psi_{w,p}$ which is the switch on the $p-1$ drying curve and denotes the ending of the $p-1^{\text{th}}$ drying phase, and the start of the p^{th} wetting phase: and the switch point denoted by $\Psi_{d,p}$ on the p^{th} wetting curve denotes the ending of the p^{th} wetting phase, and the start of the p^{th} drying phase.

It should be noted that (2.10) is an algebraic recurrence relation that uses the ($p-1$) wetting curve and the switch points, to give the p^{th} wetting curve. Where the main wetting curve is given, (2.9) is also an algebraic recurrence equation for the main drying curve ($p=0$). Where the main drying curve is known, then (2.9) is a first order differential equation for $\Theta_{w,0}$, which is the main wetting curve. Once the differential equation is solved, all subsequent wetting and drying curves can be found by algebraic recursion. These simple algebraic functions are readily programmed into a computer to provide scanning curves to any finite order, and are ideal for use in simulation models, which seek to incorporate hysteresis.

2.4 CURRENT NUMERICAL METHODS IN SOIL PHYSICS

The use of numerical methods to solve water flow problems has experienced rapid development during the past decade, as the high-speed digital computer has become more accessible. In this section, contemporary numerical methods in soil physics for combating steep wetting front infiltration will be reviewed. In particular, the difficulties in achieving such solutions due to the strong nonlinearities of the equation of the flow and the sharp pressure gradient in the wetting front will be discussed, along with the procedures that minimize such difficulties.

2.4.1 ITERATIVE METHODS

Substantial research of implicit iterative, mass-conserving numerical methods has occurred in the last few decades, with the finite difference or finite element method being used for the spatial and temporal discretization. In fact, the Picard Iteration (PI), the Modified Picard Iteration (MPI) and the Newton Iteration (NI), also known as Newton-Raphson Iteration, are far and away the most frequently used methods to resolve the nonlinearities in RE [Forsyth et al., 1995]. While PI and MPI converge linearly, NI converges quadratically. Moreover, NI offer greater robustness for highly nonlinear problems, but at considerable additional computation [Huyakorn and Pinder, 1983]. Recently, Lehmann and Ackerer [1998] found that a combination of the modified Picard and Newton schemes is more efficient than either the modified Picard or Newton scheme.

Several techniques have been proposed in the literature to enhance the performance of these iterative schemes. These enhancements comprise techniques for obtaining a mass-conserving solution, and techniques for achieving convergence with the highly non-linear RE. It should be noted that these ‘mass conserving’ methods still

generate numerical errors, and that the mass is not fully conserved. However, they are generally more accurate in relation to mass conservation, through clever discretisation.

While a solution with a significant mass balance error is always inaccurate, a mass-conserving solution does not necessarily guarantee a good approximation [Celia et al., 1990]. However, a mass balance error is frequently used as a performance measure of a numerical scheme [Milly, 1985; Celia et al., 1990]. Berg [1999] stated that these mass-conserving schemes represent a major step forward that allowed much larger time steps, while still achieving a good numerical approximation for the solution of RE. Moreover, Huyakorn and Pinder [1983] advocated that these mass-conserving schemes are most suitable for an initially dry homogeneous soil.

Milly [1985], Cooley [1983] and Rathfelder and Abriola [1994], found that the proper evaluation of the capacity coefficient C can achieve good global mass balance for certain types of problem. This is achieved by a discretized chain rule expansion of the storage term. Then the equivalence in the storage term expansion is maintained in finite difference models when C is computed with a standard chord slope approximation.

Huang et al. [1996], Kirkland et al. [1992], Celia and Binning [1992], Ross [1990], Hills et al. [1989], Milly [1985] and Neuman [1973] noted that when solving the RE, the form of the governing PDE is an important factor in the development of a numerical approximation in an iterative scheme. In particular, the Ψ -based form of the RE is usually non-mass conserving (depending on the numerical methods used) and has very steep wetting fronts (in term of Ψ) for infiltration into an initially very dry soil, while the mixed and \mathbf{q} -based forms are superior at conserving mass. Furthermore, the \mathbf{q} -based formulation is insensitive to a dry initial condition. However a particular drawback of the \mathbf{q} -based formulation is that it cannot be used where the solution domain becomes saturated. However, the Ψ -based form of the RE is applicable for both

saturated and unsaturated soils, as well as for layered soils. Celia et al [1990] showed that the mixed form of the RE overcomes many limitations of their individual forms. Clement et al. [1994] advocated that the mixed form of RE can be solved in a computationally efficient manner and is capable of modelling a wide variety of problems, including infiltration into dry soil.

Brutsaert [1971] was one of the first authors to use the mixed-form of RE with Newton iteration to deal with steep wetting fronts. Other iteration methods include a modified Picard iteration and a preconditioned conjugate gradient solver both of which have shown some success in combating steep wetting fronts for saturated/unsaturated flow [Celia et al., 1987; Bouloutas, 1989; Celia et al., 1990; Celia and Bining, 1992]. In particular, Celia et al.[1987] advocated that the appropriate solution methodology for general unsaturated flow problems is one that is based on the mixed form of RE, and uses a lumped form (diagonalization of the time matrix) in the finite-element formulation. This was especially true for looking at problems of steep moving infiltrating fronts in initially very dry soils. Neuman [1973] and Cooley [1983] also advocated the use of the lumped approach.

However, El-Kadi and Ling [1993] demonstrated that the pressure-based form of the RE, and other forms, could produce accurate and mass-conserving solutions, provided that care is taken in designing the spatial and temporal mesh. They proposed the use of the Courant and Peclet numbers as criteria for estimating the spatial and temporal step. Their study showed that efficient solutions were obtained with these proposed criteria, even for soils that are characterized by a very sharp moisture front.

The other major enhancement of the iterative scheme is in techniques for achieving convergence with the highly non-linear RE. The literature abounds with such enhancements. These include relaxation techniques, over and underrelaxation techniques, chord slope differentiation, a new nonlinear convergence criterion, a

quadratic/cubic line search, and the mixed transform finite element method, which were all mostly proposed in the 1990s.

Paniconi and Putti [1994] proposed the use of relaxation and chord slope differentiation, along with a mixed approach involving the use of Picard iteration to improve the initial solution estimate for the Newton scheme.

Huang et al. [1996] used a new nonlinear convergence criterion in conjunction with the mixed-form algorithm of Celia et al. [1990]. This criterion consists of a term containing the absolute error of volumetric water content, and a term involving the soil water capacity:

$$C^{n+1,m} |\mathbf{d}^m| = |\mathbf{q}^{n+1,m+1} - \mathbf{q}^{n+1,m}| \leq \mathbf{d}_q, \quad (2.11)$$

where n denotes the time level,

m denotes the iteration level

$C^{n+1,m}$ is the water capacity at $n+1$ time and m iteration,

\mathbf{d}_q is a tolerance on \mathbf{q} ,

$|\mathbf{d}^m|$ is the absolute error at m iterations, and

$\mathbf{q}^{n+1,m+1}$ is the volumetric water content at $n+1$ time and $m+1$ iteration.

Their results indicated that reductions in computational effort using their proposed criterion were particularly significant for infiltration into relatively coarse-textured soils, infiltration in initially dry soils, and multidimensional infiltration problems.

Baca et al [1997] developed a mixed formulation of the Richards equation but expressed in terms of a partitioned transform. An iterative finite element algorithm is derived using a Newton-Galerkin weak statement. This new method was called the mixed transform finite element method. The mixed transform finite element method is shown to converge faster than the modified Picard method in a number of cases and to accurately represent pressure head and moisture content profiles with very steep fronts.

Williams and Miller [1999] showed that a quadratic/cubic line search [Dennis and Schnabel, 1996] with NI is much more robust than NI, whereby the global convergence properties of NI methods is greatly improved by this technique. They called this new procedure a NI-line search (NILS) approach.

2.4.2 NONLINEAR TRANSFORMATION

The use of transformations with iterative, implicit, mass-conserving numerical methods has been shown to result in more efficient solutions, especially for problems with very high-pressure gradients near the wetting fronts. These transformation methods for solving RE, which seek to reduce the high-pressure gradients to lower gradients in a transformed variable, have existed for more than three decades. [Rubin, 1968; Raats and Gardner, 1974; Haverkamp et al., 1977; Baca et al., 1978; Vauclin et al., 1979; Ross, 1990; Pan and Wierenga, 1995; Forsyth et al, 1995; Williams et al., 2000].

The current transformation approaches include the use of the inverse hyperbolic sine transform [Ross, 1990], the K transform [Ross, 1990], the volumetric fraction of the aqueous phase [Kirkland et al, 1992], rational transform functions [Pan and Wierenga, 1995 and 1997], variable switching methods [Forsyth et al, 1995], as well as the integral and water-content-based (IT2) transformation [Williams et al., 2000].

One of the oldest transformations is the Kirchhoff integral transform (IT1) [Rubin, 1968; Raats and Gardner, 1974; Haverkamp et al., 1977; Vauclin et al., 1979; Redinger et al., 1984; Campbell, 1985]. Similar to the q -based method, the Kirchhoff transformation (IT1) depends on the soil hydraulic properties, which causes difficulties in heterogeneous and/or hysteretic soils. Thus, IT1 is restricted to homogeneous media, but it can be adapted to layered and gradational media by adding a flux balancing correction [Ross and Bristow, 1990]. It also directly reduces the nonlinearity of the

conductivity terms in RE, and, as a result, is effective in reducing the total number of iterations required for a solution [Williams et al, 2000]. Generally the integral in the Kirchhoff transformation is difficult to evaluate, and special numerical interpolation methods are needed [Ross 1992].

Ross [1990] uses an inverse hyperbolic sine transform for the matric potential that allows large spatial increments even in dry, inhomogeneous soil. This transform appears to be the first transformation that is independent of the soil properties, and it is applicable under saturated conditions and with inhomogeneous soils. It has two arbitrary parameters and both are needed for optimizing the transform for a given problem [Williams et al., 2000].

Kirkland et al. [1992] defined another new variable transform to RE. This variable is a linear function of water content if the soil is unsaturated ($\Psi < \Psi_0$) and a linear function of pressure head in saturated or near saturated soils ($\Psi \geq \Psi_0$). This transform variable is applicable to variably saturated soils and is reported to have very good computational speed, particularly for relatively dry initial conditions [Kirkland et al, 1992]. However, the method is CPU intensive with respect to required memory particularly decreases for heterogeneous and/or hysteretic soils. Also the numerical coding of this method is complex.

Another well-known transformation method is the K transform. The K transform excludes the conductivity from the gradient term in the flow law, hence hysteresis cannot be incorporated in the $K(\Psi)$ relation. It is also restricted to homogeneous soils. However by making the appropriate correction due to the change in soil properties with depth, one can still capitalize on the numerical advantages of using the K transform to solve RE for water flow in layered and gradational soils (different types of soil) [Ross, 1990].

Pan and Wierenga [1995,1997] presented a simple nonlinear transformation, called the rational transform, of Ψ to the variable $P_t = \Psi / (1 + b\Psi)$, with b a transformation constant independent of soil type. They use the modified Picard method to solve the resulting forms of RE. This method of transformed pressure Ψ -based approach is numerically robust for all cases of variably saturated, heterogeneous media, and Dirichlet or Neumann types boundary conditions. The results show that the new method offers excellent CPU time performance and is not affected by complicated heterogeneous and hysteretic media. Most of all, the P_t transformation is easy to incorporate into existing Ψ -based codes. However, Lee [1996] applied this P_t transformation to the RE with an ODE/MOL approach, and concluded that its effect on the solution is marginal at best, and it is not effective in handling steep wetting fronts.

Forsyth et al [1995] introduced a new transformation using a variable switching technique. It is similar to the Kirkland et al (1992) transform in that it is based on switching between dependent variables q and Ψ , yet it differs from Kirkland et al (1992) in that it does not define a continuous dependent variable over the entire domain [Williams et al., 2000].

Williams et al. [2000] defined a new transformation called IT2, which is a linear combination of volumetric water fraction of the aqueous phase and integrated hydraulic conductivity terms. They showed that IT2 can significantly improve solution efficiency and robustness compared to standard solution approaches. Furthermore, IT2 compares favorably with existing transforms in terms of efficiency and robustness.

It is clear that most of the above transforms involve arbitrary parameters. Hence, the efficiency and robustness of a particular transformation depends on choosing the right or optimal value of the transform parameters. In a recent report [Williams et al., 2000], a rigorous optimization of the transform parameters over a wide range of test

problems was carried out. Their results indicate that the IT2 transformation was the best method for solving Richards' equation with a steep wetting front in an initially dry media. Furthermore, IT2 seems to perform better than the rational transformation in many cases. Thus the IT2 transform is more efficient and stable, but it is also a lot more complex and cumbersome to program than the rational transform.

2.4.3 ADAPTIVE SCHEMES

Recently, the application of MDCS (a moving, deforming coordinate system) has attracted the attention of researchers all over the world. It is the state of the art in numerical modelling.

In general, some measure of the solution of the PDE, i.e., steepness, error, volume variation etc., is minimized using a transform of the spatial grid. The first use of MDCS in modelling water flow in soil physics was by Dane and Mathis [1981]. A recent application of MDCS in soil physics can be found in Cox et al. [1994] in which moving finite elements are used to model one-dimensional infiltration in an unsaturated soil. The theoretical beauty of the method is that one may concentrate computational attention when and where it is most appropriate, also possibly greatly reducing the effective magnitude of terms which cause trouble, e.g. 'gradient' terms that have very high or near infinity values. These high gradient values could cause numerical difficulty in computation. "Thus even very steep fronts may be represented well, with no oscillations, using space and time step sizes well beyond conventional (Peclet, Courant number) constraints." commented O'Neill [1981]. "This constitutes orders of magnitude savings in effort over what the application of conventional constraints would require."

However, the moving grid method or adaptive scheme has limitations when applied to layered systems or used with time-varying boundary conditions. The method

was found to be less mass-conservative than conventional fixed-grid formulations [Huang et al., 1996].

The use of adaptive schemes or MDCS is scarce in the literature of soil physics, whereas in other fields, e.g. fluid mechanics, meteorology and dynamics engineering, it is popular and common. The ability to capture the behavior of shocks and steep fronts is the motivation for these adaptive schemes. It appears that the self-adaptive approach to computational resolution in space and time is the solution to these concerns.

The use of adaptive schemes in one spatial dimension is well understood and current research is largely carried out in higher spatial dimensions. Over the past several years, a great number of efficient adaptive grid methods and various sophisticated techniques have been developed for 1-D PDEs [see Furzeland et al., 1990; Hawken et al., 1991; Carroll and Stewart, 1996; and Wouwer et al., 1998]. In particular, moving grid methods have been applied successfully to every class of PDE [e.g. Carlson and Miller, 1994; Huang et al., 1994; Miller and Miller, 1981; Zegeling, 1992 and 1993; and Biswas et al., 1993]. The interested reader is referred to the above mentioned papers and the references therein.

Adaptive schemes in higher spatial dimension, even for two-space dimensions, are far less trivial than in 1 D. Among other difficulties are the control of skewness and overlap in the grids generated in the high dimensional formulation. A lot of work has been conducted on linear systems of 2-D PDEs [Thompson, 1985; Mulholland et al, 1997], but much less has been done on nonlinear 2-D PDEs [Oden, 1989]. In particular, we are interested in the application of an adaptive scheme for a 2-D nonlinear PDEs. The following is some review of the work carried out by these researchers to deal with various PDEs, whose solutions contain steep moving gradient solutions.

Zegeling [1998] studied evolutionary non-linear PDE models using an r-refinement technique (increase nodal densities in areas of large errors by relocating

among a fixed number of nodes) via the equidistribution principle. Equidistribution involves a transformation of the original equation, based on some known quantity of the solution, into an equation and transformed spatial variable that has equal error distribution on the uniform mesh with the same number of nodal points. Zegeling [1998], states that whilst MFD (moving finite difference) may be used to efficiently approximate solutions of convection-diffusion-reaction systems in one space dimension having steep moving transitions, in 2D the method needs further investigation. A MFE (moving finite element) may be used for reaction-diffusion systems in ‘any’ number of dimensions (with the possibility of grid-distortion when a first-derivative term is present in the PDE model). In general applications, Zegeling [1998] advocates an h-r-refinement technique (increase nodal densities by refining and relocating the mesh in an area of large errors) for both of MFD and MFE methods in order to deal with possible grid de-generations. Other researchers including Baines [1994], and Ainsworth and Senior [1998] have also given the same recommendations.

Brackbill and Saltzman [1982] used variational principles to minimize a functional that contains a measure of grid smoothness, orthogonality and volume variation. Forming a variational principle using linear combinations of the integral measures yielded a system of PDEs. These equations were solved numerically using a relaxation algorithm. In that paper, singular problems in one and two dimensions were successfully solved, and moreover, the effect of each term in the variation principle on the mesh can be controlled. They had also applied their method successfully to time-dependent problems [Brackbill and Saltzman, 1980]. In addition, Dietachmayer and Droegemeier [1992] have successfully applied the technique to model the evolution of a 2-D nonlinear buoyant thermal in a neutral environment. Although this method gives remarkable results, the generating equations are very complicated, and computational time is extensive [Thompson, 1985].

Huang and Russell [1999] developed the moving mesh partial differential equations (MMPDE), which control the mesh movement by solving a system of parabolic equations. MMPDE uses a mesh generation functional involving the mesh adaption to the underlying physical solution, mesh alignment to some known vector fields, and mesh orthogonality controls. They advocated that MMPDE is simple and relatively easy to program, especially for structured grids. A brief and clear description of the MMPDE approach for structured mesh movement is given in Cao et al. [1999a].

Regardless of the methods used, the key of success in adaptive schemes is the use of proper monitor functions [Cao et. al., 1999a,b]. These monitor functions consist of different error measures, and many error measures have been proposed in the literature, often chosen heuristically, with little or no justification [Hawken et. al, 1991]. If error measures do not contain all relevant error information, grid adaptation might not lead to any improvement of the solution [Warren et. al., 1993]. The design of good error measures is an active research area. Thus, one has to choose the monitor function prudently.

2.4.4 MOL AND STIFF SOLVERS

The Method of Lines (MOL) dates back to 1930 due to the work by Rothe [1930]. The development of MOL has been rapid during the past 30 years and many applications have been reported [Schiesser 1994]. Recently, the adaptive MOL has become the modern version of the method of lines [Wouwer et al., 2001].

In the classical finite difference method, a Partial Differential Equation (PDE) is discretized with respect to all variables, resulting in a system of algebraic equations. In MOL, the PDE is discretized with respect to all spatial variables but one variable (usually time) is left in continuous form, resulting in a system of ODEs. The right hand side of the ODEs is generated from the spatial discretization. An ODE integrator is then

used to integrate the system to obtain the solution at each spatial grid point in the domain. The evolution of the numerical solution can be represented along time lines at each spatial grid point. The line marches forward in time from each spatial grid point, hence the name the method of lines.

There is no need for linearity in the discretized spatial derivatives in an ODEs system, thus MOL applies equally well to both linear and nonlinear problems. And also any discretization method can be used for the space variables. Boundary conditions can easily be formulated by algebraic equations leading to the use of differential-algebraic equation (DAE) or ODE software. Furthermore, the fruits of decades of research and development of mathematical software for solving initial value problems for ODEs and DAEs can be utilized fully with MOL. Moreover, the following properties in the system of ODEs or DAEs (created by the MOL) have been observed:

1. The right side is often “banded” and this structure can be exploited in the numerical scheme.
2. The system of ODEs or DAEs can be very large for 2 or 3 space variables. Some 20 to 50 lines in 1-D corresponds to 400 to 2500 lines in 2-D, and in 3-D this corresponds to 8000 to 125000 lines [Rice, 1993], and
3. The system of ODEs or DAEs is often stiff and requires a special stiff solver.

Surprisingly in the field of soil physics, the application of MOL on a transient flow model has been uncommon. However, in recent years, the interest in using the MOL for modelling water infiltration problem is gaining momentum. Recently, Tocci et al. [1997], Kelley et al. [1998], Miller et al. [1998b], Williams and Miller [1999] and Williams et al [2000] applied the MOL to the pressure form of RE (with fluid compressibility) and solved the resultant DAE with variable-order variable-stepsize DAE solvers.

Tocci et al. [1997] proposed the DAE/MOL approach to solve RE for two difficult 1-D sharp-front models. Compared with the traditional approaches (Newton Iteration-line search approach and Modified Picard Iteration), the DAE/MOL approach was much more efficient in terms of high accuracy results. The method also achieved an excellent mass balance. However their DAE/MOL approach required a standard stiff integrator, DASPK, to be modified to yield good results. Modifications were made to three internal parameters (that are not accessible through the standard user interface): (1) decreasing the allowable size of the initial time step; (2) decreasing the error tolerance limit for the nonlinear solver; and (3) forcing the reformation of the Jacobian matrix at the beginning of each new time step. With the modified DASPK, they show that the DAE/MOL implementation can give solutions to RE that are accurate, have good mass balance properties, explicitly control temporal truncation error, and are more economical than standard approaches for a wide range of solution accuracy.

Kelley et al. [1998], motivated by the numerical results made in Tocci et al. [1997], advocated further modification of the DAE solver for problems that develop sharp moving fronts in variably-saturated porous media. They observed that with the sharp moving front problem, the corrector iteration used in many integrators could terminate prematurely, leading to incorrect results. This can be solved by tightening the tolerances in the DAE solver. However, they advocated that it is more efficient to modify the termination criteria of the nonlinear solver and/or recompute the Jacobian matrix more frequently. Of these two, recomputation of the Jacobian matrix is the more important. They proposed a criteria based on an estimate of the norm of the time derivative of the Jacobian for recomputation of the Jacobian and a second criteria based on a condition estimate for tightening of the termination criteria of the nonlinear solver.

Miller et al. [1998b] developed an accurate, efficient and robust variably saturated flow simulator for 1-D problems with a sharp infiltration front between an unsaturated and a saturated zone. These 1-D problems are characterized by discontinuities in the derivative of the specific moisture capacity and relative permeability as a function of capillary pressure, which often cause convergence difficulties using standard numerical approaches. The flow simulator uses an integral representation of mean interblock values [Schnabel and Richie, 1984; Warrick, 1991; Zaidel and Russo, 1992] approximated by Hermite polynomials and the DAE/MOL approach [Tocci et al., 1997] to solve the highly non-linear RE.

Williams and Miller [1999] incorporated the integral and water-content-based (IT2) transformation into the Miller et al. [1998b] solution, which improved its ability to model difficult problems, especially those that give rise to sharp fronts that propagate through the domain. Their results showed that in most cases, transforming the dependent variable led to more efficient solutions than the untransformed approaches, especially as the pore-size uniformity increased. Moreover, the selection of the optimal value of the parameter in the transformation may be simplified due to the relative insensitivity of the solver performance to deviations from the optimal parameter value.

It is clear that the DAE/MOL approach has great potential for the applications of modelling flow and transport phenomena: multidimensional problems, two- and three-phase flow problems, and coupled flow and transport problems. However, more research is needed for the development of a fully efficient and robust DAE/MOL simulator [Tocci et al., 1997].

2.5 SOFTWARE LIBRARIES, MATLAB AND TEMPLATE IN NUMERICAL COMPUTATION

John Rice commented [Rice, 1993]: “Forty years ago everyone was writing their own code for the logarithm. That happens rarely now. Such computations are incredibly

more reliable, and some real progress has been made.” This statement has clearly indicated that “re-inventing the wheel” is out. The synergetic effect of adapting high quality, efficient, accurate, reliable, robust, easy to use and easy to maintain software libraries is the future of numerical computation. Software libraries such as the Numerical Recipes, IMSL, NAG, LAPACK, LINPACK, and MATLAB are built upon long established foundations of mathematical and computational knowledge and are highly reliable. Other less tested sources are journals that publish individual computer programs, e.g. ACM Transactions on Mathematical Software, Applied Statistics, BIT, The Computer Journal, and Numerische Mathematik. Another advocator for using “Software Libraries” is Köckler [1994]. He was astonished that engineers, physicists or mathematicians usually prefer to write their own programs rather than using existing reliable software. He stated two theses to explain the reasons for this situation:

First thesis:

‘Software libraries and program packages are either reliable or user-friendly, but not both.’

Second thesis:

‘Every program with more than 50 lines is wrong.’

Kockler [1994] recommended the use of libraries like NAG and IMSL, containing hundreds of routines, which are regularly tested and updated. Regular usage tests and retests the library codes in a wide variety of applications. He pointed out that the documentation for each of these two libraries fills at least five binders and some preparation is needed even when using one of the simpler routines. This is the major reason that deters many potential users, who do not see that the effort is rewarded with error-free programs, and either reliable results or warnings. He also stresses that by using library routines within our programs we will attempt to make the second thesis less and less correct.

These software libraries provide “off-the-shelf” programs as tools for practicing scientists and engineers, whereby the tools have been known to routinely solve a variety of problems reliably and accurately. With the availability of these software libraries, the next question we would ask is what programming environment should we use it with? We would need a programming environment that is easy to incorporate these software libraries and to customize the black-box programs to our specific need.

MATLAB is a widely used modern programming language with adequate constructs for scientific software. Although the programs are interpreted, not compiled, the many built-in functions are in compiled form and a great deal of time is saved. Also the vectorizing capability, sparse matrix operations and object-oriented structure of the MATLAB architecture can be used. As compared to the conventional method of programming using Fortran, Pascal or C languages, MATLAB programming is efficient, easily debugged and easy to implement. Also MATLAB has a suite of state-of-the-art ODE integrators that are conveniently available for the purpose of handling stiff systems of ODEs and user-friendly graphical interfaces for displaying the numerical results.

Most importantly, the MATLAB programming language is oriented towards matrix computation in favor of vectorization and its storage system is handled in such a way that obviates the complicated declarations and allocations found in most scientific computing. Another important issue is the availability of the NAGware Gateway Generator for interfacing a Fortran 77 program to MATLAB and the MAGNUM collection of NAG Library routines are provided with MATLAB interfaces. In fact any C, FORTRAN 77 or 90 program can be interfaced with MATLAB easily by creating a “Gateway” file that links the original compiled C or Fortran program to MATLAB. Most of all, the usefulness of MATLAB lies in its open-system philosophy whereby one can learn and modify the codes found in its extensive numerical libraries. All these factors

have greatly simplify its use with current software libraries and reduced the time required for the development of software.

Another recent development in numerical computation is the use of a “Template” to facilitate the usage of numerical algorithms, i.e. “Templates for the solution of linear systems: building blocks for iterative methods”, Barrett et al. [1994]. They defined a Template as a description of a general algorithm rather than the executable object code or the source code more commonly found in a conventional software library. They also advocated that a Template offered whatever degree of customization the user may desire; a Template is not language specific and is readily translatable into the target language such as MATLAB, FORTRAN, C, etc. Thus, a Template is general and reusable, reflecting the in-depth knowledge of a specific numerical technique.

A Template is frequently used in industrial software packages, whereby the user is guided by a user-interface driven Template to define a problem and subsequently the problem is solved automatically without the intervention of the users. Packages such as the MATLAB PDE toolbox, ELLPACK and Diffpack are typical examples. Other packages that are less user-friendly include Fastflo, BNALib and VECFEM (Vectorized Finite Element Method). Usually these other packages use a high level language to define the problem and a structured instructional format to initiate the computation and the rest is mostly automatic to obtained good numerical computation.

2.6 CONCLUSIONS

The accurate modelling of RE with steep wetting fronts moving in very dry soils, represents a great leap in the field of numerical approximation of nonlinear PDE's. This is because the problem represents a most ubiquitous and challenging modelling of the interaction between convective and diffusive processes. As seen from

the literature review, methods that are up to this task are usually very complicated for the novice. Simpler methods, such as the DAE/MOL or ODE/MOL approach look promising and are worth further investigation. Numerical methods that are simple to implement, efficient and robust is the solution to the challenges cited in this thesis.

In order to meet the above aims, the vectorized ODE/MOL will be used as the base numerical method whereby new methods are developed and investigated. The adoption of the vectorized ODE/MOL is based its simplicity and versatility for incorporating other methods into its implementation. Chapter 4 will give a detailed description of the vectorized ODE/MOL using MATLAB programming in a form of a Template and the use of a numeric library. Sufficient details are given so that the readers could implement the method themselves if chosen to do so. Moreover, all the numerical programs used in this thesis are included in the attached CD-ROM for your perusal and improvement; with the condition that it is not used for commercial purposes and due reference is made.

Models Used

Discovery consists of seeing what everybody has seen, and thinking what nobody has thought.

Albert Szent-Gyorgyi

3.1 INTRODUCTION

An important part of the development of new methods is systematic testing [Cowell, 1984]. He advocated that test problems should be chosen to check the performance of the software on a wide range of typical problems and also on types of problems likely to prove difficult. In this thesis, three groups of test problem are used:

- a) Burgers' Equation where there are analytical solutions for a range of problem,
- b) Analytical solution of RE for constant flux infiltration in very dry soil, using constitutive relations of Sander et al. [1988], and
- c) 2-D Model of water infiltration using the van-Genuchten-Mualem model.

To avoid repetition and for future reference, most of the models and all measures of numerical performance used in the thesis are discussed in this section.

3.2 BURGER'S MODEL OF ADVECTIVE-CONVECTIVE EQUATION

Burgers' equation can be considered as a special case of the one-dimensional Richards' equation for vertical flow. It is a simplified representation of vertical water flow through the vadose zone that retains many of the nonlinear characteristics of Richards' equation and its solutions.

Hills and Warrick [1992] took the soil water diffusivity as a constant and the unsaturated hydraulic conductivity as proportional to the square of the reduced water content, thus reducing RE into a form of BE. They showed that infiltration into a dry soil using BE gives developing wetting front profiles similar to those observed for the more general forms of Richards' equation. In a recent paper, Warrick and Parkin [1995] derived the BE as limiting cases of Richards' equation using diffusivity and conductivity functions from Fujita as extended by Broadbridge and White [1988].

In this thesis, the well-known Burgers' Equation [Schiesser, 1991], a simple model incorporating both diffusion and advection processes, is used for the numerical experiments. For Burgers' Equation (BE) in the form of

$$\Omega_t = -\Omega \Omega_z + n \Omega_{zz}, \quad (3.1)$$

subject to the boundary conditions

$$\Omega_z(0, t) = \Omega_z(L, t) = -(1/2n)(1 + E)^{(-2)}, \quad (3.2)$$

where $E = \exp(0.5z/n - 0.5t/n)$,

has the analytic solution

$$\Omega(z, t) = 1 / \{1 + \exp(0.5z/n - 0.5t/n)\}. \quad (3.3)$$

The initial condition for the system is obtained from (3.3) evaluated at $t = 0$. The value of the diffusivity n , determines the nature of Burgers' equation; for $n \geq 1$, it is predominantly parabolic (a relatively easy problem); for $n < 1$, it is highly hyperbolic and is a relatively difficult problem to solve. Thus with a small n , it becomes a stringent test problem for which the solution exhibits steep moving fronts or shock waves.

3.3 ANALYTICAL SOLUTION OF 1-D CONSTANT FLUX INFILTRATION USING RICHARDS' EQUATION

Sander et al. [1988] provides a convenient solution benchmark for the rigorous testing of numerical schemes for constant flux type infiltration in unsaturated porous materials. It is applicable to any initial water content of the porous material, and residual water content.

Sander et al. [1988], in developing their analytical solution, use a form of RE involving reduced variables, where the Θ -based form of RE is

$$\Theta_t = (D(\Theta)\Theta_z)_z - K_z, \quad (3.4)$$

where $\Theta = \frac{(\mathbf{q} - \mathbf{q}_i)}{(\mathbf{q}_s - \mathbf{q}_i)}$, $0 \leq \Theta \leq 1$, where \mathbf{q} is the actual water content, \mathbf{q}_i and \mathbf{q}_s are the actual initial and saturated water contents, respectively, z is the vertical dimension, and the subscript z denotes partial derivative. In (3.4), D is the diffusivity and K is the hydraulic conductivity. Note that (3.4) is the reduced form of (2.7) in the vertical space dimension only.

This Θ -based form can be transformed to the mixed form

$$\Theta_t = (K\Psi_z)_z - K_z, \quad (3.5)$$

with the corresponding Ψ -based or pressure form given by

$$C\Psi_t = (K\Psi_z)_z - K_z, \quad (3.6)$$

where $C = d\Theta/d\Psi$ is the specific water capacity. Note that (3.6) is the reduced form of (2.6) in the vertical space dimension only.

The constitutive relations are required to close the above conservation laws, and the relations among Ψ , $\Theta(\Psi)$, and $k(\Theta)$ are specified as

$$\Theta = \frac{e^{a\Psi}}{1 - \mathbf{u} + \mathbf{u}e^{a\Psi}}, \quad (3.7)$$

$$D(\Theta) = \frac{D_o}{(1-\mathbf{u}\Theta)^2}, \quad (3.8)$$

$$k(\Theta) = \frac{K_s(1-\mathbf{u})\Theta}{(1-\mathbf{u}\Theta)}, \quad (3.9)$$

$$\mathbf{a} = \frac{K_s(1-\mathbf{u})}{D_o}, \quad (3.10)$$

where K_s is the saturated hydraulic conductivity, and \mathbf{u} and D_o are constants determined from soil properties.

Sander et al. [1988] also consider the initial and boundary conditions

$$t = 0, \quad z \geq 0, \quad \Theta = 0, \quad (3.11)$$

$$t > 0, \quad z = 0, \quad \Psi_z = 1 - \frac{Q}{K}, \quad (3.12)$$

$$t > 0, \quad z \rightarrow \infty, \quad \Theta = 0, \quad (3.13)$$

where Q and K are the reduced constant surface flux and reduced conductivity respectively

$$Q = \frac{q - K(\mathbf{q}_i)}{\mathbf{q}_s - \mathbf{q}_i}, \quad (3.14)$$

$$K = \frac{k - k(\mathbf{q}_i)}{\mathbf{q}_s - \mathbf{q}_i}, \quad (3.15)$$

and q is the actual constant surface flux.

Using the Θ based form of RE, and the above boundary and initial conditions, Sander et al. [1988] were able to obtain an analytic solution. An attractive feature of the solution is that realistic soil-water characteristics can be described by the specified diffusivity and conductivity functions [Watson et al., 1995]. Furthermore, as with the BE, the solution can be tuned using the value of \mathbf{u} to depict steep wetting front behaviour (nearly a square wave) or a relative gentle wetting front. As \mathbf{u} approaches 1 the form of the Fujita diffusivity (which is the one used to get the exact solution),

approaches the behaviour of a delta function, i.e. it is zero everywhere except at $\Theta=1$. Consequently the wetting front behaves as a square wave, and it is very difficult to handle numerically. With smaller values of \mathbf{u} , less than 1, the moving wetting front is relatively gentle and usually does not pose a problem to the numerical method.

To simulate a shorter depth of infiltration, the boundary condition at $z = \infty$ was not used. Instead, the boundary condition

$$t > 0, \quad z = B, \quad \Psi_z = 1, \quad (3.16)$$

was used and B is the specified depth of infiltration. This boundary condition allows free drainage flow. A value of B was also chosen to ensure that the front is far from $z = B$ at the end of the integration i.e. this new boundary condition will not affect the numerical accuracy. This technique permitted the movement of the wetting fronts into the soil. Comparisons of the analytic and numerical solutions showed that the wetting fronts were relatively unaffected by the bottom boundary. Naturally there was considerable effect from the bottom boundary as the wetting front hit $z = B$. While B=50cm was used for the short-term integration, B=150cm were used for long-term integration. It was found that the solution obtained was comparable to the analytical solution with the semi-infinite boundary condition [as in Sander et al, 1988]. Although the analytical solution is programmed and used in the thesis for the validation of the numerical schemes, the reader is referred to Sander et al. [1988] for the details of the analytical solution.

The relevant material properties used for the model were $\mathbf{q}_s=0.35 \text{ cm}^3 \text{ cm}^{-3}$, $\mathbf{q}_i=0.06 \text{ cm}^3 \text{ cm}^{-3}$, $\mathbf{u}=0.85$, $q=0.08 \text{ cm/min}$, $K_s=0.1 \text{ cm/min}$ and $D_o=2.75862 \text{ cm/min}$. For the above material properties, the model solution depicts a very steep wetting front at the point of influx and this front gradually become less steep with time. Hence, it is a

relatively easy model to solve numerically. For easy reference, we refer to this constant flux infiltration model as model A.

To impose a more difficult model, the parameters $u = 0.99995$, $q = 0.344$, and $D_0=0.5$ were used in the above model. With these values, the solution develops a very steep-moving front. Moreover, the initial phase of the integration of RE could have some numerical difficulties, as the initial water pressure is extremely high. For easy reference, we refer to this second model of constant flux infiltration as model B.

Moreover, the value of $q_i=0.06 \text{ cm}^3 \text{ cm}^{-3}$ used in the analytical solution could not be used in the numerical model of the Ψ -based form of Richard's equation. This is because $\Psi = -\infty \text{ cm}$, when $\Theta = 0$. Following Watson et al. [1995], to overcome this problem, q_i was given a value as closed as possible to $0.06 \text{ cm}^3 \text{ cm}^{-3}$ to avoid excessively large negative values of the corresponding Ψ value. For all cases, a value of $q_i=0.060001 \text{ cm}^3 \text{ cm}^{-3}$ was specified.

3.4 MODEL OF 2-D INFILTRATION IN A VERY DRY SOIL

To verify the numerical techniques developed in the thesis and demonstrate their capability to model steep wetting fronts in a very dry soil, we use an example of constant flux infiltration into a 2-dimensional domain, which basically simulates a gravity-driven fingered flow. The Ψ -based composed form of Richards' equation is adopted to describe the physical process of constant flux infiltration into a 2-dimensional domain, which basically simulates a gravity-driven fingered flow. In 2 space dimensions x , and z , the Ψ -based composed form of Richards' equation is

$$C\Psi_t = \left(K(\Psi_z - 1) \right)_z + \left(K\Psi_x \right)_x. \quad (3.17)$$

where $\Psi = \Psi(x, z, t)$, and subscripts indicate partial derivatives. The van Genuchten-Mualem (VGM) relations are used to close the Richards' equation model (see sections

3.5). The relevant material properties used for this example are $\mathbf{a}=0.1 \text{ cm}^{-1}$, $\mathbf{q}_s=0.45$, $\mathbf{q}_r=0.05$, $n=2.5$, $K_s=72 \text{ cm/day}$.

By imposing a constant flux on the top boundary of a rectangular region and maintaining no flow elsewhere on the boundary, the growing of a ‘finger-like’ structure (gravity-driven) was achieved. Also, the fact that gravity-driven finger-like flow is convection dominated (by the gravity term in the RE) made it a challenging problem to solve. However, Nieber (1996) advocated that for porous media, a small perturbation will grow into a finger and during sequential drainage and wetting the finger will persist, if the water-entry capillary pressure on the main wetting curve is less than the air-entry capillary pressure on the main drainage function. In contrast, for porous media where the water-entry capillary pressures on the main drainage function is greater than the air-entry capillary pressure on the main drainage function, the same perturbation will dissipate by capillary diffusion.

The region consists of a domain of 160 cm width and 80 cm depth. All sides of the flow region were considered to be impervious, except for a region of 24 cm length at the surface where a constant flux was applied. The soil initially has a uniform matric pressure of, Ψ_i , -10^3 cm , -10^4 cm or -10^5 cm of water pressure. This range of initial pressure heads were selected to simulate soil wetness ranges from relatively dry to extremely dry soil. Hence the example simulated water infiltration with a steep moving front in soil with soil wetness ranges from relatively dry to extremely dry. Fig. 3.1 shows the physical region and boundary conditions.

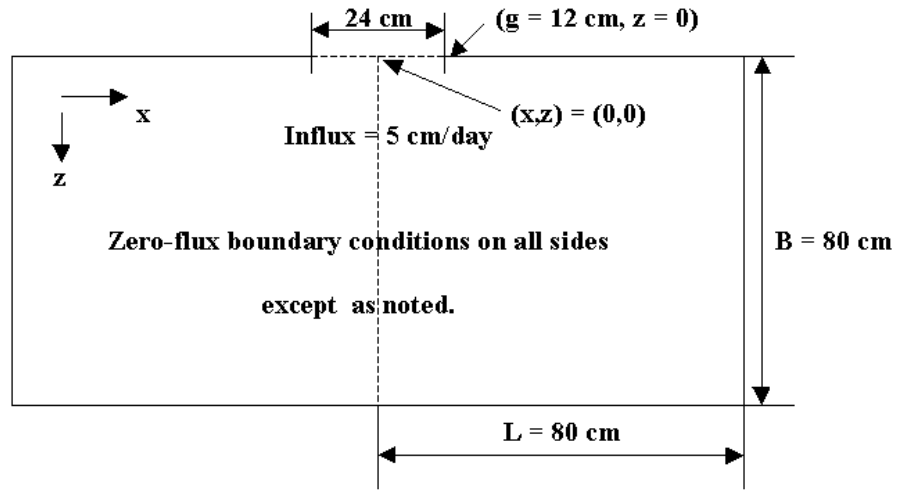


Fig. 3.1 The physical region and boundary conditions.

Since the model is symmetrical, only the right half of the model is modelled, and the following initial and boundary conditions are applied.

$$t = 0, \quad 0 \leq z \leq B, \quad 0 \leq x \leq L, \quad \Psi = \Psi_i, \quad (3.18)$$

$$g \leq x \leq L, \quad \Psi_z(x, 0, t) = 1, \quad (3.19)$$

$$0 \leq x \leq g, \quad \Psi_z(x, 0, t) = \left(1 - \frac{q}{K}\right), \quad (3.20)$$

$$0 \leq z \leq B, \quad \Psi_x(0, z, t) = 0, \quad (3.21)$$

$$0 \leq z \leq B, \quad \Psi_x(L, z, t) = 0, \quad (3.22)$$

where $g = 12$ cm is the size of the region of constant flux on the top boundary.

The model uses the van Genuchten [1980] relations to describe the interdependence of fluid pressures and saturation and the Mualem [1976] relation to describe the interdependence between fluid saturation and relative permeability. We refer to these relations collectively as the van Genuchten-Mualem (VGM) relations. This is the most commonly used constitutive relation to close the RE in the literature

[e.g. Yeh, 1987; Simunek and van Genuchten, 1994; Simunek et al., 1994, 1997]. The VGM equations are

$$S_e = \left[\frac{1}{1 + [\mathbf{a} |\Psi|]^n} \right]^m, \quad (3.23)$$

$$k(\mathbf{q}) = K_s S_e^{(1/2)} \left[1 - (1 - S_e^{1/m})^m \right]^2, \quad (3.24)$$

$$S_e = \frac{\mathbf{q} - \mathbf{q}_r}{\mathbf{q}_s - \mathbf{q}_r}, \quad (3.25)$$

where \mathbf{q}_r is the residual moisture content and \mathbf{a} , n , and m are fitting parameters, with m related to n by

$$m = 1 - 1/n. \quad (3.26)$$

While \mathbf{a} is a parameter related to the mean pore size, the exponent, or n , is a measure of pore-size uniformity. Natural porous media have typical values of n which range between 1.0 and 2.0. These values denote a highly non-uniform pore-size distribution media. [Kool et al., 1985]. For such media, the VGM relations are not smooth, i.e. C is not differentiable at $\Psi = 0$. This non-differentiability can lead to the failure of the numerical scheme which relies on these relations [van Genuchten, 1980; Mualem, 1976; Vogel and Cislserova, 1988; Vogel et al., 1991; Simunek et al, 1994, 1997; Forsyth et al., 1995]. Moreover, with $n < 2.0$, relative permeabilities can vary greatly for a small change in capillary pressure, and convergence of the nonlinear solver is very sensitive to the method used to estimate the interblock permeabilities [Miller et al., 1998b]. The relevant material properties used for this example were $\mathbf{a} = 0.1 \text{ cm}^{-1}$, $\mathbf{q}_s = 0.45$, $\mathbf{q}_r = 0.05$, $n = 2.5$, $K_s = 72 \text{ cm/day}$. The data set was taken from Huang et al. [1996] who used it for testing their numerical method to solve a 2-dimensional variably saturated flow. They advocated that the above soil type is characterized by highly

nonlinear hydraulic properties. For easy reference, we refer to this model of constant flux infiltration using the VGM relations, as the VGM model.

3.5 PERFORMANCE INDICATORS.

All computations were done on a Pentium III 866 machine with 256 MB of SDRAM and 512KB level 2 cache. For all cases, the relative tolerance and absolute tolerance of 10^{-16} or 10^{-10} or 10^{-6} were used in the MATLAB ODE45 solver routines, or as otherwise stated. Tight error tolerances have been selected so that the spatial error dominates. From this point onwards, we will refer to the relative tolerance and absolute tolerance as *tol*. The MATLAB ODE45 is the standard solver used for the integration of the resultant systems of ordinary differential equations for all models. Valuable statistics about the cost of the integration is obtained by specifying 'stats' in the integration options and are included in the results to analyze the integrator performance. The following statistics about the integrator— successful steps, failed attempts, derivative functions call and CPU times are used.

Most of the models tested have analytical solutions. In cases where analytical solutions are not available, the dense-grid solution is used to represent the analytical solutions (dense-grid means a very fine spatial step). To ensure a unified basis for analysis of accuracy, the following measures of performance were used:

1. The Maximum Relative Error,

$$\text{MRE} = \max \left(\frac{|\mathbf{q}_{Exact} - \mathbf{q}_{Numeric}|}{|\mathbf{q}_{Exact}|} * 100 \right), \quad (3.27)$$

was calculated as the maximum of the relative error between the analytic and the corresponding numerical solution at the required point of the integration. The maximum was taken across the z domain, or x-z domain, for the 1-D, or 2-D, problems. Note that

this measure can also be applied to Ψ based solutions, and this measure gives a precise quantification of the precision of the primary variable in the solution.

2. The Maximum Absolute Error,

$$MAE = \max(|\mathbf{q}_{Exact} - \mathbf{q}_{Numeric}|), \quad (3.28)$$

was calculated as the maximum of the absolute error (for cases where zeros occur in the exact solutions) between the analytic and the corresponding numerical solution at the required point of the integration. Similarly, the maximum was taken across the z domain, or x-z domain, and gives a precise quantification of the precision of the primary variable in the solution. This measure can also be applied to Ψ based solutions.

3. 1-D Global mass balance error,

$$GME(t) = \left(1 - \frac{(M^t - M^0)_{numerical}}{(M^t - M^0)_{analytical}} \right) * 100, \quad (3.29)$$

where M^0 represents the initial mass in the flow domain and M^t represents the mass at time t, [Celia et al., 1990; Rathfelder and Abriola, 1994]. This allows the quantification of the relative error in the gain or loss of the amounts of water infiltrated into the media.

4. 2D Flow Mass errors,

$$FME = \left| \frac{(M^t - M^0) - Q}{Q} \right| 100, \quad (3.30)$$

where the Q is the accumulated net inflow across the boundaries. Equation (3.30) is the sum of the calculated changes in the water volume at each node minus the accumulated net inflow relative to the accumulated net inflow (based on the volumetric flux entering at the surface). This quantifies the overall percentage error in the calculated gain or loss of water infiltrated into the media.

5. CPU is the execution time and exclude the time used reading input files, defining initial conditions, and writing output files.

PDE TEMPLATE

Every year a few research results pay
the freight for all the rest.

Robert A. Frosch

4.1 INTRODUCTION

In the Method of Lines (MOL), a PDE is discretized with respect to all spatial variables while one variable (usually time) is left in continuous form, resulting in a system of Ordinary Differential Equations (ODEs) [Wouwer et al., 2001]. An ODE integrator is then used to integrate the system of ODEs to obtain the solution of the original PDE. The discretization of the spatial variables can be carried out in several ways. In Weideman and Reddy [2000], the concept of a differentiation matrix and the matrix-based approach to scientific computing was described. In that paper, these two concepts have proven to be very useful tools in the numerical solution of differential equations (both partial and ordinary). Canuto et al. [1988] and Fornberg [1996] have also drawn the same conclusion. In this chapter, these two concepts are utilized to create a template that can solve 1-D or 2-D PDE problems.

The concept of a differentiation matrix has been popular, especially in pseudospectral or spectral solution techniques. Using spectral collocation, Weideman and Reddy [2000] designed a software suite for MATLAB 5 consisting of seventeen functions for solving differential equations. It includes functions for computing differentiation matrices of arbitrary order corresponding to Chebyshev, Hermite,

Laguerre, Fourier, and Sinc interpolants. Jameson [1995] has also used wavelet differentiation matrices. Recently Abarbanel and Ditkowski [1997 and 1999], used 2nd and 4th order finite differencing for the differentiation matrix.

In this chapter, finite differencing is also used to create the differentiation matrix, but the order of finite differencing is variable depending on the order of the stencil used. In section 4.2, a simple one dimensional heat equation is used to show the usage of a PDE template to solve a general 1-D PDE problem, and demonstrate the concept of a differentiation matrix and the matrix-based approach to scientific computing. Section 4.3 shows the usage of the template for solving various 1-D PDE problems. Section 4.4 uses the template to solve a 2-D problem within a rectangular domain. Section 4.5 illustrates how to modify the template subroutine to handle a 2-D problem with an irregular shaped domain. The last section gives further extensions that can be made to this PDE template. The attached disc includes the full calling command and description of the MATLAB routines used in the templates.

Moreover, the source code for the subroutines and examples may be obtained gratis from the Internet site, <http://www.mathworks.com/support/ftp/diffeqv5.shtml>, which is the Differential Equations category of the Mathworks user contributed (MATLAB 5) M-file repository. The subroutines and examples require MATLAB version 5.2 or later.

4.2 ONE DIMENSIONAL HEAT EQUATION

Consider the one dimensional heat equation in normalized form, in a rod of unit length under a sine wave initial temperature distribution i.e.,

$$\begin{aligned} T_t = T_{zz} = F, \\ T(z, 0) = \sin(\mathbf{p}z/L). \end{aligned} \tag{4.1}$$

This is a simple PDE model for finding the temperature, $T = T(z, t)$, of a 1-D conducting medium as a function of time t and position z for $0 \leq z \leq 1$. The ends of the rod are held at zero temperature, so the Dirichlet boundary conditions are

$$T(0, t) = T(L, t) = 0. \quad (4.2)$$

The analytical solution satisfying these initial and Dirichlet boundary conditions can be derived as

$$T(z, t) = \exp\left(-\left(\mathbf{p}^2 / L^2\right)t\right) \sin(\mathbf{p} z / L), \quad (4.3)$$

and this analytic solution will be used for validation of the numerical solution schemes.

4.2.1 REPRESENTATION OF DERIVATIVES

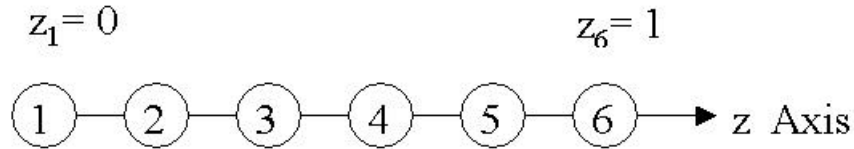


Fig. 4.1 Discretization and Numbering

Consider a function $f(z)$ on a region of the z -axis, which is discretized as shown in Fig. 4.1. Finite difference approximations of the first derivative can be developed about a particular grid point or node in the physical region. The accuracy of the scheme can be improved by increasing the order of the derivative approximations. Wouwer et al. [2001] proposes fourth-order formulae for first-order derivatives; the corresponding derivative formulae at the various nodes can be summarized in terms of a differentiation matrix; or data structure, of the form:

$$\frac{d}{dz} \begin{bmatrix} f_{i-2} \\ f_{i-1} \\ f_i \\ f_{i+1} \\ f_{i+2} \end{bmatrix} = \frac{1}{24\Delta z} * \begin{bmatrix} -50 & 96 & -72 & 32 & -6 \\ -6 & -20 & 36 & -12 & 2 \\ 2 & -16 & 0 & 16 & -2 \\ -2 & 12 & -36 & 20 & 6 \\ 6 & -32 & 72 & -96 & 50 \end{bmatrix} \begin{bmatrix} f_{i-2} \\ f_{i-1} \\ f_i \\ f_{i+1} \\ f_{i+2} \end{bmatrix}, \quad (4.4)$$

$$= \frac{1}{24\Delta z} S \begin{bmatrix} f_{i-2} \\ f_{i-1} \\ f_i \\ f_{i+1} \\ f_{i+2} \end{bmatrix},$$

where $f_i = f(z_i)$ are the values of $f(z_i)$, evaluated at the grid points along the z axis and S represents the matrix, or data structure of coefficients. Note that the diagonal elements of the data structure correspond to the nodes where the first order derivatives are estimated. When applied to a finite region such as shown in Fig 4.1, then the row of S is selected depending on proximity to the ends of the rod. For $i = 3$ or 4 , the third row of S is selected as its elements all lie on the lattice. Near the end points, i.e. at $i = 1$, then the first row of S is selected. To represent the differentiation data structure graphically, stencils may be used to depict the finite difference approximations (Fig. 4.1). A stencil shows the pattern of connection in the difference equation, and the entries in the stencil are the coefficients at the grid points [Lee et al., 1998].

4.2.2 APPLICATION TO THE HEAT EQUATION

Now consider the 1-D Heat equation (4.1), with the grid displayed in Fig. 4.1. The temperature at node i is given by

$$T_i = T(z_i), \quad i = 1, 2, \dots, 6, \quad (4.5)$$

and the vector of these values is $\mathbf{T}^{node} = [T_1 \ T_2 \ \dots \ T_6]^T$.

Then the derivative $\frac{\partial T}{\partial z}$ at the nodes can be estimated using the derivative representations, which are selected from the rows of the data structure in (4.4). For

diffusive terms in the PDE, Wouwer et al. [2001] advocated the use of the central difference scheme for the derivative representations. Schiesser [1996] also recommended the use of the upwinding finite difference schemes for handling the

convective terms. Since (4.1) involved only a diffusive term, $\left. \frac{\partial T}{\partial z} \right|_{z_i}$ for $i=3$, or 4 , can be

estimated using row 3 of S , while values for the end nodes may involve other rows of S , for use with the boundary conditions. With the appropriate choice of stencils,

$\frac{\partial}{\partial z}(T^{node})$ is easily obtained by replacing the partial derivatives with algebraic

approximations (fourth order computational stencils) using the appropriate rows of S .

Thus

$$\mathbf{T}_z^{node} = A_z * \mathbf{T}^{node}, \quad (4.6)$$

where

$$A_z = \frac{1}{24\Delta z} * \begin{bmatrix} -50 & 96 & -72 & 32 & -6 & 0 \\ -6 & -20 & 36 & -12 & 2 & 0 \\ 2 & -16 & 0 & 16 & -2 & 0 \\ 0 & 2 & -16 & 0 & 16 & -2 \\ 0 & -2 & 12 & -36 & 20 & 6 \\ 0 & 6 & -32 & 72 & -96 & 50 \end{bmatrix}, \quad (4.7)$$

\mathbf{T}_z^{node} is the vector of first order derivatives for nodes in the domain, A_z is called the symmetric first order differentiation matrix (FODM), and we denote the upwind FODM as A_{zup} for handling the convective terms. Note that the matrix may not be fully symmetric due to the treatment of the boundary conditions. The symmetry is in the central differencing at fully internal points. Note that the discretized numbering of the nodal points gives the row indices of A_z or A_{zup} , and the indices of the neighbouring nodes gives the column indices of A_z or A_{zup} . Knowing the placement of the derivative stencil in the physical region gives the exact location of the placement of the stencil

coefficients in A_z or A_{zup} via the discretization numbering of the nodes. All this useful information has been used to create an Automatic Differencer or FODM function.

Finally, the right hand side, F , of (4.1) is fully discretized by applying the first-order differential matrix, A_z twice to \mathbf{T}^{node} , a step which is known as stage-wise differentiation, to give

$$\mathbf{T}_{zz}^{node} = A_z * (A_z * \mathbf{T}^{node}) . \quad (4.8)$$

Note that the right hand side (RHS) of $T_t = T_{zz}$ is now fully discretized, to obtain a system of ODEs. Depending on the type of boundary conditions involved, different procedures at different parts of the stage-wise differentiation have to be used to incorporate the boundary conditions directly into the set of ODEs.

For Dirichlet boundary conditions, the values are assigned to the respective boundary points before computation of the derivatives, so that the correct dependent variables \mathbf{T}^{node} are retained. Also zeroing the time derivatives at the boundaries is recommended to restrain the integrator from moving the constant value away from its true value. [Wouwer et al., 2001].

Where Neumann boundary conditions are involved, the boundary conditions are imposed on the boundary points in the first order derivative vector, \mathbf{T}_z^{node} , so that the correct first-order derivatives at the boundaries move into the second-order derivative calculation.

If mixed type conditions are involved, these are treated in the same manner as for the Neumann type problem. This implies that the method is able to handle all types of time-dependent boundary conditions. The initial condition in (4.1) may also be discretized to form the vector set of initial conditions for the system of ODEs. An ODE integrator is then used to integrate the system of ODEs. A point to note is that the application of MOL may result in a stiff system of ordinary differential equations. This means that special ODE solvers designed for solving stiff systems must be used in cases

where conventional methods of integration are inadequate [Wouwer et al., 2001]. MATLAB provides a suite of ODE and also stiff ODE solvers.

Each time the integrator calls for a value of the RHS, a computer routine or m-file is activated to give the values of this RHS vector. From the above, it can be seen that FODM is central to the implementation of the vectorized Schiesser approach to MOL. Notice that Az and $Azup$ are constant for all program calls to the derivative function, and this drastically reduces the computation time by the integration solver.

4.2.3 THE MATLAB PROGRAM TEMPLATE

In this section, the processes for constructing the final system of equations for a general case are incorporated into an algorithm and developed in MATLAB code. ODE solvers only require computer-based routines that provide estimates of the right hand sides of the system of ODEs that they are attempting to solve. The algorithm can be implemented using the following MATLAB template (Table 4.1).

The MATLAB program template is written as a script file, while all other subroutines are function files. The main program is divided into six sections as indicated in Table 4.1. These sections are the initialization of parameters, discretization and numbering of the domain, assignment of the initial condition, creation of the first order differentiation matrices, integration of the ODEs using an integrator and the calculation of the error where exact solutions are available. The following is a brief description of the six sections in the PDE template. For a detailed description of the full calling command and description of the functions used in the template, please refer to the attached disc or Appendix A.

Table 4.1 The PDE Template for solving the heat equation.

<code>global Az Azup b1 b2 k</code>	<code>% Initialize parameters</code>
<code>A=0; B=1; dzn = 6; tstart =0; tfinal=1;</code>	
<code>[G,k,z,dz,m,mm,b1,b2] = domain(A,B,dzn);</code>	<code>% Domain- discretize and number</code>
<code>T0 = sin(pi*z);</code>	<code>% Initial conditions</code>
<code>[Az,Azup] = fodm(G,dz,z,m,b1,b2);</code>	<code>% Az & Azup is created</code>
<code>tspan = [tstart:tfinal/4:tfinal];</code>	<code>% Solver to integrate the ODEs</code>
<code>options = odeset('RelTol',1e-6, 'AbsTol',1e-6);</code>	
<code>[t, T] = ode45('rhs1', tspan, T0, options);</code>	
<code>Exact = exp(-pi*pi*t(5)).*sin(pi*z);</code>	<code>% Absolute Error at T(5)& output</code>
<code>err = (Exact - T(5,:));</code>	
<code>Relerr = abs(err./Exact);</code>	

The initialization section initializes the physical properties of the physical region. For a 1-D problem, only three physical variables need to be defined: (1) The number of discretized points, (2) the boundary points in the physical grid system, and (3) the period of the simulation.

A global declaration is used to initialize all global variables used in the program; this has the effect of shortening the number of inputs used in calling the various subroutines.

The domain function creates and numbers the discretization region using the three physical variables defined in the initialization section. The physical domain is created and manifested as matrices and vectors with numerical values to be used in Az or Azup. The shape of the domain can be irregular. Note that the computational stencil used is a string of nodes, which does not “bend” to suit the shape of the domain. The numbering of the discretization points may be stored in an array. For the 1-D case in (3.5), these arrays are vectors such as

$$G = [1 \ 2 \ 3 \ 4 \ 5 \ 6] \text{ and } k = [1 \ 2 \ 3 \ 4 \ 5 \ 6],$$

where G is a matrix containing all the discretized points with values equal to the numbering or node index scheme and zeros for points located externally to the discretized domain. There is a direct correspondence between the matrix location and the actual physical domain. The variable k contains the indices of the non-zero elements in G (when G is reshaped into a column vector by stacking each column of the original G matrix on top of each other sequentially). For a one-dimensional problem, there is no difference between k and G . This column based numbering is used since data manipulation in MATLAB storage works in a column-wise manner. These indices will be elaborated further in the two dimensional case.

The FODM function accepts four inputs and yields the first order differential matrices. There are five types of computational stencils in a fourth order representation of a first order derivative and the central nodes in the domain that can fix several forms of stencils as mentioned previously. The discretization points are divided into 6 batches or categories i.e. $pt1$, $pt2$, $pt3$, $pt4$, $pt5$ and $ptBz$. These batches of points are in turn used to create a row of the first order differential matrix (using function $sten1$), and their combination makes up the first order differential matrix. Note that the points are the location of the nodes where the derivatives are approximated. For example, in (4.1) subjected to its boundary conditions, $b1=1$ and $b2=6$, and the differential matrix is given by Az , as shown in (4.7).

Additional important outputs from the domain function are $b1$ and $b2$. While $b1$ is the node index of the grid point at the starting point of the discretized region, $b2$ is the node index at the far end point of the discretized region. These two points are important inputs for the FODM function, which uses them to create the different batches of points assigned to the respective derivative stencils, and which subsequently generate the correct start and finish rows in the first order differential matrices.

The initial condition section initializes the initial condition of the physical domain and it needs to be in vector form. Note that the use of the matrix based approach in MATLAB enables mathematical formulae to be expressed as they are in the coding.

The MATLAB ODE solver suite, developed by Shampine and Reichelt, [1994] comprises 3 non-stiff system solvers (ODE45, ODE23, ODE113) and four stiff system solvers (ODE15s, ODE23s, ODE23t, and ODE23tb). While ODE45 is based on an explicit Runge-Kutta (4,5), the Dormand-Prince pair, ODE23 is based on an explicit Runge-Kutta (2,3) pair of Bogacki and Shapmine [Shampine and Reichelt, 1994]. The third explicit solver, ODE13, is a multistep solver- a variable order Adam-Bashforth-Mouth PECE solver. Note that solving an implicit ODE approximation formula by a prediction step and a single correction step is called PECE, where E denotes evaluations of the derivative function (thus, Prediction, Evaluation, Correction, Evaluation). On the other hand, the stiff solvers, i.e. ODE15s, ODE23s, ODE23t, and ODE23tb, are based on the numerical differentiation formulas (NDFs), a modified Rosenbrock formula of order 2, the trapezoidal rule (using a “free” interpolant), and an implicit Runge-Kutta (using a trapezoidal rule and a backward differentiation formula of order 2) respectively. All these variable-step solvers are designed to support sparse systems and are capable of handling mass matrices i.e. the more general form:

$$M(t) (\mathbf{h}^{nodes})_t = f(t, \mathbf{h}^{nodes}), \quad (4.9)$$

where $M(t)$ is a mass matrix that is non-singular and (usually) sparse. If M is singular, then (4.9) is a differential algebraic equation (DAE). For a detailed description of these stiff or nonstiff solvers, the readers are referred to the MATLAB’s user guide. The stiff solvers in the MATLAB’s ODE suite were not used because it has been shown that they are not suitable for our type of problems [Lee, 1996]. For all numerical experiments, the ODE45 integrator was used.

Table 4.2 The right-hand side function for the heat equation.

```
function dT = rhs1(t,T)
global Az Azup b1 b2

T(b1) = 0*ones(size(b1));           % Enforce Dirichlet boundary condition
T(b2) = 0*ones(size(b2));

Tz  = Az * T;                       % Stage-wise differentiation
Tzz = Az * Tz;

dT = Tzz;
dT(b1) = zeros(size(b1));           % Restrain integrator from moving bc.
dT(b2) = zeros(size(b2));
```

The routine `rhs1` shown in Table 4.2 uses `Az` and applies the specified boundary conditions to yield the appropriate RHS of the system of ODEs. Note that this routine is easily modified to use `Azup`. With the solutions t and T obtained from the ODE integrator, it is relatively easy to display the results visually using the MATLAB graphical facility. Table 4.3 shows the relative error for the heat equation at time, $t=1$. Even for a crude spatial step of 0.2cm (consisting of 6 discretized points) and a tolerance of 10^{-6} , an accuracy of 2 significant figures is obtained.

Table 4.3 Results for the heat equation at $t=1$.

z	Relative Error * 10^{-2}
0.2	5.1
0.4	4.8
0.6	4.8
0.8	5.1

4.3 TEMPLATE FOR SOLVING VARIOUS 1-D PDE

This section shows how the PDE templates in Table 4.1 and Table 4.2 are modified to solve various 1-D PDE problems. The codes given are readily adapted to the mathematical description of each problem. Simple logic will be sufficient to understand the relationship between the examples and the MATLAB code; little knowledge of technical MATLAB is needed.

The sample problems are:

Example 1. The scalar hyperbolic transport equation.

$$\begin{aligned}
 T_t &= -T_z, \quad z \in [-1,1], \quad t > 0, \\
 \text{ICs: } T(z,0) &= -\sin(\mathbf{p}^*z), \\
 \text{BCs: } \begin{cases} T(-1,t) = -\sin(\mathbf{p}^*(-1-t)), \\ T(1,t) = -\sin(\mathbf{p}^*(1-t)), \end{cases} & \quad (4.10) \\
 \text{Exact: } T(z,t) &= -\sin(\mathbf{p}^*(z-t)).
 \end{aligned}$$

This problem represents the convection of an initial profile and the exact solution is given above. The template in Table 4.4 initializes the parameter values and constructs the differentiation matrices.

Table 4.4 The PDE Template for solving the hyperbolic equation.

```

global Az Azup b1 b2 k % Initialize parameters
A=-1; B=1; dzn = 21; tstart =0; tfinal=1;

[G,k,z,dz,m,mm,b1,b2] = domain(A, B, dzn); % Domain – discretize and number

T0 = -sin(pi*z); % Initial conditions

[Az,Azup] = fodm(G,dz,z,m,b1,b2); % Az & Azup is created

tspan = [tstart:tfinal/4:tfinal]; % Solver to integrate the ODEs
options = odeset('RelTol',1e-6, 'AbsTol',1e-6);
[t, T] = ode45('rhs2', tspan, T0, options);

Exact = -sin(pi*(z-t(5)));; % Error at T(5)&output
err = (Exact - T(5,:));
Relerr = abs(err./Exact);

```

The template in Table 4.5 generates the right hand side of the system of equations to be solved by the ODE solver.

Table 4.5 The right-hand side function for the hyperbolic equation.

```

function dT = rhs2(t,T)
global Az Azup b1 b2

T(b1) = -sin(pi*(-1-t)).*ones(size(b1)); % Enforce Dirichlet b.c.
T(b2) = -sin(pi*(1-t)).*ones(size(b2));

Tz = Azup * T; % Stage-wise differentiation
dT = -Tz;
dT(b1) = zeros(size(b1)); % Restrain integrator from moving bc.
dT(b2) = zeros(size(b2));

```

The relative errors in the solution for this scalar hyperbolic transport equation at $t=1$, are shown in Table 4.6. For 21 discretization points, an accuracy of 3 significant figures is obtained.

Table 4.6 Results for the hyperbolic equation at $t=1$.

z	Relative Error * 10^{-4}
-0.5	3.4822
0	13.262
0.5	4.4286

Example 2. The 1-D Burgers' equation.

This problem consists of the well-known Burgers' equation [Schiesser, 1991] which represents the interaction of convection and diffusion effects. The complete definition of the equation and model are described in section 3.2. The template in Table 4.7 initializes the parameter values and constructs the differentiation matrices.

Table 4.7 The PDE Template solving the 1-D Burgers' equation.

<code>global Az Azup b1 b2 k vis z</code>	<code>% Initialize parameters</code>
<code>A=0; B=1; dzn = 21; tstart=0; tfinal=0.5; vis=0.1;</code>	
<code>[G,k,z,dz,m,mm,b1,b2] = domain(A, B, dzn);</code>	<code>% Domain– discretize & number</code>
<code>T0= 1./(1+exp(z./(2*vis)- (tstart)/(4*vis)));</code>	<code>% Initial conditions</code>
<code>[Az,Azup] = fodm(G,dz,z,m,b1,b2);</code>	<code>% Az & Azup is created</code>
<code>tspan = [tstart:tfinal/4:tfinal];</code>	<code>% Solver to integrate the equation</code>
<code>options = odeset('RelTol',1e-10,'AbsTol',1e-10);</code>	
<code>[t, T] = ode45('rhs3', tspan, T0, options);</code>	
<code>Exact = 1./(1+exp(z./(2*vis)- t(5)/(4*vis)));</code>	<code>% Abserror at 0.5 & output</code>
<code>err = (Exact - T(5,:));</code>	
<code>Relerr = abs(err./Exact);</code>	

The template in Table 4.8 generates the right hand side of the system of equations to be solved by the ODE solver.

Table 4.8 The right-hand side function for the 1-D Burgers' equation.

```
function dT = rhs3(t,T)
global Az Azup b1 b2 vis z

Tz = (Az * T);
Tz(b1) = hderiv(z(b1),vis,t);           % Enforce Neumann b.c.
Tz(b2) = hderiv(z(b2),vis,t);
Tzz = (Az * Tz);                         % Stage-wise differentiation

Tzup = (Azup * T);
Tzup(b1) = hderiv(z(b1),vis,t);         % Enforce Neumann b.c.
Tzup(b2) = hderiv(z(b2),vis,t);

dT = -T.*Tzup + vis*Tzz ;

function hh = hderiv(z,vis,t)            %function of a function
Ex= exp( z./(2*vis)- t./(4*vis) );
hh= -1*(1+Ex).^(-2).*Ex.*(1/(2*vis));
```

The relative errors in the solution for this Burgers' equation at $t=1$ and $n=0.1$ is shown in Table 4.9. For 21 discretization points, an accuracy of 5 significant figures is obtained, which represents an excellent result.

Table 4.9 Results for the 1-D Burgers' equation.

z	Relative Error * 10 ⁻⁵
0	3.7808
0.2	1.8195
0.4	7.7551
0.6	1.4049
0.8	12.116
1	28.071

Example 3. 2 Coupled Nonlinear PDE's

$$\begin{cases} T_t = ((H - 1)T_z)_z + (16zt - 2t - 16(H - 1))(H - 1) + 10ze^{-4z}, \\ H_t = H_{zz} + T_z + 4T - 4 + z^2 - 2t - 10te^{-4z}, \\ 0 \leq z \leq 1, \text{ and } 0 \leq t \leq 4, \end{cases}$$

ICs: $T(z, 0) = H(z, 0) = 1,$ (4.10)

BCs: $\begin{cases} T(0, t) = H(0, t) = 1, \\ T_z(1, t) = 3 - 3T(1, t), \quad H_z(1, t) = e^4(T(1, t) - 1), \end{cases}$

Exact: $\begin{cases} T(z, t) = 1 + 10zte^{-4z}, \\ H(z, t) = 1 + z^2t, \end{cases}$

This problem consists of two nonlinear-coupled PDE [Madsen and Sincovec, 1976], which can be considered as one parabolic and the other hyperbolic. The template in Table 4.10 initializes the parameter values and constructs the differentiation matrices. Note that in Table 4.10 and Table 4.11, the coupled PDE are solved by combining the two independent variables into a single variable, i.e. TH0. Moreover, note that the type of boundary conditions for the coupled PDE are different; different procedures at different parts of the stage-wise differentiation have to be used to incorporate the boundary conditions directly into the two different set of ODEs, i.e. dT and dH.

Table 4.10 **The PDE Template solving the 2 Coupled nonlinear PDE's.**

<code>global Az Azup b1 b2 k z M</code>	<code>% Initialize parameters</code>
<code>A=0; B=1; dzn = 21; tstart =0; tfinal=2;</code>	
<code>[G,k,z,dz,m,mm,b1,b2] = domain(A, B, dzn);</code>	<code>% Domain - discretize & number</code>
<code>M = size(z,1);</code>	
<code>T0= 1*ones(M,1);</code>	<code>% Initial conditions</code>
<code>H0= T0; TH0 = [T0;H0];</code>	
<code>[Az,Azup] = fodm(G,dz,z,m,b1,b2);</code>	<code>% Az & Azup is created</code>
<code>tspan = [tstart:tfinal/4:tfinal];</code>	<code>% Solver to integrate the ODEs</code>
<code>options = odeset('RelTol',1e-6,'AbsTol',1e-6);</code>	
<code>[t, TH] = ode45('rhs4', tspan, TH0, options);</code>	
<code>Exact = [1+10*z*t(5).*exp(-4*z);1+t(5)*z.^2];</code>	<code>% Results at t=2</code>
<code>err = abs(Exact-TH(5,:));</code>	
<code>Relerr = abs(err./Exact);</code>	
<code>Relerr = [z Relerr(1:M,1) Relerr(M+1:2*M,1)];</code>	

The template in Table 4.11 generates the right hand side of the system of equations to be solved by the ODE solver.

Table 4.11 The right-hand side function for the 2 Coupled nonlinear PDE's.

```

function dTH = rhs4(t,TH)
global Az Azup b1 b2 z M
T = TH(1:M,1);
H = TH(M+1:2*M,1);

T(b1) = 1*ones(size(b1));           % Enforce Dirichlet b.c.
Tz  = Az * T;                       % Stage-wise differentiation
Tz(b2) = 3-3*T(b2);                 % Enforce Neumann b.c.
Tzz  = Az * Tz;                     % Stage-wise differentiation

Tzup(b1) = 1*ones(size(b1));        % Enforce Dirichlet b.c.
Tzup  = Azup * T;                   % Stage-wise differentiation
Tzup(b2) = 3-3*T(b2);               % Enforce Neumann b.c.

H(b1) = 1*ones(size(b1));           % Enforce Dirichlet b.c.
Hz  = Az * H;                       % Stage-wise differentiation
Hz(b2) = ((exp(1)).^4)*(T(b2)-1)/5; % Enforce Neumann b.c.
Hzz  = Az * Hz;                     % Stage-wise differentiation

dT = Az*((H-1).*Tz)+(16*z*t-2*t-16*(H-1)).*(T-1) + 10*z.*exp(-4*z);
dT(b1) = zeros(size(b1));           % Restrain integrator from moving bc.

dH = Hzz + Tzup + 4*T - 4 + z.^2 -2*t - 10*t*exp(-4*z);
dH(b1) = zeros(size(b1));           % Restrain integrator from moving bc.

dTH = [dT ; dH];

```

The relative errors in the solution for (4.10) at $t=2$ is shown in Table 4.12- good accuracy of 4 significance figures is obtained even for a relatively large tolerance (10^{-6}) used in the integrator and 21 discretize points.

Table 4.12 Results for the 2 Coupled nonlinear PDE.

z	Relative Error for T * 10^{-5}	Relative Error for H * 10^{-5}
0	0	0
0.2	68.152	14.672
0.4	16.929	16.051
0.6	9.3288	12.697
0.8	13.489	9.3772
1	1.1712	2.7322

4.4 TEMPLATE FOR SOLVING 2-D PROBLEM ON A RECTANGULAR DOMAIN

In this section, the 1-D PDE template is modified to handle a 2-D problem within a rectangular domain. Again, we will use an example to illustrate this modification by extending Burgers' Equation (see (3.1)) to two dimensions. In this way, the reader can readily see the mechanics involved in the transformation of the coding in the 1-D PDE template to a 2-D PDE template. The main operation (with the addition of another dimension) is that components that were used in the one-dimensional case are duplicated for the second dimension. This can be viewed as rearranging the discretized nodes in the rectangular domain into a column vector where a one-dimensional operation used in the 1-D PDE template can be performed. The strategy of treating each dimension separately and combining them is also used. Once the 2-D PDE template is created, it can easily be utilized for any general 2-D PDE problem within a rectangular or a square domain. The following is the 2-D PDE template with a brief description of the changes made. For a detailed description of the full calling command and description of the functions used in the template, please refer to Appendix B.

Example 4. The 2-D Burgers' equation.

$$\begin{aligned}
 \Omega_t &= -\Omega \Omega_z + \mathbf{n} \Omega_{zz} - \Omega \Omega_x + \mathbf{n} \Omega_{xx}, \quad z, x \in [0, 1], \quad t > 0, \quad \mathbf{n} = 0.1, \\
 \text{ICs: } \Omega(x, z, 0) &= 1 / \{1 + \exp(0.5z/\mathbf{n} + 0.5x/\mathbf{n})\}, \\
 \text{BCs: } \begin{cases} \Omega_z(x, z, t) = \Omega_x(x, z, t) = -(1/2\mathbf{n})(1 + E)^{-2}, \\ \text{where } E = \exp(0.5z/\mathbf{n} + 0.5x/\mathbf{n} - 0.5t/\mathbf{n}), \end{cases} & \quad (4.11) \\
 \text{Exact: } \Omega(x, z, t) &= 1 / \{1 + \exp(0.5z/\mathbf{n} + 0.5x/\mathbf{n} - 0.5t/\mathbf{n})\}.
 \end{aligned}$$

The template in Table 4.13 initializes the parameter values and constructs the differentiation matrices. In the initialization section, the physical variables now contain the length (A1 and B1) and breath (A2 and B2) of the domain. This information is used in the domain function to discretize, number and transform the domain into vectors and matrices, which can be used in the numerical scheme. Note that two matrices are

created to reflect the x and z Cartesian co-ordinates of the physical system via the domain function. Making use of logical operators and these matrices, the discretized region is generated and transformed to the column wise numbering in the G matrix.

Table 4.13 The PDE Template solving the 2-D Burgers' equation.

```

global b1 b2 b3 b4
global Ax Az Axup Azup k z x vis

A1=0; B1=1; A2=0; B2=1; dxn=11; dzn=11; vis=1;           %Section 1
tstart =0; tfinal=1;

[G,k,x,z,dx,dz,m,mm,b1,b2,b3,b4] = domain(dxn,dzn,A1,B1,A2,B2); %Section 2

T0= 1./(1+exp(x./(2*vis) + z./(2*vis)- (2*tstart)/(4*vis))); %Section 3
T0= T0(:); % stack into a % column vector

[Ax,Axup,Az,Azup] = fodm(G,dx,dz,x,z,m,mm,b1,b2,b3,b4); %Section 4

tspan = [tstart:tfinal/4:tfinal]; %Section 5
options = odeset('RelTol',1e-6, 'AbsTol',1e-6);
[t, T] = ode45('rhs5', tspan, T0, options);

Exact = 1./(1+exp(x./(2*vis)+z./(2*vis)-(2*tfinal)/(4*vis))); %Section 6
Exact = Exact(:);
Abserr = Exact - T(5,:);
Relerr = abs(err./Exact);
Relerr = reshape(Relerr,size(G))

```

The creation of the different forms of first order differential matrices (Az, Azup, Ax, Axup) is just a duplication of the method used for the MATLAB variable Az in the 1-D example, clearly depicted in the FODM coding. The MATLAB variables Ax and Axup are the differentiation matrices for the functional values in the x-axis, and Az and Azup are the MATLAB variables for the differentiation matrices for the z-axis.

The template in Table 4.14 generates the right hand side of the system of equations to be solved by the ODE solver. In the rhs routine (rhs5 for this example), the right hand side of (4.11) is discretized using an appropriate discretization matrix, leading to different forms of first order differentiation matrix for each term. Thus the spatial derivatives in the coding can be readily expressed in forms similar to $(\mathbf{I}^{\text{nodes}})_z$

in (4.6), but with appropriate differentiation matrices designed for two space dimensions.

Table 4.14 The right-hand side function for the 2-D Burgers' equation.

```
function dT=rhs5(t,T)
global Ax Az Axup Azup b1 b2 b3 b4 x z vis

Tx  = Ax * T;                               % Stage-wise differentiation
Tx(b3) = hderiv(x(b3),z(b3),vis,t);         % Enforce Neumann condition
Tx(b4) = hderiv(x(b4),z(b4),vis,t);
Txx  = Ax * Tx;
Txup = Azup * T;                             % Stage-wise differentiation
Txup(b3) = hderiv(x(b3),z(b3),vis,t);      % Enforce Neumann condition
Txup(b4) = hderiv(x(b4),z(b4),vis,t);

Tz  = Az * T;                               % Stage-wise differentiation
Tz(b1) = hderiv(x(b1),z(b1),vis,t);        % Enforce Neumann condition
Tz(b2) = hderiv(x(b2),z(b2),vis,t);
Tzz  = Az * Tz;
Tzup = Azup * T;                             % Stage-wise differentiation
Tzup(b1) = hderiv(x(b1),z(b1),vis,t);      % Enforce Neumann condition
Tzup(b2) = hderiv(x(b2),z(b2),vis,t);

dT = -T.*Txup + vis*Txx - T.*Tzup + vis*Tzz;

function hh = hderiv(x,z,vis,t)             % function of a function
Ex = exp(x./(2*vis) + z./(2*vis) - t/(2*vis));
hh = -1*(1+Ex).^(-2).*Ex.*(1/(2*vis));
```

To control the numerical diffusion, upwind finite differences are used to estimate the spatial derivatives in the z-axis and x-axis on the lattice. The dominant flow is in the direction of the positive z and x-axis, hence the appropriate Axup and Azup can be used to assist in the calculation of the derivatives. Note that if the flow is in the reverse direction, it can be easily handled by replacing the finite difference stencils with one that is upwinding in the reverse direction. Also note the role of the Neumann boundary conditions in the coding and the strategy of dealing with each term in the RHS individually and then combining them together. The relative errors shown in Table 4.15; for the 2-D Burgers' equation at t=1, and $n=1$, shows that good accuracy of 7 significant figures is obtained even for the relatively small tolerance (10^{-6}) used in the integrator and $11*11 = 121$ spatial discretized points.

Table 4.15 Results for the 2-D Burgers' equation at t=1.

(x, z)	Relative Error
(0,0)	$5.6 * 10^{-8}$
(0,0.6)	$5.0 * 10^{-10}$
(0,1)	$1.9 * 10^{-7}$
(0.6,0)	$5.0 * 10^{-10}$
(0.6,0.6)	$1.1 * 10^{-8}$
(0.6,1)	$2.0 * 10^{-8}$
(1,0)	$1.9 * 10^{-7}$
(1,0.6)	$2.0 * 10^{-8}$
(1,1)	$5.9 * 10^{-7}$

The above 2-D PDE template has shown that the principle used in the 1-D PDE template can easily be extended for a 2-D problem. The essential stages are the same as in the 1-D example, and the coding is readily understood on a walk through step by step basis. This 2-D PDE template can be used to solve any similar 2-D PDE on a rectangular or square domain, in a similar manner to those used in section 3.

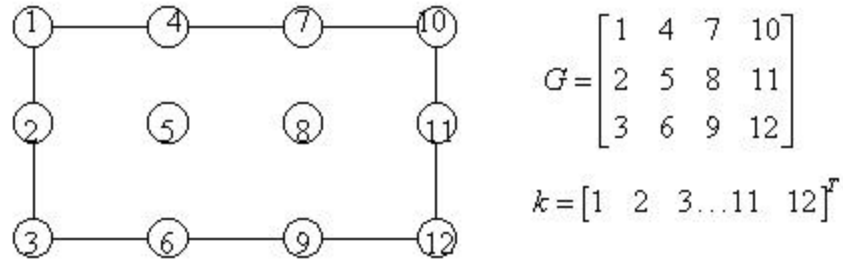
4.5 MODIFICATION OF TEMPLATE FOR 2-D PROBLEM WITH IRREGULAR SHAPED DOMAIN

Finite difference methods are not normally able to handle irregularly shaped domains. Finite elements can handle a wide variety of irregularly shaped domains, and can also be incorporated into a MOL type solution.

Despite the fact that finite difference are not normally able to handle irregularly shaped domains, the template approach described in this thesis can be applied to irregularly shaped regions.

For the irregularly shaped domain, the variables G, k and the boundary points need to be modified in order to create the first order differentiation matrices. To illustrate the major changes in these variables, we will first depict the G, k and the boundary points for a rectangular domain and then those for an irregularly shaped domain. In this way, the modification of the G, k and the boundary points needed in the implementation of the template can be fully appreciated.

Consider the discretized and numbered rectangular domain and the G matrix shown in Fig. 4.2. It can be seen that the G matrix is a rectangular matrix, whose values are the discretization indices. Furthermore, the outline of the non-zero values in matrix G gives the shape of the physical domain, i.e. a rectangle. From the G matrix, the boundary points can easily be derived, and are given by b1, b2, b3 and b4.



$$b1=[1 \ 4 \ 7 \ 10], b2=[3 \ 6 \ 9 \ 12], b3=[1 \ 2 \ 3] \text{ and } b4=[10 \ 11 \ 12].$$

Fig. 4.2 The discretized domain, G matrix, k vector and the boundary points for a rectangle domain.

The difference between G and k is clearly seen when an irregular shape is considered. For example, we consider the problem domain shown in Fig. 4.3, where the region is irregular. The problem domain is located within a further rectangular domain which is latticed and the nodes numbered as shown.

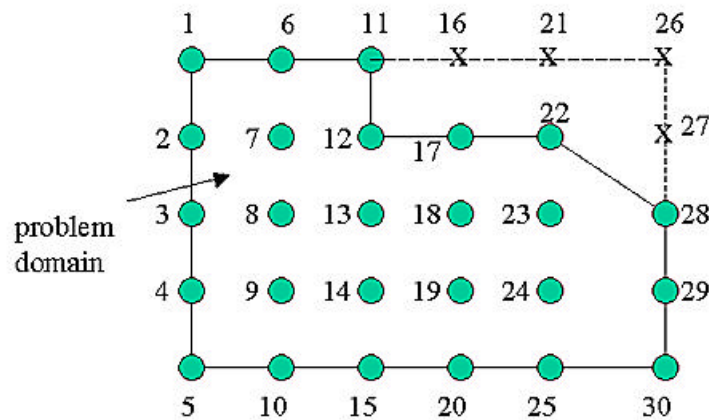


Fig. 4.3 The discretized domain of an irregularly shaped domain.

The vector k contains the node indices, in order, of the nodes in the problem domain. Thus

$$k = [1 \dots 15 \ 17 \ 18 \ 19 \ 20 \ 22 \ 23 \ 24 \ 25 \ 28 \ 29 \ 30].$$

The quantity of G is

$$G = \begin{bmatrix} 1 & 6 & 11 & 0 & 0 & 0 \\ 2 & 7 & 12 & 16 & 20 & 0 \\ 3 & 8 & 13 & 17 & 21 & 24 \\ 4 & 9 & 14 & 18 & 22 & 25 \\ 5 & 10 & 15 & 19 & 23 & 26 \end{bmatrix},$$

where the numbering is in order for internal points in the problem domain.

The outline of the non-zero values in matrix G gives the shape of the physical domain. Zeros are assigned to points, which are outside the discretization of the problem domain. The vector k contains the indices of the non-zero elements in the concatenated G , which is created by stacking the columns in G on top of each other from left to right. Note that nodes 16, 21, 26 and 27 are not in vector k , since they are zero elements in G . Another point to note is that with a two-dimensional shape, the Cartesian coordinates of the domain are recorded in two matrices instead of one, whose size equals that of G .

With the variables G and k , the boundary points in the domain can be computed and subsequently the FODM can be created. Again the concept of viewing the problem as a 1-dimensional problem in each dimension is used. For example, to obtain the bottom boundary points in the irregular-shaped domain, we can implement:

```

b2 = max(G);

Index = find(b2~=0);
if ~isempty(Index)      % not empty
    b2 = b2(Index);
end

```

The above codes apply the maximum function on the G matrix. This yields all the discretized points in the bottom boundary points. Then it removes the points with zero values in the b2 vector. The resultant b2 vector yields the locations of the bottom boundary points in the irregular-shaped domain. Similarly, the other three modified variables, i.e. b1, b3, and b4 can be obtained in this manner. The FODM function needs these modified variables, i.e. G, k, b1, b2, b3, and b4 to create the differentiation matrices. Hence, the methodology used for a 2-D PDE with an irregular shaped domain is similar to that used in solving a 2-D PDE problem with a square domain. The only extra work is in the computation of the variables G, and k, the boundary points in the domain function, and some modifications of the FODM to yield the differentiation matrix. For a detailed description of modifications of the FODM, the readers are referred to the FODM function in the attached disc (Appendix C).

We shall not solve a problem on the above shape, but will illustrate the implementation of the 2-D PDE template with an example that has a curved boundary. The example consists of heat conduction in a semi-circle plate, which can be described via the Laplacian equation [Gerald and Wheatley, 1989]. The constant diffusivity Heat Model is

$$\begin{aligned}
 T_t = T_{xx} + T_{zz} &= 0, \quad x^2 + z^2 \leq 1, \quad t > 0, \quad x \in [-1,1], \quad z \in [0,1], \\
 \text{ICs: } T(x, z, 0) &= 30, \\
 \text{BCs: } \begin{cases} T(x, 0, t) = 0, \\ x^2 + z^2 = 1, \quad T(x, z, t) = 100. \end{cases}
 \end{aligned} \tag{4.12}$$

The example serves two purposes. Firstly, it shows that a curved domain can be implemented using the PDE template (with some modification). Secondly, the PDE templates can be used to solve a steady state problem. This is achieved by creating an initial-value independent variable, $T = T(x, z, t)$, and allowing the solution to proceed to infinite t to obtain the steady state solution which does satisfy Laplace's equation. An

arbitrary initial condition is also given to start the integration and it does not affect the final steady state solution so obtained. This approach of solving the Laplacian in an initial value problem is called the method of false transients. The problem is solved numerically in Cartesian coordinates and its numerical solution is compared with the analytical solution in Cartesian coordinates given in Gerald and Wheatley [1989]. Note that the computation of the relative error is not incorporated into the PDE template, but rather was done manually in respect to the analytical Cartesian coordinates solution given in Gerald and Wheatley [1989]. The template and Right Hand Side for the ODE solver are given in Tables 4.16 and 4.17. Table 4.18 shows the relative error of the solution at $t=500$, where the solution has reached its steady state profile.

Table 4.16 PDE Template solving the 2-D Laplacian on a irregular shaped domain with spatial step of $dx=0.1143$ and $dz=0.0571$, and $21*21$ points.

```

global b1 b2 b3 b4
global Ax Az Axup Azup k z x k

A1=-1.2; B1=1.2; A2=0; B2=1.2; dxn=21; dzn=21; %Section 1
tstart =0; tfinal=500;

[G,k,x,z,dx,dz,m,mm,b1,b2,b3,b4] = domain(dxn,dzn,A1,B1,A2,B2); %Section 2

T0= 0*ones(size(k)); %Section 3

[Ax,Axup,Az,Azup] = fodm(G,dx,dz,x,z,m,mm,b1,b2,b3,b4); %Section 4

tspan = [tstart:tfinal/4:tfinal]; %Section 5
options = odeset('RelTol',1e-10,'AbsTol',1e-10);
[t, T] = ode45('rhs', tspan, T0, options);

y1s = mapped(G,T(5,:)); %Section 6
contour(x,z,(y1s));

```

Table 4.17 The right-hand side function for the Laplacian equation with irregular shaped domain.

```
function dT=rhs6(t,T)
global Ax Az Axup Azup b1 b2 b3 b4 x z

T(b3) = 100*ones(size(b3));           % Enforce Dirichlet condition
T(b4) = 100*ones(size(b4));
Tx   = Ax * T;                         % Stage-wise differentiation
Txx  = Ax * Tx;

T(b1) = zeros(size(b1));               % Enforce Dirichlet condition
T(b2) = 100*ones(size(b2));
Tz   = Az * T;                         % Stage-wise differentiation
Tzz  = Az * Tz;

dT = Txx + Tzz;
dT(b3) =0; dT(b4) =0;                 % Restrain Integrator from moving b.c.
dT(b1) =0; dT(b2) =0;
```

Table 4.18 Relative error in the domain at t=500 for an irregular shaped domain.

(x, z)	Relative Error
(0, 0.8)	$1.3489 * 10^{-2}$
(0.2,0.8)	$1.1775 * 10^{-2}$
(0.4,0.8)	$8.0678 * 10^{-3}$
(0, 0.6)	$8.0588 * 10^{-3}$
(0.2,0.6)	$3.3455 * 10^{-3}$
(0.4,0.6)	$4.6059 * 10^{-3}$
(0.6,0.6)	$1.1559 * 10^{-2}$
(0, 0.4)	$4.0167 * 10^{-3}$

The above 2-D PDE template has clearly shown that the principle used in the 1-D PDE template can easily be extended for a 2-D problem with an irregular shaped domain. The essential stages are the same as in the 1-D or 2-D example, and the coding is readily understood on a walk through step by step basis. Note that the results in Table 4.18 are of low accuracy. This could be due to the fact that the contour of the irregular shaped domain has not been captured precisely using the logical operator in MATLAB and the coordinate matrices. This is one of the disadvantages of using the PDE template for an irregular-shaped domain. Another disadvantage is that the storage, allocation and selection of the boundary points also need some modifications to successfully implement the PDE templates.

4.6 EXTENSIONS

We have described a PDE template that uses the concept of a differentiation matrix, and a matrix based approach to solving classes of Partial Differential Equations. The above has clearly shown that the use of a PDE template based on these two concepts is simple and efficient. It also illustrates that the PDE templates are relatively simple, short, and versatile. With a few changes in the coding (in a plug and play manner), one can easily solve various forms of PDE problem within minutes. Clearly, the size of the problems and degree of difficulties the PDE may impose on the numerical computation are additional factors that will account for the overall modelling times.

The subroutines can be treated as built-in functions with input and output variables. The attraction is that the modular nature of these subroutines allows the user to implement them without much technical knowledge of each algorithm. Moreover, with precise information of the algorithm, one can modify and extend their capability for more complex problems. With the understanding that writing and verifying computer code is very time consuming, we believe that this PDE template and modular structure is very attractive.

There are several ways to improve the efficiency, reliability and robustness of this PDE template. Since the template is modular, each module can be replaced or modified to suit individual needs. For example the creation of the differentiation matrix can be replaced with one that uses spectral methods. For an irregular shaped domain, the FODM modules need to be modified to handle and allocate different nodal points, this had been done in Matthews [2002], and in this chapter. In fact, the PDE templates can be developed into a modelling software package for solving one, two or three-dimensional problems, especially via the full utilization of the MATLAB graphical interfaces.

Forms of Richards' Equation

The way to do research is to attack the facts at the point of greatest astonishment.

Celia Green

5.1 INTRODUCTION

It is well known that numerical simulation of water infiltration into dry soils is a very difficult numerical problem to solve [Forsyth et al., 1995]. Moreover, the form of Richards' equation in the numerical model is a crucial factor in the development of the numerical approximation. The most common way to model water flow through soil is to use the Ψ -based or pressure form of Richards Equation [Richards, 1931]. This is due to its applicability for both saturated and unsaturated soils, as well as for layered soils. However, numerical solutions for this model suffer from two inadequacies. First, they exhibit poor mass balances for unsaturated soils, and second the numerical solution techniques do not handle very steep wetting fronts (in terms of Ψ) for infiltration into soil. [Celia et al. 1990, Babajimopoulos, 2000]. On the other hand, the q -based or water content formulation of Richards Equation was found to be superior at conserving water mass and is insensitive to dry initial conditions [Dullien, 1979]. Unfortunately, the q -base algorithms are not suitable for near saturation because of large changes in the soil water diffusivity near saturation [Hills et al, 1989].

To consolidate the advantages of the above two forms of RE, many researchers have advocated the use of the mixed form of RE. The mixed form consists of the q -based time derivative and Ψ -based spatial derivative. Brutsaert [1971] was one of the first authors to use this equation to model steep wetting fronts and obtained good mass conservation as well. The strategy of evaluating the water content changes over one time step directly from the change of the pressure head [Celia et al., 1990] results in much improved mass balances. However, the use of a mass-conservative method does not guarantee good solutions [Celia et al., 1990].

Usually these three forms of RE are expressed in the local balance form or composed form where the diffusivity term exists as a single term in the PDE [Hanks and Bowers, 1962; Whisler and Klute, 1965; Rubin and Steinhardt, 1963; Pikul et al., 1974; Vaculin et al., 1976]. Haverkamp and Vaculin [1981] found that although the composed form is mass conservative, it does not give the most accurate solutions due to the influence of weighting (of conductivity values). In addition to the composed form, one can develop the decomposed forms of Richards Equation (where the diffusivity term exists as two terms in the PDE), though Haverkamp and Vaculin [1981] did not recommend their use. They found that the decomposed form was strongly affected by weighting errors, and was limited insofar as the choice of mesh ratio was concerned [Liakopoulos, 1966; Molz and Remson, 1970]. Furthermore, they advocated that the RE with a Kirchhoff integral transformation yields better results than that found on either the composed or the decomposed forms of Richard's equation.

In this chapter, an ordinary differential equation implementation of the method of lines or ODE/MOL approach is used on the various forms of Richards' equation. The aim is to find the best form of RE to model variably saturated flow, which can also handle steep moving fronts in an initially very dry soil. The exact solutions for constant flux infiltration presented by Sander et al. [1988], Models A and B are used to enable

rigorous testing of the computer-based numerical solutions using the error measures introduced in section 3.5.

5.2 THE ODE/MOL SOLUTION OF RICHARD'S EQUATION

For practical purposes, it is usual to model flow in both the saturated and unsaturated zone. This will rule out a pure water content-based formulation [Forsyth et al., 1995]. Hence, the pressure and mixed forms of RE along with its decomposed forms are normally used for modelling variably saturated flow through soil. For completeness, the Θ -based form of RE will also be investigated in this chapter. A total of 18 forms of RE which could model variably saturated flow, and 3 forms of the Θ form were tested. These were grouped into 6 categories as follows

Ψ -based Composed Form of RE (G1)

$$\text{Case 1} \quad \Psi_t = \left\{ (K\Psi_z)_z - K_z \right\} / C \quad (5.1)$$

$$\text{Case 2} \quad \Psi_t = \left\{ (K\Psi_z)_z - \Theta_\Psi K_\Theta \Psi_z \right\} / C \quad (5.2)$$

$$\text{Case 3} \quad \Psi_t = \left\{ (K\Psi_z - K)_z \right\} / C \quad (5.3)$$

$$\text{Case 4} \quad \Psi_t = \left\{ (K(\Psi_z - 1))_z \right\} / C \quad (5.4)$$

Ψ -based Decomposed Form of RE (G2)

$$\text{Case 5} \quad \Psi_t = \left\{ \Psi_z K_z + K\Psi_{zz} - K_z \right\} / C \quad (5.5)$$

$$\text{Case 6} \quad \Psi_t = \left\{ \Psi_z K_z + K\Psi_{zz} - (\Theta_\Psi K_\Theta \Psi_z) \right\} / C \quad (5.6)$$

$$\text{Case 7} \quad \Psi_t = \left\{ \Psi_z (\Theta_\Psi K_\Theta \Psi_z) + K\Psi_{zz} - (\Theta_\Psi K_\Theta \Psi_z) \right\} / C \quad (5.7)$$

$$\text{Case 8} \quad \Psi_t = \left\{ (\Psi_z - 1) K_z + K\Psi_{zz} \right\} / C \quad (5.8)$$

$$\text{Case 9} \quad \Psi_t = \left\{ (\Psi_z - 1) (\Theta_\Psi K_\Theta \Psi_z) + K\Psi_{zz} \right\} / C \quad (5.9)$$

Mixed Composed Form of RE (G3)

$$\text{Case 10} \quad \Theta_t = \left\{ (K\Psi_z)_z - K_z \right\} \quad (5.10)$$

$$\text{Case 11} \quad \Theta_t = \left\{ (K\Psi_z)_z - \Theta_\Psi K_\Theta \Psi_z \right\} \quad (5.11)$$

$$\text{Case 12} \quad \Theta_t = \left\{ (K\Psi_z - K)_z \right\} \quad (5.12)$$

$$\text{Case 13} \quad \Theta_t = \left\{ (K(\Psi_z - 1))_z \right\} \quad (5.13)$$

Mixed Decomposed Form of RE (G4)

$$\text{Case 14} \quad \Theta_t = \{\Psi_z K_z + K\Psi_{zz} - K_z\} \quad (5.14)$$

$$\text{Case 15} \quad \Theta_t = \{\Psi_z K_z + K\Psi_{zz} - (\Theta_\Psi K_\Theta \Psi_z)\} \quad (5.15)$$

$$\text{Case 16} \quad \Theta_t = \{\Psi_z (\Theta_\Psi K_\Theta \Psi_z) + K\Psi_{zz} - (\Theta_\Psi K_\Theta \Psi_z)\} \quad (5.16)$$

$$\text{Case 17} \quad \Theta_t = \{(\Psi_z - 1)K_z + K\Psi_{zz}\} \quad (5.17)$$

$$\text{Case 18} \quad \Theta_t = \{(\Psi_z - 1)(\Theta_\Psi K_\Theta \Psi_z) + K\Psi_{zz}\} \quad (5.18)$$

Theta Composed Form of RE (G5)

$$\text{Case 19} \quad \Theta_t = (D\Theta_z)_z - K_z \quad (5.19)$$

$$\text{Case 20} \quad \Theta_t = (D\Theta_z)_z - K_\Theta \Theta_z \quad (5.20)$$

$$\text{Case 21} \quad \Theta_t = (D\Theta_z)_z - K_\Theta \Theta_\Psi \Psi_z \quad (5.21)$$

Theta Decomposed Form of RE (G6)

$$\text{Case 22} \quad \Theta_t = D_z \Theta_z + D\Theta_{zz} - K_z \quad (5.22)$$

$$\text{Case 23} \quad \Theta_t = D_z \Theta_z + D\Theta_{zz} - K_\Theta \Theta_z \quad (5.23)$$

$$\text{Case 24} \quad \Theta_t = D_z \Theta_z + D\Theta_{zz} - K_\Theta \Theta_\Psi \Psi_z \quad (5.24)$$

It can be seen that all these cases are obtained through rearranging the original equations (3.5) and (3.6), using the chain rule for differentiation, and that mathematically they are all equivalent. Note that K is given by (3.9) and that the K_z term has, in certain cases, been replaced with $\Theta_\Psi K_\Theta \Psi_z$. When K_z is unchanged, then the boundary condition requires modification for implementation in the MOL. Given

that $t > 0$, $z = 0$, $\Psi_z = 1 - \frac{Q}{K}$, we can rewrite this boundary condition in terms of K_z

using the chain rule as follows

$$\begin{aligned} K_z &= \left(1 - \frac{Q}{K}\right) / (\Theta_K \Psi_\Theta) \\ &= \left(1 - \frac{Q}{K}\right) * \Theta_\Psi K_\Theta. \end{aligned} \quad (5.25)$$

Similarly, we can rewrite the boundary conditions in terms of Θ_z and D when the Θ -based form of RE is used –

$$\Theta_z = (K - Q) / D. \quad (5.26)$$

5.3 NUMERICAL EXPERIMENTS

All of the 24 models were handled using the template discussed in section 4, and the numerical solutions calculated as specified in section 3.5. The time integration was over regions $0 \leq z \leq 50$ cm (see section 3.3), where these short-term integrations were from 0 to 0.3625 minutes in model time. The long-term integration was over the interval 0 to 36.25 minutes, and $0 \leq z \leq 150$ (see section 3.3). These ranges were selected to ensure that the front was not affected by the artificial boundary (see (3.25)). The models were also run for two values of the tunable parameter \mathbf{u} , i.e. $\mathbf{u}=0.1$ and $\mathbf{u}=0.99995$, corresponding to Models A and B as defined in section 3.3. The performance measures of section 3.5 were calculated using the analytic solutions of Sander et al.[1998].

The experiments were designed to test the performance of the different forms of RE with a steep ($\mathbf{u}=0.99995$) and gentle ($\mathbf{u}=0.85$) moving front solution for short-term and long-term integration. For all cases, a tolerance of $tol = 10^{-10}$, was used in the MATLAB ODE45 solver routines. Tight error tolerances have been selected so that the spatial error dominates. A spatial step of $dz = 0.1$ and $dz = 0.3$ were also used for the short-term integration and long-term integration respectively.

5.4 RESULTS AND DISCUSSION

Table 5.1 shows that the results at a model time of 0.3625 minutes (short-term integration) for the solution with a gentle moving front. These results show that the mixed forms and the Θ -based form of RE (cases 10-18 and 19-24) perform better in terms of CPU time with an improvement of 70%, although their accuracy and GME are similar to that of the pressure forms (cases 1-9). The reasons for this improvement could be that the Θ -based form restrains the size of Θ and may control the gradient a bit better than the pressure form, which allows the pressure to get large in magnitude.

Table 5.1 also shows that the decomposed forms yield better accuracy and GME value in dealing with a gentle moving front in dry initial condition as shown by cases 7, 9, 16 and 18. Fig. 5.3 shows the movement of the initial step front from 0 to 0.3625 minutes in model time.

Table 5.1 Errors and CPU-Time at model time 0.3625 min ($u=0.85$).

GROUP		MRE $* 10^{-4}$	GME (%)	CPU Time(s)	GROUP		MRE $* 10^{-4}$	GME (%)	CPU Time(s)
G1	Case 1	2.2	0.1196	7.91	G3	Case 10	2.2	0.1196	1.92
	Case 2	2.3	0.1251	7.58		Case 11	2.3	0.1251	2.03
	Case 3	2.3	0.1252	7.85		Case 12	2.3	0.1252	1.87
	Case 4	2.3	0.1252	7.75		Case 13	2.3	0.1252	1.92
G2	Case 5	1.4	0.8161	10.16	G4	Case 14	14	0.8161	2.91
	Case 6	1.4	0.8210	10.38		Case 15	14	0.8210	3.02
	Case 7	0.56	0.0193	10.01		Case 16	0.56	0.0193	3.40
	Case 8	14	0.8241	10.16		Case 17	14	0.8241	2.64
	Case 9	0.57	0.0199	10.38		Case 18	0.57	0.0199	3.41
G5	Case 19	2.8	0.1177	1.64	G6	Case 22	2.4	0.0614	1.87
	Case 20	2.8	0.1187	1.76		Case 23	2.4	0.0624	1.92
	Case 21	4.6	0.1649	1.70		Case 24	4.6	0.1086	1.93

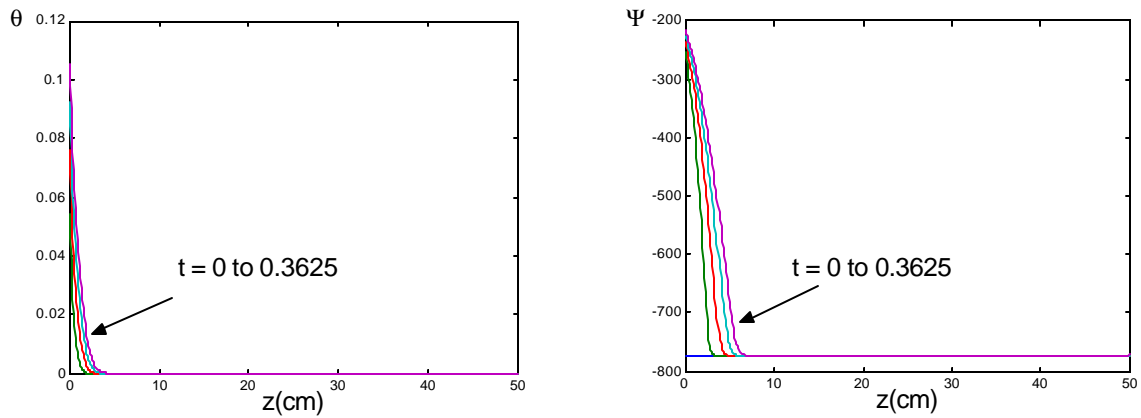


Fig. 5.1 q and Ψ profile with respect to z from 0 to 0.3625 min ($u=85$).

Table 5.2 shows the results at a model time of 36.25 minutes (long-term integration) for the solution with the gentle moving front. The results show that all forms of RE perform well and there are fewer differences in their MRE values. This is possibly because the transition from the relatively steeper moving front (at the start of integration) to a gentle one is short, and the efficient handling of the gentler moving front offsets the computational gain in handling the steeper front. However, with respect

to GME and CPU time, the composed models (cases 1-4, 10-13, 19-21) generally perform better than the decomposed models (cases 5-9, 14-18, 22-24). Fig. 5.2 shows that the steepness of the infiltration front is abating from 0 to 36.25 minutes in model time. Thus, the integration of a gentle moving front posed no difficulty for these 24 forms of RE. Also note that the GME in Table 5.2 is 10 times better than that in Table 5.1.

Table 5.2 Errors and CPU-Time at 36.25 min ($u=0.85$).

GROUP		MRE * 10^{-4}	GME (%)	CPU Time(s)	GROUP		MRE * 10^{-4}	GME (%)	CPU Time(s)
G1	Case 1	0.65	0.0015	34	G3	Case 10	0.76	0.0016	22
	Case 2	0.91	0.0015	35		Case 11	1.1	0.0016	24
	Case 3	1.1	0.0073	34		Case 12	1.2	0.0022	23
	Case 4	1.1	0.0023	35		Case 13	1.2	0.0022	23
G2	Case 5	9.4	0.0903	41	G4	Case 14	9.4	0.0903	27
	Case 6	9.6	0.0917	42		Case 15	9.6	0.0917	25
	Case 7	1.3	0.0113	41		Case 16	1.3	0.0113	28
	Case 8	9.9	0.0942	41		Case 17	9.9	0.0942	26
	Case 9	1.5	0.0134	39		Case 18	1.5	0.0134	26
G5	Case 19	0.91	0.0018	24	G6	Case 22	0.94	0.0023	25
	Case 20	1.0	0.0022	25		Case 23	0.74	0.0063	25
	Case 21	97	0.1301	25		Case 24	97	0.1344	26

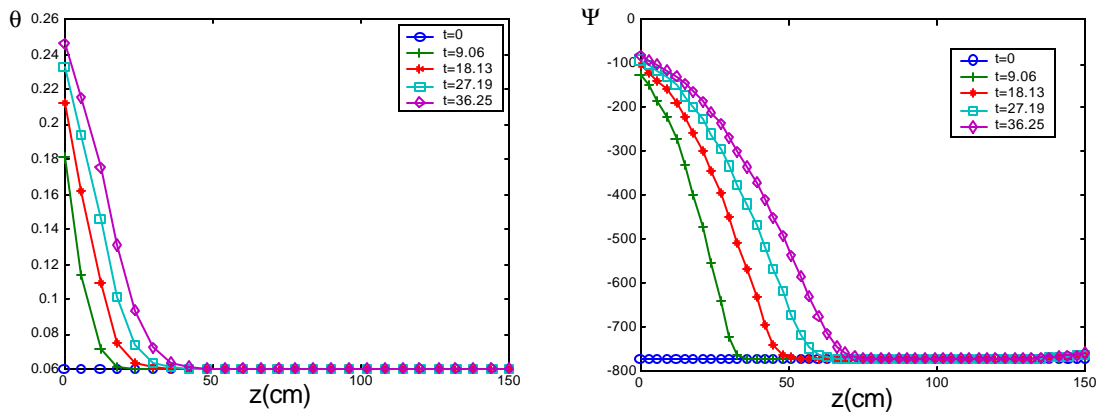


Fig. 5.2 q and Ψ profile with respect to z from 0 to 36.25 min ($u=0.85$).

Table 5.3 shows the results at a model time of 0.3625 minutes (short-term integration) for a steeper moving front model as shown in Fig. 5.3, as compared to Fig. 5.1. Table 5.3 shows that the Θ based forms of RE (cases 10-18, 19-24) perform better in terms of CPU time with an improvement of 40% over the Ψ based forms, although

their accuracy and GME values are comparable to that of the Ψ based forms. Moreover, the mixed and the Θ based forms (cases 1-4, 10-13, 19-24) outperform the decomposed form (cases 5-9, 14-18) by a ten-fold improvement in its MRE and GME. These results are similar to those obtained in Table 5.2 and Table 5.1.

Table 5.3 Errors and CPU-Time at 0.3625 min ($u=0.99995$).

GROUP		MRE $* 10^{-3}$	GME (%)	CPU Time(s)	GROUP		MRE $* 10^{-3}$	GME (%)	CPU Time(s)
G1	Case 1	5.3	0.1231	6.75	G3	Case 10	5.3	0.1226	3.13
	Case 2	5.3	0.1231	6.92		Case 11	5.3	0.1227	3.40
	Case 3	5.3	0.1231	6.70		Case 12	5.3	0.1227	3.18
	Case 4	5.3	0.1231	6.40		Case 13	5.3	0.1227	3.13
G2	Case 5	35	2.9376	9.23	G4	Case 14	35	2.9376	5.71
	Case 6	35	2.9376	9.67		Case 15	35	2.9376	5.55
	Case 7	9.3	2.9376	9.12		Case 16	9.3	0.8133	6.80
	Case 8	35	0.8435	9.02		Case 17	35	2.9376	5.55
	Case 9	9.3	0.8133	9.53		Case 18	9.3	0.8133	6.42
G5	Case 19	6.5	0.1285	3.18	G6	Case 22	7.1	0.7332	3.91
	Case 20	6.5	0.1285	3.29		Case 23	7.1	0.7332	3.84
	Case 21	6.5	0.1286	3.35		Case 24	7.1	0.7333	4.12

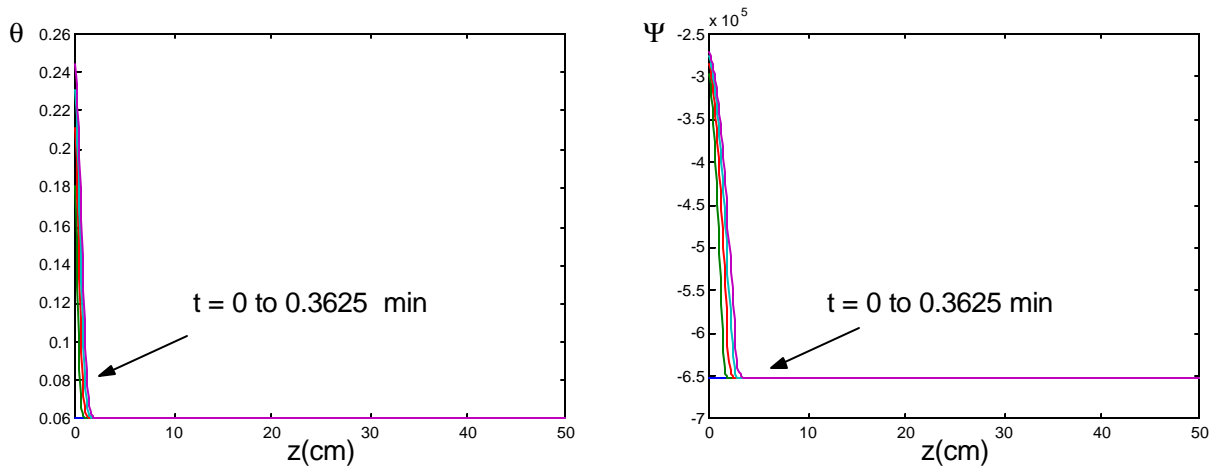


Fig. 5.3 q and Ψ profile with respect to z from 0 to 0.3625 min ($u=0.99995$).

Table 5.4 shows the results at a model time of 36.25 minutes (long-term integration) for a steeper front model as shown in Fig. 5.4. The results show that the composed form of RE outperforms the decomposed form of RE (Ψ based, Θ based and Mixed based RE), yielding much better GME and MRE values. In particular, the Θ based composed form yields the best MRE value but at the expense of the CPU time

which is 10 times larger. This probably due to more calls to the ‘rhs’ due to time stepping constraints. On the other hand, for the decompose form of RE (cases 7,9, 16 and 18) the integration proceeds very slowly at 0.5607 min. due to the integrator being constrained by the imposed tolerance. These cases clearly show that the composed form of RE is better than the decomposed form of RE for long-term integration with a highly steep moving-front solution.

Table 5.4 Errors and CPU-Time at 36.25 min ($u=0.99995$).
 @ implies integration proceeds very slowly at 0.5607 min.

GROUP		MRE * 10 ⁻²	GME (%)	CPU Time(s)	GROUP		MRE * 10 ⁻¹	GME (%)	CPU Time(s)
G1	Case 1	5.8	1.1*10 ⁻⁴	286	G3	Case 10	0.61	3.7*10 ⁻⁵	304
	Case 2	5.8	7.8*10 ⁻⁴	302		Case 11	0.61	7.1*10 ⁻⁴	289
	Case 3	5.8	9.3*10 ⁻⁵	293		Case 12	0.61	2.2*10 ⁻⁵	298
	Case 4	5.8	9.3*10 ⁻⁵	300		Case 13	0.61	2.2*10 ⁻⁵	295
G2	Case 5	47	9.9*10 ⁻¹	316	G4	Case 14	4.7	9.9*10 ⁻¹	300
	Case 6	47	9.9*10 ⁻¹	342		Case 15	4.7	9.9*10 ⁻¹	316
	Case 7	@				Case 16	@		
	Case 8	47	9.9*10 ⁻¹	875		Case 17	4.7	9.9*10 ⁻¹	302
	Case 9	@				Case 18	@		
G5	Case 19	1.5	2.4*10 ⁻⁶	2841	G6	Case 22	1.2	3.1*10 ⁻¹	4101
	Case 20	1.5	1.9*10 ⁻⁴	4525		Case 23	1.2	3.1*10 ⁻¹	3979
	Case 21	1.5	5.7*10 ⁻⁵	3464		Case 24	1.2	3.1*10 ⁻¹	4074

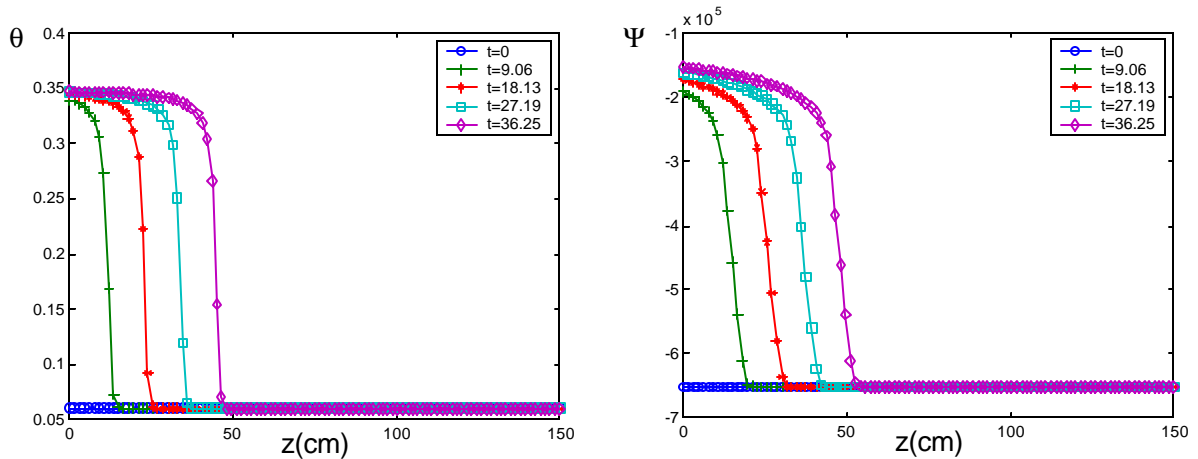


Fig. 5.4 θ and Ψ profile with respect to z from 0 to 36.25 min ($u=0.99995$).

5.5 CONCLUSIONS

We have shown that using the ODE/MOL approach is generally efficient and robust for solving the mixed, Θ forms and Ψ forms of Richard’s Equation, regardless of whether it is the decomposed or composed form. The template discussed in section 4

is readily able to handle all of the cases considered in this chapter. The boundary conditions were also easily handled. All the 24 forms of RE performed well when there is a gentle sloping wetting front, but the mixed composed forms were superior for handling the case where a sharp wetting front develop. However, the Θ based composed form achieved the best MRE value at the expense of the CPU time, which is 10 times larger.

Also, the constitutive relations might differ in other RE models, but the 24 different forms of RE gives us choices to deal with difficult variably saturated infiltration problem. The replacement of one form with another is simple and no additional computational cost is added. These different forms can made a different as one form might give better solutions than others, as shown in this chapter, especially for dealing with steep moving front in very dry initial condition.

Higher Order Schemes

It is a good morning exercise for a research scientist to discard a pet hypothesis every day before breakfast. It keeps him young.

Konrad Lorenz

6.1 INTRODUCTION

Higher order finite difference schemes, e.g. fourth order and higher, are being used to solve both hyperbolic and parabolic partial differential equations (PDE) [Carpenter et al., 1997; Olson, 1993; Strad, 1991; Abarbenel and Ditkowski, 1997; Schiesser, 1991]. These higher order finite difference representations of derivative terms offer a means of obtaining more accurate numerical solutions with relatively large step sizes, and less computational burden than for the commonly used second order finite difference approximations, using relatively small step sizes. Finite difference methods are attractive because of the relative ease of implementation and high level of flexibility: this is especially true in the application of an automated differentiation matrix used in the vectorized form of the Method of Lines (MOL).

Derivatives can be represented by symmetric e.g., central, or asymmetric e.g., forward or backward, finite difference schemes. Thus upwinding schemes can be readily applied to handle convective terms in the PDE. Following [Wilson and Turcotte, 1994], these representations can be expressed in vector form, and the expressions also include estimates of the truncation errors. This is a suitable form for use with the template and the vectorized Method of Lines

Derivative representations can be represented by the product of the differentiation matrix, and a vector of function values at the nodes of the grid [Lee et al., 1998]. The rows of the differentiation matrix contain finite difference representations of first derivatives, and higher derivatives can be obtained by repeated application of the differentiation matrix. The rows of the differentiation matrix have traditionally been representations of second order finite difference approximations, although higher order representations are now being used [Carpenter et al., 1997; Olson, 1993; Strad, 1991; Abarbenel and Ditkowski, 1997; Schiesser, 1991].

In this chapter, we investigate the 2nd to 16th order derivative representations using differentiation matrices in MOL, and the upwind representation to handle the convective terms. The upper limit of the 16th order was selected to fully test the properties of the scheme and also because the machine precision used is of 15 significant figures. The results will suggest an optimal order for some problems, and the extension to the 16th order allows a proper determination of the optimum. The aim is to examine whether a higher order scheme is better at handling highly convective flows with steep moving fronts.

6.2 NUMERICAL DIFFERENTIATION

Consider the 3-point lattice

$$x_i = x^* + \mathbf{a}_i h, \quad i = 0, 1, 2 \quad (6.1)$$

where h is fixed, x^* is some starting point, and \mathbf{a}_i (with $\mathbf{a}_0=0$) are arbitrary. The \mathbf{a}_i , are consecutive integers starting from $\mathbf{a}_0=0$ and the lattice of points are uniformly spaced. The Taylor series representation for a sufficiently smooth function, $F(x)$, can be represented by

$$\begin{aligned}
F(x_i) &= F(x_0 + \mathbf{a}_i h), \\
&\approx F(x_0) + \mathbf{a}_i h F'(x_0) + \frac{(\mathbf{a}_i h)^2}{2!} F''(x_0) \\
&\quad + \frac{(\mathbf{a}_i h)^3}{3!} F'''(x_0) + \frac{(\mathbf{a}_i h)^4}{4!} F^{(4)}(x_0) + O(h^5),
\end{aligned} \tag{6.2}$$

where the expansions are taken about x_0 . In matrix form

$$\underline{F} = C \underline{F}_p + \frac{h^3}{3!} F^{(3)}(x_0) \underline{U}_3 + \frac{h^4}{4!} F^{(4)}(x_0) \underline{U}_4 + \mathbf{o}(h^5), \tag{6.3}$$

where

$$\begin{aligned}
\underline{F} &= [F(x_0), F(x_1), F(x_2)]^T, \\
\underline{F}_p &= [F(x_0), F'(x_0), F''(x_0)]^T, \\
\underline{U}_j &= [\mathbf{a}_0^j, \mathbf{a}_1^j, \mathbf{a}_2^j]^T, \quad , j = 3, 4
\end{aligned}$$

and the matrix C is given by

$$C = \begin{bmatrix} 1 & \mathbf{a}_0 h & (\mathbf{a}_0 h)^2 / 2! \\ 1 & \mathbf{a}_1 h & (\mathbf{a}_1 h)^2 / 2! \\ 1 & \mathbf{a}_2 h & (\mathbf{a}_2 h)^2 / 2! \end{bmatrix}. \tag{6.4}$$

Solving the system for \underline{F}_p , the vector of derivative values at x_0 , yields

$$\underline{F}_p = C^{-1} \underline{F} - \frac{h^3}{3!} F^{(3)}(x_0) C^{-1} \underline{U}_3 - \frac{h^4}{4!} F^{(4)}(x_0) C^{-1} \underline{U}_4 + O(h^5), \tag{6.5}$$

Then the generalized derivative vector may be approximated by

$$\underline{F}_p \approx B \underline{F}, \tag{6.6}$$

where $B = C^{-1}$ is the differentiation data structure, and the subsequent terms in (6.5)

represent the remainder or truncation error terms. Note that C can also be written as

$$C = \begin{bmatrix} 1 & \mathbf{a}_0 & (\mathbf{a}_0)^2 / 2! \\ 1 & \mathbf{a}_1 & (\mathbf{a}_1)^2 / 2! \\ 1 & \mathbf{a}_2 & (\mathbf{a}_2)^2 / 2! \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & h & 0 \\ 0 & 0 & h^2 \end{bmatrix}, \tag{6.7}$$

and this representation can be used to rewrite (6.5) in other forms. Thus B is a derivative data structure, which when applied to \underline{F} , yields the derivatives of various

orders at x_0 . The different order derivative representations at x_0 can easily be extracted from these derivative matrices. The theoretical development of (6.1) can be readily extended to more nodes, and the Taylor series in (6.2) can be taken to any order. Thus (6.6) can be written to any order, e.g. the 20th order, with an appropriate definition for B , the differentiation data structure. Similarly, if the derivative representations at other nodal points are required, these can be derived using the above theoretical development. Wilson and Turcotte [1994] have devised the above method for generation of finite difference formulas for any order of derivative and to any order of accuracy on one-dimensional grids with arbitrary spacing.

Many researchers had attempted to generate finite difference formulas using different ways, i.e. Lagrangian interpolation polynomials, least-squares polynomial approximation and symbolic manipulation [Bickley, 1941; Keller and Pereyra, 1978; Lakin, 1986; Fornberg, 1988]. However, we considered the method devised by Wilson and Turcotte [1994] to be the best in generating the differentiation data structure. It is simple and vectorized for matrix computation. Furthermore, Wilson and Turcotte [1994] have written a compact MATLAB code that derives any order of the derivative data structure at any nodal points and this code is being utilized in this thesis. With the differentiation data structure, any order of derivative stencils can be implemented into the template in section 4, to generate appropriate differentiation matrices.

6.3 FIRST ORDER DIFFERENTIATION MATRIX (FODM)

As noted in Lee et al. [1998], the nodes in a lattice or mesh area can be assigned different computational stencils using the rows of the differentiation data structure of the specified order of accuracy. Also, each derivative term in the right hand side of the PDE can be approximated using an appropriate row of the differentiation data structure, leading to different forms of FODM or differentiation matrix for each term. Thus appropriate forward and backward approximations can be used to obtain upwinding

schemes to handle convective terms in the original PDE. Let A_{zup} represent a non symmetric FODM to be used for upwinding schemes along the z -axis, for handling convective terms. The matrix A_z represents a symmetric (or central difference) FODM to be used in stage wise differentiation to be applied to diffusive terms following the notation in chapter 4 [Lee et al., 1998].

Both the A_z and A_{zup} matrices have a precise pattern of “location of the derivative stencil” that can be utilized for the creation of automatic routines yielding the FODM of any order of accuracy required. Two FODM templates were created for the even and odd order approximation of the first order derivatives. The two templates comprise the odd order FODM template and the even order FODM template.

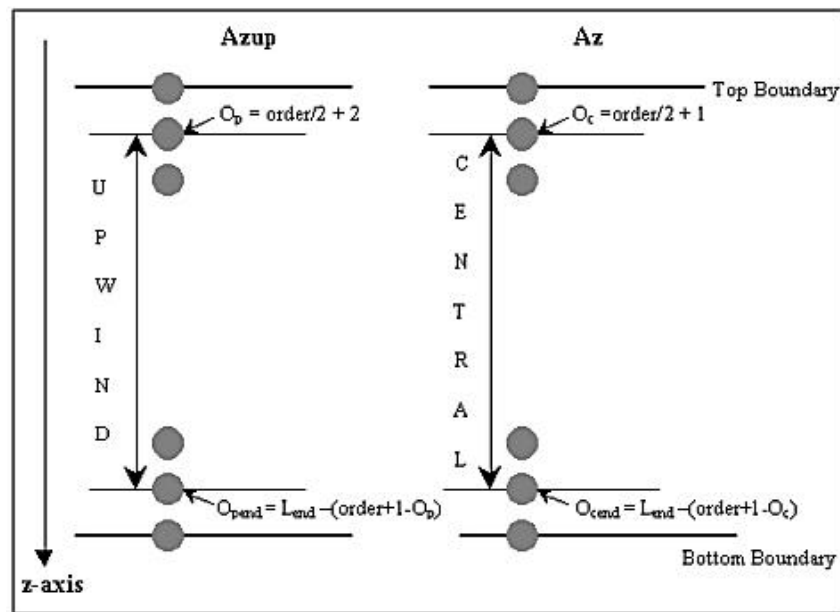


Fig. 6.1 Even Order FODM Template

First consider the even order FODM template. Fig. 6.1 shows the placement of the upwind and central difference stencils in the physical domain. The assignment of the upwind stencil starts at O_p and terminates at O_{pend} in the physical domain. O_p and O_{pend} contain the number of nodes away from the top boundary. L_{end} is the number of nodes column wise in the physical domain, and the ‘order’ is the order of accuracy

required. With these statistics, the assignment of any order of upwind difference stencil can be easily inserted in Azup.

There is a direct correspondence between the physical region and Azup. Azup is a square matrix whose row counter value equals the discretized numbering of the nodal points where the derivative is being approximated in the physical region. Its column counter value equals the discretized numbering of the adjacent nodal points. The discretized numbers of the specific nodes give the row indices of Azup and the discretized numbers of an adjacent node gives the column index of Azup. The diagonal index coincides with the nodes in the derivative stencil where the derivative is being approximated. Thus knowing the placement of the derivative stencil in the physical domain gives the exact location of the placement of the stencil coefficients in Azup or any differentiation matrix via the discretizing numbering of the nodes. The nodal points for the upwind domain may be assigned with the respective order of backward or forward difference stencils of the required order. Similarly, the placement of the central difference stencil in Az is obtained using the above template that ranges from O_c to O_{cend} . O_c and O_{cend} contain the number of nodes away from the top boundary.

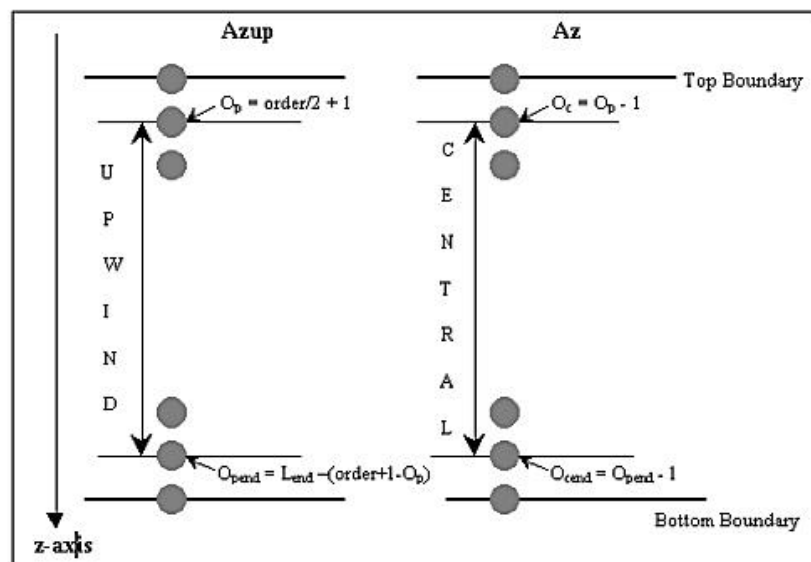


Fig. 6.2 Odd Order FODM Template

The odd order FODM template is shown in Figure 6.2. With the same methodology, the Azup and Az matrices are created from the odd order FODM template. For the odd order FODM template, central differencing is not used and the odd order or asymmetric derivative stencils of the backward differencing is used instead for the creation of the Az matrix. These FODM templates are fully utilized and implemented in the first order differentiation matrix (FODM or Automatic Differencer) in chapter 4.

6.4 NUMERICAL EXPERIMENTS

The Burgers' Model (see section 3.2) and Sander's Model B (see section 3.3) are used in this chapter for the numerical models. Both models were implemented using the template approach, and solved for finite difference representation of orders 2 to 16. The models were also run for various values of the tunable parameter n (for Burgers' Equation) and u (for Sanders model B). The experiments were designed to test the performance of the different orders on problems where the steepness of the moving front can be tuned through n or u .

6.5 RESULTS AND DISCUSSION

Example 1. Burgers' Model of Advective-Convective Equation's Results.

Fig. 6.3 and 6.4 shows the analytic solution profile for $n=0.1$ and 0.001 at model times from 0 to 0.5. Note that the steepness of the fronts varies with the values of n . For $n=0.001$, the solution profile depicts a very steep moving front moving forward with time. Table 6.1 to Table 6.5 show the statistics of the integrator where n ranges in value from 0.001 to 2 for Burgers' equation. In particular, "Failed attempts" in the table means the number of failed attempts by the integrator to evaluate a good solution, not satisfying the error control in ODE in the course of computation. For $n=1$ and 2, [Tables 6.1 and 6.2] the results show a rapid increase in accuracy as the order of

the finite differencing increases. The accuracy peaks at approximately order 8-10, and the even order difference approximations yield slightly better accuracy than the neighboring odd order approximations. As the order increases beyond the peak, the accuracy declines, probably due to accumulating rounding error. For the higher orders, greater than 15 or 16, the number of function calls also increases and this is indicative of increasing rounding error. Hence the number of time steps is also increasing as the solver shortens the step length to control accuracy.

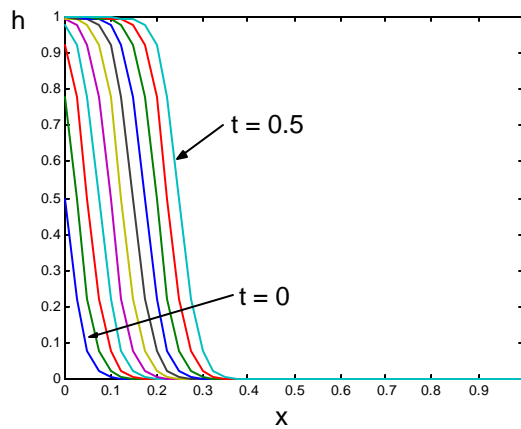


Fig. 6.3 Solution profile in Ψ for BE, $n = 0.01$.

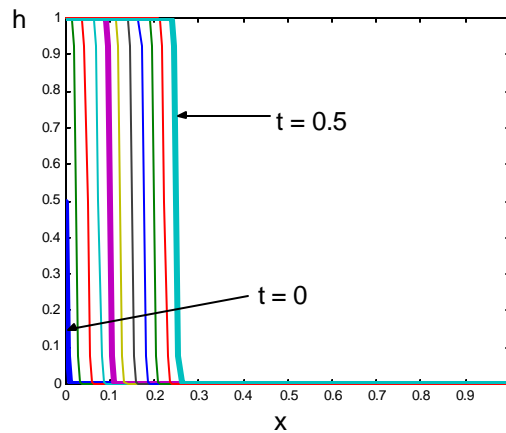


Fig. 6.4. Solution profile in Ψ for BE, $n = 0.001$.

Table 6.1 Example 1, MRE for $n = 1$ where integration is from 0 to 0.5 second for a spatial step of 1/20 and $tol = 10^{-16}$.

Order	MRE $n = 1$	CPU Time (s)	Successful steps	Failed attempts	Derivative function calls
2	$1.2 * 10^{-5}$	1.9	709	3	4273
3	$4.5 * 10^{-3}$	2.5	898	2	5401
4	$1.9 * 10^{-9}$	2.6	918	2	5521
5	$1.6 * 10^{-8}$	2.7	964	2	5797
6	$3.2 * 10^{-12}$	2.8	973	2	5851
7	$5.5 * 10^{-12}$	2.8	968	2	5821
8	$2.3 * 10^{-14}$	2.8	954	2	5737
9	$6.0 * 10^{-14}$	2.6	914	2	5497
10	$2.6 * 10^{-14}$	3.5	870	2	5233
11	$3.2 * 10^{-13}$	2.1	723	2	4351
12	$2.2 * 10^{-13}$	2.3	668	2	4021
13	$6.1 * 10^{-13}$	2.4	774	39	4879
14	$2.1 * 10^{-12}$	2.7	874	52	5557
15	$1.1 * 10^{-11}$	3.2	1021	46	6403
16	$1.5 * 10^{-11}$	4.4	1401	56	8743

Table 6.2 Example1, MRE for $n = 2$ where integration is from 0 to 0.5 second for a spatial step of 1/20 and $tol = 10^{-16}$.

Order	MRE $n = 2$	CPU Time (s)	Successful steps	Failed attempts	Derivative function calls
2	$1.7e-07$	2.0	733	5	4429
3	$8.7e-04$	2.6	917	3	5521
4	$7.5e-11$	2.7	957	3	5761
5	$2.9e-10$	2.9	1004	3	6043
6	$2.3e-14$	2.9	1002	3	6031
7	$2.3e-13$	2.9	989	3	5953
8	$1.6e-14$	2.8	947	3	5701
9	$4.2e-13$	2.9	867	3	5221
10	$6.9e-14$	2.1	707	3	4261
11	$3.6e-13$	3.2	842	109	5707
12	$1.3e-13$	2.9	928	68	5977
13	$1.1e-12$	3.4	1043	87	6781
14	$1.8e-13$	3.7	1203	60	7579
15	$5.8e-12$	5.2	1669	65	10405
16	$7.9e-12$	8.1	2608	72	16081

For the hyperbolic cases with $n = 0.1, 0.01, \text{ and } 0.001$, [Tables 6.3,6.4,and 6.5] the results show a similar trend of improvement, but at a lesser overall accuracy. Moreover, the use of $tol = 10^{-16}$ in the ODE45 integrator, solving these hyperbolic cases, yields similar accuracy to those using $tol = 10^{-16}$ (results for $tol = 10^{-16}$ not shown here). However, the CPU time is drastically shortened, i.e. 10-40 times faster. Hence

tighter error tolerances in the ODE45 integrator greater than 10^{-6} only improve the accuracy marginally at the expense of the CPU time for a hyperbolic dominated PDE. Furthermore, a finer spatial step, 1/200 has to be used for a highly hyperbolic dominated PDE to resolve the higher steep front and achieve an accurate result.

Table 6.3 Example 1, MRE for $n=0.1$ where integration is from 0 to 0.5 second for a spatial step of 1/20 and $tol = 10^{-6}$.

Order	MRE $n=0.1$	CPU Time (s)	Successful steps	Failed attempts	Derivative function calls
2	$1.6 * 10^{-2}$	0.1	20	1	127
3	Failure at t=4.623013e-001s unable to meet tolerance.				
4	$2.8 * 10^{-4}$	0.1	24	1	151
5	$1.1 * 10^{-3}$	0.1	25	1	157
6	$2.6 * 10^{-5}$	0.1	23	1	145
7	$3.7 * 10^{-3}$	0.1	23	1	145
8	$6.4 * 10^{-5}$	0.1	22	1	145
9	$9.2 * 10^{-5}$	0.1	23	1	163
10	$5.0 * 10^{-5}$	0.1	26	1	169
11	$3.4 * 10^{-4}$	0.1	28	0	175
12	$9.9 * 10^{-6}$	0.1	29	1	181
13	$2.1 * 10^{-4}$	0.1	29	1	181
14	$3.3 * 10^{-5}$	0.1	29	1	181
15	$7.2 * 10^{-6}$	0.1	29	2	187
16	$5.5 * 10^{-6}$	0.1	30	2	193

Table 6.4 Example 1, MAE for $n=0.01$ where integration is from 0 to 0.5 second for a spatial step of 1/200 and $tol = 10^{-6}$.

Order	MAE $n=0.01$	CPU Time (s)	Successful steps	Failed attempts	Derivative function calls
2	$1.7 * 10^{-2}$	0.6	137	0	913
3	Stop at 0.02s (a).				
4	$1.1 * 10^{-4}$	0.7	161	0	1027
5	Stop at 0.04s (a).				
6	$1.3 * 10^{-6}$	0.9	179	27	1237
7	Stop at 0.07s (a).				
8	$2.1 * 10^{-6}$	1.2	198	38	1417
9	Stop at 0.08s (a).				
10	$4.6 * 10^{-7}$	1.3	222	26	1489
11	Stop at 0.1s (a).				
12	$2.9 * 10^{-7}$	1.8	245	57	1813
13	Stop at 0.11s (a).				
14	$1.9 * 10^{-7}$	2.1	275	64	2035
15	Stop at 0.22s (a).				
16	$1.7 * 10^{-7}$	2.3	312	42	2125

(a) unable to meet tolerance.

Table 6.5 Example 1, MAE for $n=0.001$ where integration is from 0 to 0.5 second for a spatial step of 1/200 and $tol = 10^{-6}$.

Order	MAE $n=0.001$	CPU Time (s)	Successful steps	Failed attempts	Derivative function calls
2	$9.9 * 10^{-1}$	1.2	306	2	1849
3	Stop at 0.04s (a).				
4	$4.3 * 10^{-1}$	1.5	381	1	2293
5	Stop at 0.15s (a).				
6	Stop at 0.03s (a).				
7	Stop at 0.02s (a).				
8	Stop at 0.02s (a).				
>9	integration stops at around 0.02s				

(a)unable to meet tolerance.

In these cases, the even order approximations can be computed using the small error tolerance in the solver. However, the odd order finite difference approximations fail as the ODE45 integrator stops; it cannot meet the tolerance requirements (see table 6.4), even when a lower tolerance is used. The odd order finite difference approximations are not symmetrical on the stencils and the truncation error is of lower order than for the even order and symmetric finite difference approximations. There is still an improvement in accuracy as the order initially increases to a less well defined peak or optimum. The accuracy then decreases as the order increases beyond order 12-14. This optimum order yields the better accuracy with comparable CPU time, in relation with the other orders. For the smallest value of $n=0.001$, the odd order integration, at any order, is unable to meet the specified tolerance given to ODE45.

For n less than 0.01 (using the same spatial steps) the BE becomes a singularly perturbed differential equation, and the MOL is unable to solve it – the integrator stops due to the minimum integration step being unable to satisfy the tolerance. Using odd order finite differencing in MOL may lead to an instability problem, which causes the integrator to stop because it is unable to meet the tolerance limits [Abarbanel and Ditkowski, 1997]. This instability problem is possibly due to use of the backward differencing for the computation of the diffusion term. As pointed out in Abarbanel and

Ditkowski [1997], in a numerical scheme where the differentiation matrix, or rather its symmetric part, is not negative definite, instability may occur in some problems. The results still showed that the even order finite differencing is superior to the odd order finite differencing when solving the hyperbolic dominated Burgers' Equation. The results (Table 6.5) also showed that the 2nd or 4th order finite difference is the best order for a highly viscous Burgers' equation ($n = 0.001$).

Example 2. Soil water model of constant flux infiltration

Table 6.6 Example2, Model A's results for different orders where integration is from 0 to 36.25 min for a spatial step of 0.75 and $tol = 10^{-6}$ ($u = 0.85$).

Order	GMB * 10^{-3}	MRE * 10^{-4}	CPU Time (s)	Successful Steps	Failed Attempts	Function Evaluations
2	12	7.4	1.8	300	1	1 801
4	Integration proceed slowly at 1.17e-3					
6	2.2	2.2	3.7	517	68	3 511
8	2.2	2.2	4.9	579	124	4 219
10	2.2	2.2	5.5	653	101	4 525
12	2.2	2.2	7.2	739	152	5 347
14	2.2	2.3	9.3	972	182	6 925
16	Integration proceed slowly at 0.3016					

Table 6.7 Example2, Model A's results for different orders where integration is from 0 to 36.25 min for a spatial step of 0.75 and $tol = 10^{-6}$ ($u = 0.85$).

Order	GMB * 10^{-3}	MRE * 10^{-4}	CPU Time (s)	Successful Steps	Failed Attempts	Function Evaluations
2	12	7.4	36	6 414	1	38 491
4	Integration proceed slowly at 1.17e-3					
6	2.2	2.2	61	10 020	4	60 145
8	2.2	2.2	65	9 449	3	56 713
10	2.2	2.2	73	9 682	4	58 117
12	2.2	2.2	76	9 930	5	59 611
14	2.2	2.2	126	14 870	214	90 505
16	Integration proceed slowly at 0.3016					

The results for the Model A, which have relative gentle moving-front, are shown in Table 6.6 and 6.7. The Tables show that the use of a higher order scheme for this case does improve accuracy at the expense of the CPU time. The improvement is marginal at best from the 2nd order 2 to the 6th order. In fact, from the 8th order onward, the computational time-steps are drastically reduced, requiring long computation time

to solve the problem, with little or no improvement in its accuracy. Here, the optimal order is the 6nd order, which yields the highest accuracy and a relatively shorter CPU time. Moreover, tight error tolerance in the ODE45 integrator of 10^{-16} only improves the accuracy of the solution marginally at the expense of the CPU time.

Table 6.8 Example2. Model B's results for different orders where integration is from 0 to 36.25 min for a spatial step of 0.75 and $tol = 10^{-6}$ ($u = 0.99995$).

Order	GMB	MRE	CPU Time (s)	Successful Steps	Failed Attempts	Function Evaluations
2	$3.7 \cdot 10^{-2}$	$6.9 \cdot 10^{-1}$	222	40 259	2 642	257 407
4	4 th onwards, integration stops between 10^{-4} and 10^{-3} , unable to meet tolerance.					

Table 6.9 Example2. Model B's results for different orders where integration is from 0 to 36.25 min for a spatial step of 0.75 and $tol = 10^{-10}$ ($u = 0.99995$).

Order	GMB	MRE	CPU Time (s)	Successful Steps	Failed Attempts	Function Evaluations
2	$3.7 \cdot 10^{-2}$	$6.9 \cdot 10^{-1}$	464	90 785	13	544 789
4	4 th onwards, integration stops between 10^{-4} and 10^{-3} , unable to meet tolerance.					

Higher values of u , which depict a steep-moving front solution (Model B) was also tried. The results are shown in Tables 6.8 and 6.9. The results show that the use of a higher order scheme does not work for Model B. In fact, from the 4th order onward, the computation stops at about between 10^{-4} and 10^{-3} in model time, due to the minimum integration step being not able to satisfy the tolerance in the ODE integrator. Here, the optimal order is the 2nd order, which is the only order that works, yielding low accuracy of only one significant figure, but a relatively short computational CPU time. Even the use of a tighter error tolerance in the ODE45 integrator or a finer spatial step of 0.3cm does not help to improve the accuracy; it only increases the computational time.

6.6 CONCLUSIONS

From the numerical results in section 6.5, the use of higher order (greater or equal to 4th) schemes had not been beneficial for solving a highly hyperbolic dominated PDE or steep moving front problems. For a parabolic-dominated PDE or less steep moving front problem, the use of a high order scheme is beneficial, there is a display of rapid convergence to high accuracy and an optimal order for the finite differencing. This optimal order scheme yields the best solution in term of accuracy CPU time. Better performances are obtained for even order finite difference approximations over that for odd orders. The results also show that the use of an odd-order scheme in MOL may result in an instability problem. This leads to the termination of the ODE integrator due to the minimum integration step not being able to satisfy the tolerance.

Moreover, it seems that the solutions of Richard's equation have the characteristics of a hyperbolic dominated PDE, even when their solution shows a gentle-moving front. Hence, the 2nd order schemes or lower order schemes should be used for the numerical solution of the Richard's equation. Additionally, a tighter error tolerances in the ODE45 integrator greater than 10^{-6} only improves the accuracy marginally but increases the CPU time. Hence a relatively smaller tolerance e.g. $tol = 10^{-6}$ in the ODE45 integrator is advocated for solving Richards' equation with steep-moving fronts.

Furthermore, this chapter has shown that the use of higher order finite different schemes in the form of differentiation matrices in MOL is efficient and simple, and the demand of the storage requirement and the computational effort for higher order schemes compared to lower order schemes is minimal.

Varying Order FODM over 1-D Domain

If we knew what it was we were doing, it would not be called research, would it.

Albert Einstein

7.1 INTRODUCTION

Mesh-moving, static mesh-regeneration (r-refinement) and local mesh-refinement algorithms (h-refinement) have been popular for tackling time dependent PDEs having large solution variations, such as steep moving wetting fronts, boundary layers or contact surfaces. In these methods, significant gain in accuracy and efficiency are obtained by adapting the nodes so that they are concentrated about the areas of large variation [Oden and Demkowicz, 1988].

While the h-refinement refines or coarsens the mesh locally based on spatial activity, the p-refinement increases or decreases the order of approximation in appropriate regions of the domain usually according to the smoothness of the solutions. Other researchers have shown that the proper combination of h- and p-refinement [Gui and Babuska, 1986; Ainsworth and Senior, 1998] will achieve exponential rates of convergence in terms of the number of degrees of freedom, even if the true solution is non-smooth due to singularities arising from the geometry of the domain or from mixed boundary conditions. The main difficulties associated with these hp-refinements is that the adaptive strategy for choosing combinations of h- and p-refinements is not well understood.

In this chapter, a ‘p-refinement’ adaptive scheme is implemented where we adapt the different orders of the finite difference stencil of the first derivative approximation at these areas of large variation (the mesh remains fixed), instead of the conventional way of adapting the nodes. The variation of the order of finite difference approximation in the FODM is correlated with the profile of the error indicators, i.e. slope, curvature, second derivative of the solution, solution of the dependent variable. Basic error indicators derived from the Taylor Series approximation will also be investigated.

7.2 STAGE-WISE DIFFERENTIATION VERSUS DIRECT DIFFERENTIATION

In Chapter 4, the Taylor series expansion was used in the derivation of the FODM, which implies that the solution of the PDE or ODE satisfy the truncated Taylor series polynomial expansion of degree n . Hence a good error indicator would be

$$E_n = \mathbf{a}_i^n \frac{h^n}{n!} F^{(n)}(x_1) + \mathbf{a}_i^{n+1} \frac{h^{n+1}}{n+1!} F^{(n+1)}(x_1), \quad (7.1)$$

being the two leading remainder terms. If the higher derivative information is known at x_1 , then the higher derivative contributions in (7.1) provide a good error measure.

Higher order derivative approximations can be derived directly from the derivative matrices (as in chapter 4) or via the stage-wise differentiation method used in the MOL (as in chapter 4). It is not clear which of these methods of approximations are more appropriate for the contribution to a good error indicator. This section examines the differences between these two forms of derivative approximations and determines whether stage-wise or direct derivative approximation is to be used for the computation of the higher order derivative.

To investigate these differences, we applied the 1 to 12th order derivative approximation on the known nodes i.e. $x_1 = 1, x_2 = 1.01, \dots, x_{11} = 1.11, x_{13} = 1.12$ with its

12th order polynomial values at its respective nodes. The 12th order polynomial chosen is $x^{12} + x^{11} + \dots + x + 1$ (arbitrary). The twelfth order FODM was used for the stage-wise differentiation for higher order derivative approximation. Table 7.1 shows the differences between these two methods of derivative approximation.

Table 7.1 Results of comparison between stage-wise differentiation and direct derivatives approximation. The results in the column are MRE over the 13 nodes.

Order of Derivative	Direct Differentiation	Stage-wise Differentiation
1	$-2.9*10^{-14}$	$-2.9*10^{-14}$
2	$-2.3*10^{-12}$	$1.1*10^{-12}$
3	$1.1*10^{-10}$	$4.6*10^{-11}$
4	$5.4*10^{-9}$	$1.7*10^{-9}$
5	$1.2*10^{-7}$	$1.5*10^{-7}$
6	$-4.7*10^{-6}$	$1.6*10^{-6}$
7	$-1.5*10^{-4}$	$2.2*10^{-4}$
8	$4.2*10^{-3}$	$1.4*10^{-3}$
9	$1.8*10^{-2}$	$2.6*10^{-1}$
10	$2.5*10^0$	$1.4*10^0$
11	$4.3*10^0$	$3.7*10^2$

Based on the above results, both methods are equally bad. It can be seen that for lower order derivative approximation, both methods yield relatively good results. For higher order derivative approximation, the relative error can be as high as $3.7e*10^2$, which is about 400%, overestimated. In fact, (7.1) clearly shows that the higher order derivative approximations suffer large truncation error based on the Taylor approximation. Thus these two forms of derivative approximation will be limited to approximation of lower order derivatives. Hence the higher derivative contributions in (7.1) or the two leading remainder terms in the Taylor approximation will not be used as a contribution to an error indicator.

7.3 METHOD OF VARYING THE ORDER OF THE FODM

The idea of varying the order of the finite difference stencil for the first derivative approximation at areas of large variation, instead of the conventional way of

adapting the nodes, is a logical extension of the FODM. Recall in chapter 4 that the FODM is a matrix with each row representing a first derivative finite difference computational stencil for a node in the integration domain. This representation is unique for each discretized node, i.e. each row corresponds to a discretized node in the integration domain. . The corresponding rows of the different order of the FODM matrices to vary the order of the first derivative approximation for any node in the domain is swapped. This row-swapping in the FODM matrix results in a varying order FODM.

7.4 MANUAL VARYING ORDER FODM

A graphical user interface (GUI) was built into the PDE template (chapter 4) to allow the manual creation for the varying order of finite difference representation using the concept described above. This, of course, would add to the clarity of the analysis. The GUI module is inserted into the PDE template as shown in Fig. 7.2 and Fig. 7.1 shows the GUI.

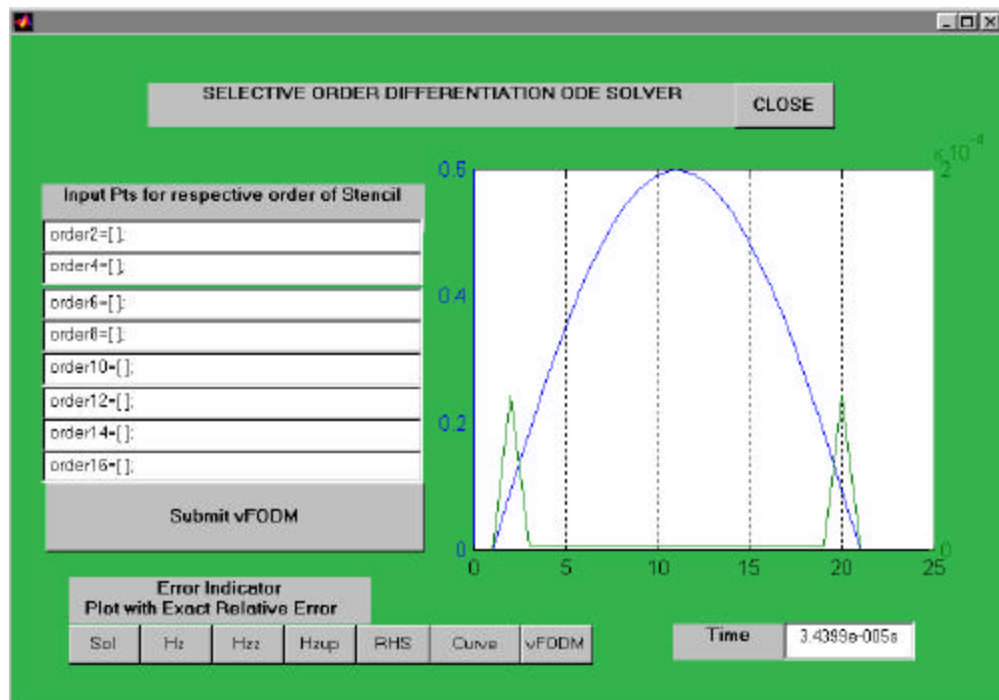


Fig. 7.1 The Graphical Interface (GUI)

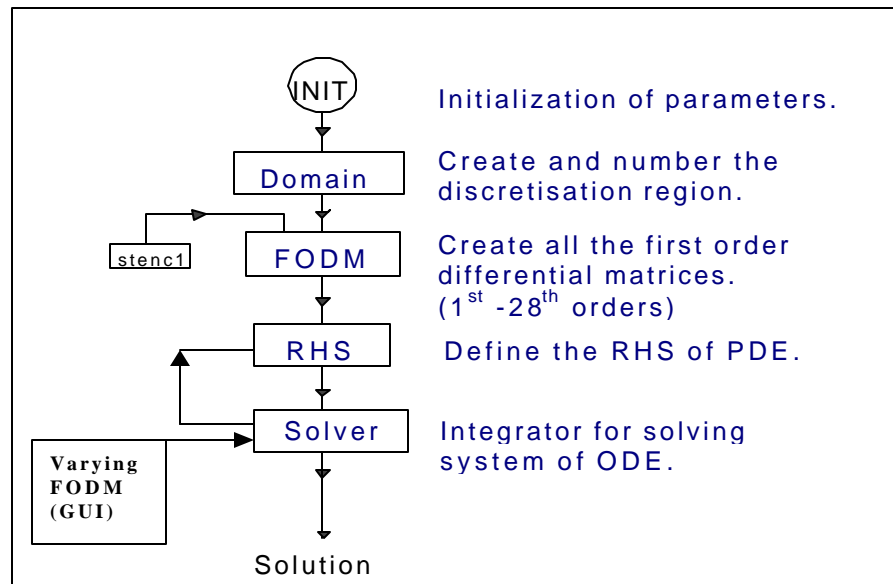


Fig. 7.2 The Template

To create a varying order FODM, the GUI is used to assign a different order to the finite different stencil for the first derivative approximation to the nodes in the integration domain. The GUI then automatically creates the varying order FODM based on this information. The creation of the varying order FODM occurs in the ODE solver at all successful steps of integration. With the aid of the GUI, at each successful step of integration, the profile of the error indicator, the profile of the exact error and the graphical presentation of the varying order FODM can be displayed on the computer screen. Based on this information, a criteria for varying the order of the computational stencils in the varying order FODM to create an optimal varying order FODM is determined.

7.5 NUMERICAL EXPERIMENTS

Our hypothesis is that a varying order FODM with the optimal selection of the order of the finite different stencil for the first derivative approximation at areas of large variation would improve the accuracy of the integration. An error indicator is used to capture these areas of large variation and the varying order FODM is selected based on

the profile of the error indicators. The error indicators include the profile of the solution, slope, second derivative of the solution, the curvature of the solution and the values of the right hand side of the PDE. The aim is to even out the profile of the error measure, which we hope will lead to an improvement in the overall solution.

For each successful integration step, a lower order scheme is used to approximate the derivative. Moreover, the Heat equation with analytical solution (see section (4.2)) is used for the numerical model in this investigation, hence a precise error measure can be obtained for each integration step. A higher order scheme is then used on those points (areas of large variation) that yield an error that is below the tolerance error values. If the errors are still above the error tolerance, the next higher order scheme is used. The process stops when the error specification is satisfied. Note that the graphs in the graphical user interface, see Fig. 7.1 shows the initial solution for the heat equation (4.2). The second curve with 2 peaks shows the exact relative error of the numerical solution.

Hence the order of the differencing scheme (for each point) is adjusted progressively from low to high order until the specified tolerance is achieved for the entire discretization domain. The size of the discretization domain used for each scheme must be identical so that a common reference is established. The overall aim is to produce an optimal combination of orders in the domain, thus giving a controlled numerical error. Hence, the objectives of the analysis of the varying FODM are

1. Test the hypothesis whether using a varying order FODM would improve the accuracy of the integration, as measured by the relative error.
2. Does an optimal selection of a varying order FODM based on the profile of an error indicator, i.e. the relative error, improve the accuracy of the integration?

In order to ascertain the accuracy of the integration with respect to the varying order FODM, the Heat equation (in section (4.2)) is used as a test problem that has an

exact analytical solution. This gives us an exact error measure. Furthermore, the property of the solution obtained from the various orders of a fixed FODM scheme is investigated before implementing the varying order FODM. This gives us a better understanding of the characteristics of the solution obtained by the ODE/MOL approach using FODM. For all cases, a tolerance of 10^{-16} was used in the MATLAB ODE45 solver routines. Tight error tolerances have been selected so that the spatial error dominates. A spatial step of $dz = 0.1$ with 21 nodes was also used for the integration to a model time of 0.1.

7.6 RESULTS AND DISCUSSION

7.6.1 Property of Fixed Order Finite Difference Scheme for the Heat equation.

Table 7.2 and Fig. 7.3 to Fig. 7.6 show the characteristics of the numerical solutions and the relative error for the heat equation, using various fixed order finite difference schemes, at model time of 0.1. In Fig. 7.3 to 7.6, the left hand scale relates to the Relative Error, and the right hand scale relates to the numerical solution. As shown in section (6.6), for a parabolic dominated PDE the use of high order scheme is beneficial, there is a display of increasing accuracy with higher orders and an optimal order for the finite differencing. Likewise, for the heat equation, an optimal order of the 12th order is obtained in Table 7.2. Fig. 7.2 to 7.4 show that an oscillation is evident in the solution for lower order finite difference scheme, i.e. order lower than the 8th order. However, this oscillation is progressively reduced with each higher order finite difference scheme; with the oscillation completely removed using the 16th order (see Fig. 7.6). However, a relatively large error in the solution still remains about the boundary points. This large error is 4 fold smaller (using the 12th order scheme) as compared to those obtained from the 2nd order scheme. This is possibly because the error is not uniform for the fixed order FODM, and is significantly lower about the boundary points due to the effect of the boundary conditions.

Table 7.2 Results for the heat equation at t = 0.1.

Order	MRE	Statistics			
		CPU (s)	Good Steps	Failed Attempts	Function Calls
2	8.1×10^{-3}	1.21	411	1	2 473
4	3.9×10^{-5}	0.49	220	1	1 327
6	1.8×10^{-7}	0.44	181	0	1 087
8	3.7×10^{-9}	0.33	167	0	1 003
10	3.1×10^{-11}	0.38	163	0	979
12	1.7×10^{-13}	0.38	165	0	991
14	3.8×10^{-13}	0.39	172	0	1 033
16	9.5×10^{-13}	0.60	225	6	1 387

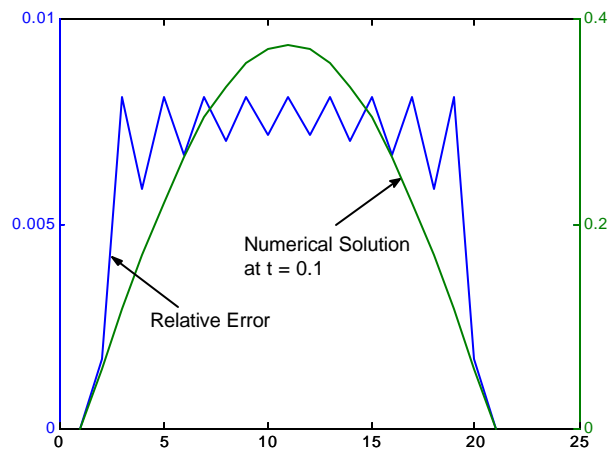


Fig. 7.3 Solution at t = 0.1 for 2nd order FD.

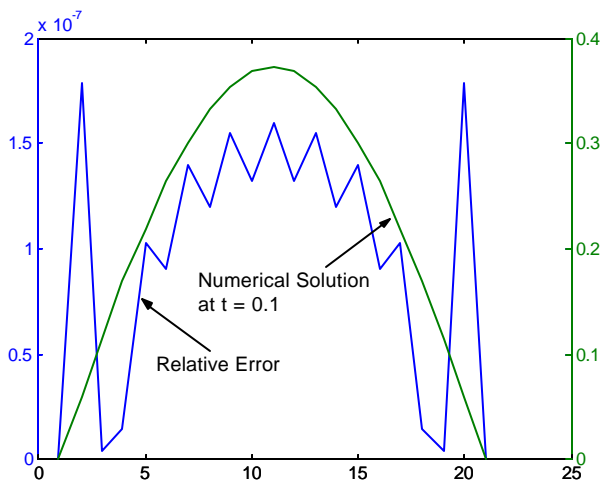


Fig. 7.4 Solution at t = 0.1 for 6th order FD.

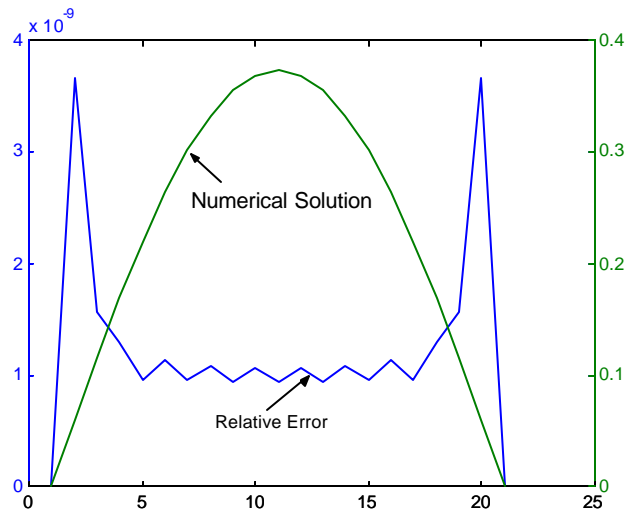


Fig. 7.5 Solution at $t = 0.1$ for 8th order FD.

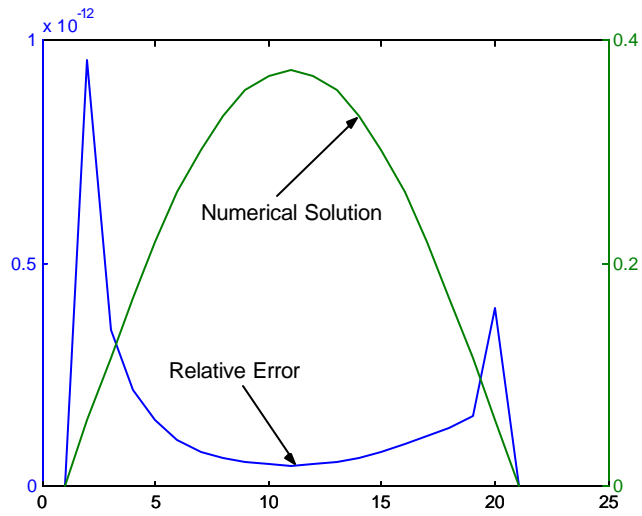


Fig. 7.6 Solution at $t = 0.1$ for 16th order FD.

7.6.2 Implementation of the Varying Order FODM.

Fig. 7.7 shows the GUI at the first successful time step of the ODE45 integrator. Our task is to achieve a controlled error of 3 significant figures with no oscillations for the solution of the heat equation. From section (7.6.1), the 2nd order scheme yields 3 significant figures accuracy with oscillation. Our expectation of 3 significant figures accuracy is reasonable, as the accuracy usually follows the lowest order or the weakest link in the numerical scheme.

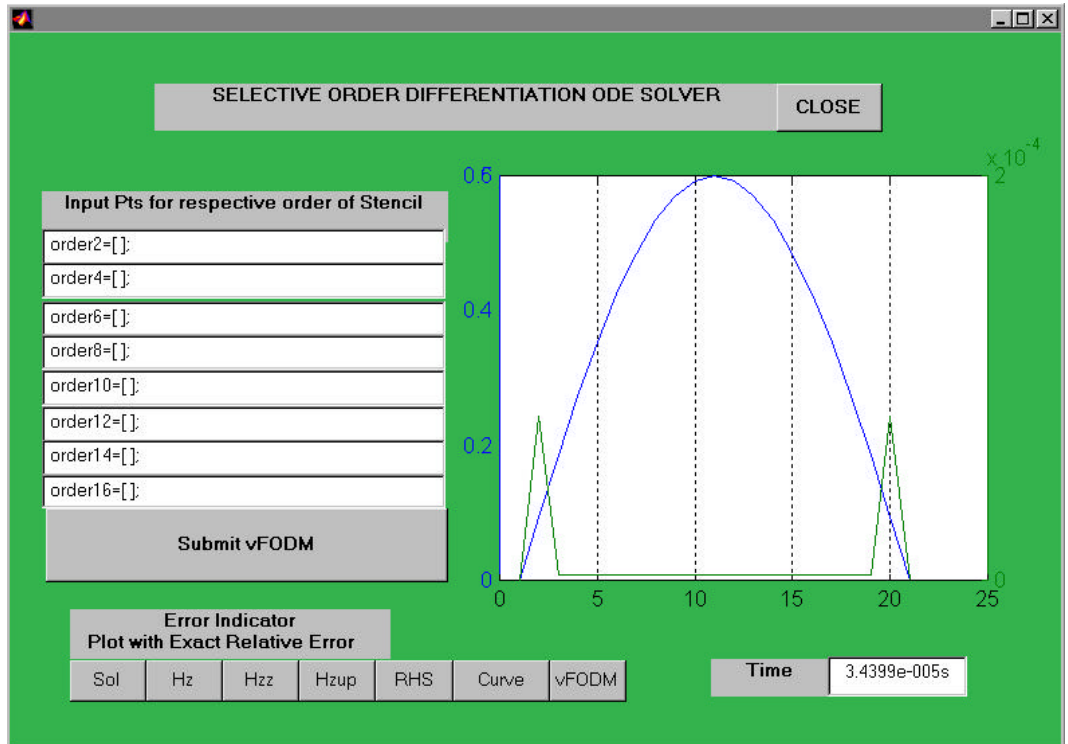


Fig. 7.7 GUI at the first successful step.

To determine an optimal-varying FODM, we use the GUI to explore the profile of the error indicators and the exact error. Figs. 7.8 to Fig. 7.10 show the results of these error profiles. The Relative Error is given by the left hand scale, and the Numerical Solution is given by the right hand scale. For this simple heat problem, it seems that the error is largely due to its boundary conditions at both ends of its domain. This suggests that a higher order scheme should be placed at the end boundaries. This can be implemented using the GUI in Fig. 7.7. The figure in the right (in Fig. 7.10) shows the placement of the 2nd order and 4th order stencil for the nodes in the domain using the GUI. Note that '5:17' in the 'Input points for the respective order stencil' button denotes the 5th to 17th nodes in the domain. These nodes are discretised using a 2nd order finite different. Once the 'submit vFODM' button in Fig. 7.10 is activated, a varying FODM will be created and this can be checked using the 'vFODM' button. Fig. 7.11 shows the result when 'vFODM' is activated. It shows that the 1st to 4th and 18th to

21st nodes are assigned the 4th order FD stencil, and the 5th to 17th nodes are assigned the 2nd order FD stencil.

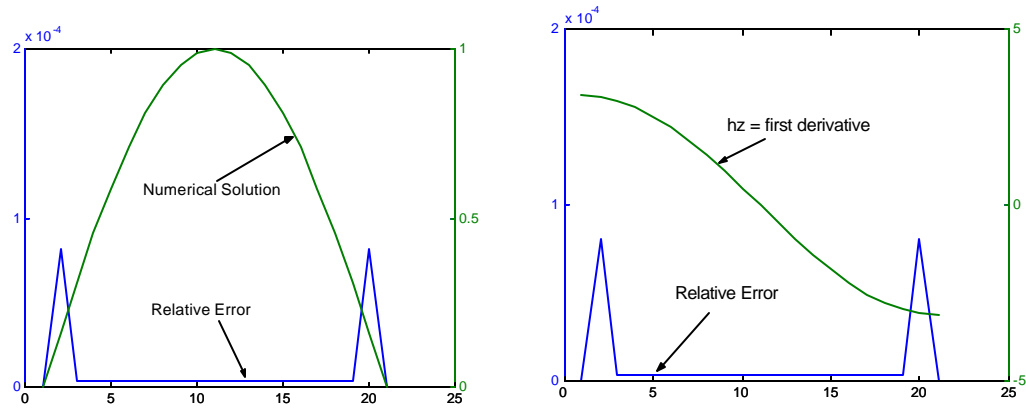


Fig. 7.8 Clicks of the ‘Soln’ and ‘hz’ button at the first successful step.

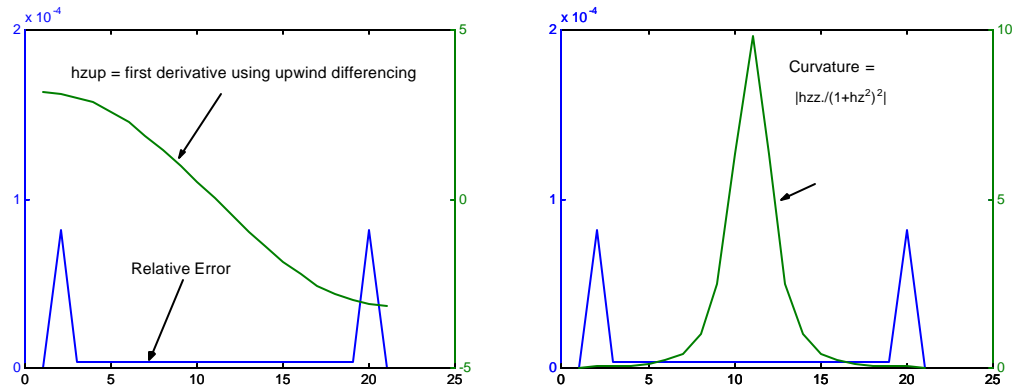


Fig. 7.9 Clicks of the ‘hzz’ and ‘Curve’ button at the first successful step.

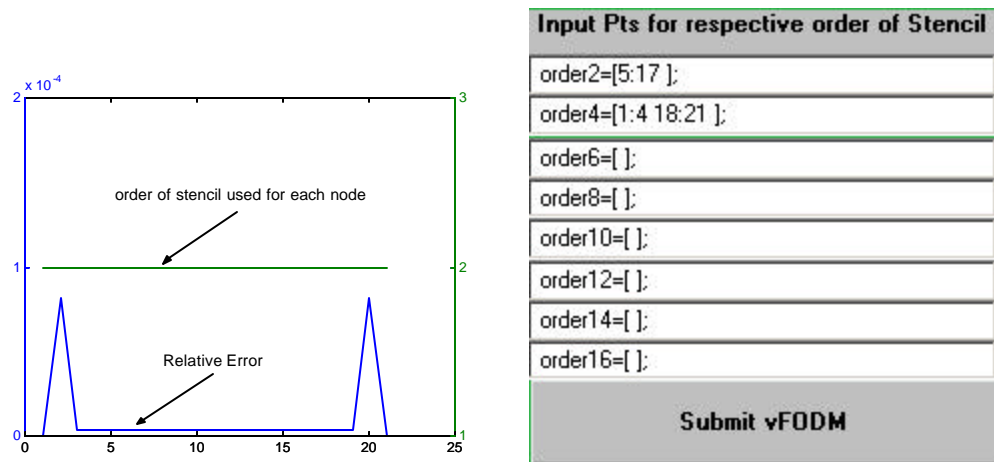


Fig. 7.10 Click of the ‘vFODM’ button and the assigned stencil for each node at the first successful step.

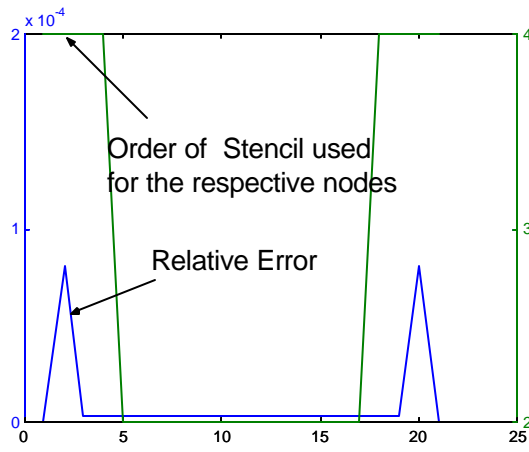


Fig. 7.11 Click of the 'vFODM' button.

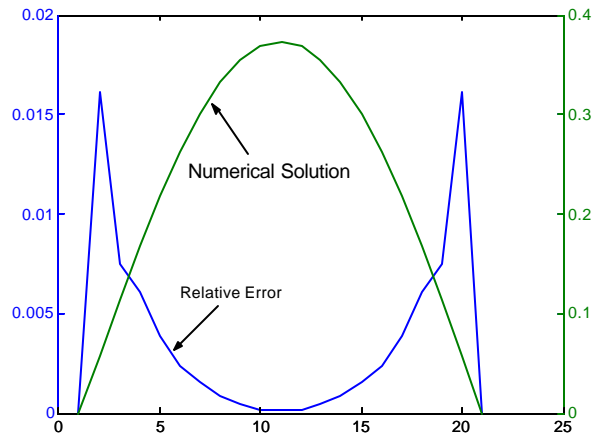


Fig. 7.12 Solution at the final successful step.

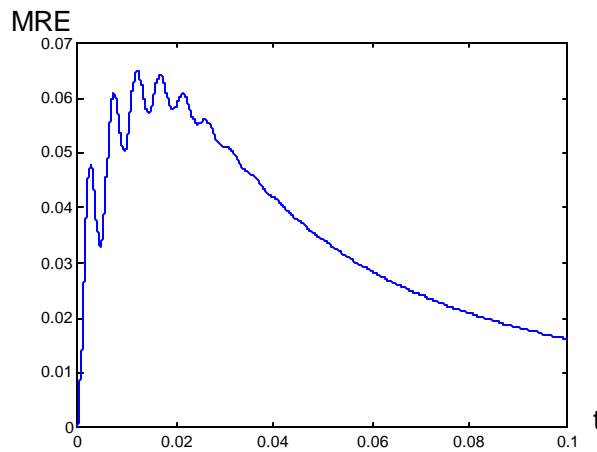


Fig. 7.13 Plot of MRE at all final successful step against time for the heat Equation using varying FODM.

Table 7.3 Results for the heat equation using varying FODM at t = 0.1.

Order	MRE	Statistics			
		CPU (s)	Good Steps	Failed Attempts	Function Calls
Varying	1.6×10^{-2}	178	3592	32	21 745

Following the procedure described in section (7.5), the results for the heat equation at model time of 0.1 are shown in Fig. 12, Fig. 13 and Table 7.3. The oscillations in Fig. 7.13 arise at early times in the solution. The result shows that the varying order FODM would not improve accuracy and no optimal varying FODM were found, as there were no correlation between the error indicators and the varying FODM. However, it has eradicated the oscillation in the solution.

7.6.3 Controlled Numerical Error with No Oscillation.

Based on the results obtained in section (7.6.2), it seems that in a numerical scheme with a mixture of orders, the weakest link or lowest order dominates the solution at the end of the integration. Additionally, the higher order scheme can remove the oscillation from the solution at the expense of a reduction of 1 significant figure in its accuracy. To exercise this non-oscillating property inherent in a higher order scheme, we implement a high order scheme for all nodes immediately after the successful step of a low order scheme. This implementation resulted in a controlled numerical error with no oscillation and a constant MRE for all nodes. Fig. 7.14 and Fig. 7.15 shows the results of the heat equation based on this scheme. Note that ‘2 → 6’ means using the 6th order scheme for all nodes immediately after the successful step of the 2nd order scheme applied on all nodes. And this 6th order scheme is then used for the rest of the integration.

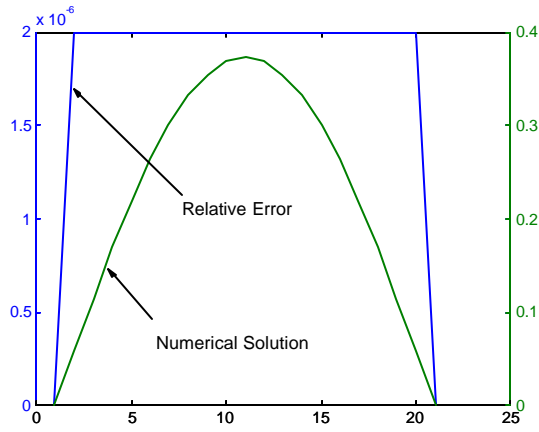


Fig. 7.14 Solution at the final successful step, 2 \rightarrow 6 FODM.

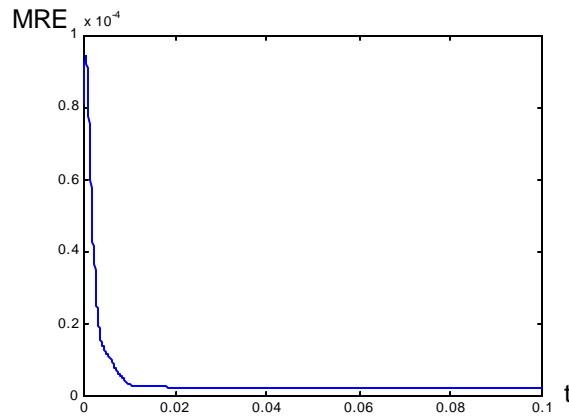


Fig. 7.15 Plot of MRE at all final successful step against time for the heat using a 2 \rightarrow 6 FODM.

Tables 7.4 to 7.8 show the results of the solution of the heat equation at a model time of 0.1, using the various orders of stencil with the above controlled numerical error with no oscillation procedure. A controlled numerical error or accuracy ranges from the 6th to 13th significant figures is obtained using the different order of finite difference. The initial order of the lower order scheme before the higher order scheme determines the accuracy of the scheme. Note that the initial orders of the 12th and 14th order are not shown in the tables because they do not yield a controlled numerical error. Fig. 7.16 shows the plot of MRE at all final successful steps against time for the heat equation using a varying FODM 12 \rightarrow 14, which depicts an oscillating MRE.

Table 7.4 Results for the heat equation using varying FODM at t = 0.1.

Order	MRE * 10 ⁻⁶	Statistics			
		CPU (s)	Good Steps	Failed Attempts	Function Calls
2 → 6	2.3	26	276	21	1 783
2 → 8	2.1	17	288	21	1 855
2 → 10	2.1	16	307	21	1 969
2 → 12	2.0	15	334	22	2 137
2 → 14	2.0	16	373	21	2 365
2 → 16	1.9	18	449	25	2 845

Table 7.5 Results for the heat equation using varying FODM at t = 0.1.

Order	MRE * 10 ⁻⁸	Statistics			
		CPU (s)	Good Steps	Failed Attempts	Function Calls
4 → 6	19	21	208	14	1 333
4 → 8	4.2	12	219	15	1 405
4 → 10	4.7	12	237	15	1 513
4 → 12	6.1	12	259	15	1 645
4 → 14	7.7	14	288	15	1 819
4 → 16	9.9	15	366	18	2 305

Table 7.6 Results for the heat equation using varying FODM at t = 0.1.

Order	MRE * 10 ⁻⁹	Statistics			
		CPU (s)	Good Steps	Failed Attempts	Function Calls
6 → 8	3.5	13	184	10	1 165
6 → 10	0.67	12	193	11	1 225
6 → 12	1.5	12	205	11	1 297
6 → 14	2.7	12	223	11	1 405
6 → 16	4.7	14	280	15	1 771

Table 7.7 Results for the heat equation using varying FODM at t = 0.1.

Order	MRE * 10 ⁻¹¹	Statistics			
		CPU (s)	Good Steps	Failed Attempts	Function Calls
8 → 10	2.3	11	170	6	1 057
8 → 12	0.81	11	175	6	1 087
8 → 14	4.7	11	186	6	1 153
8 → 16	9.0	12	232	10	1 453

Table 7.8 Results for the heat equation using varying FODM at $t = 0.1$.

Order	MRE $\times 10^{-13}$	Statistics			
		CPU (s)	Good Steps	Failed Attempts	Function Calls
10 \rightarrow 12	1.8	12	166	2	1 009
10 \rightarrow 14	5.1	11	174	2	1 057
10 \rightarrow 16	12	12	220	4	1 345

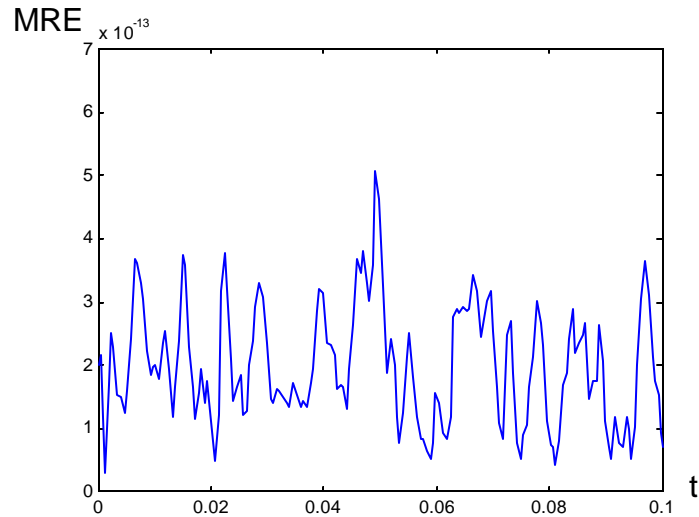


Fig. 7.16 Plot of MRE at all final successful step against time for the heat Equation using varying FODM 12 \rightarrow 14.

7.7 CONCLUSIONS

The results show that the varying order FODM does not improve accuracy and no optimal varying FODM can be found. The profile of the exact errors does not change significantly because the ad hoc adaptive strategy (of varying the FODM) does not improve the error control mechanism in the ODE solver. However, the use of a higher order FODM immediately after a lower order FODM scheme for all nodes managed to even out the error, which led to a controlled numerical error without oscillation.

The numerical experiments also show that the interactive visual computing (assisted by the GUI) is useful. Scientists can interpret changes in the data with respect to the solution in real time. Moreover, the GUI allows scientists to dynamically modify

computations while they are occurring. Note that the above solution has an exact solution so that errors can be estimated accurately. In practice, a 2-stage numerical solution is needed so that the errors can be estimated. The selection of the order needs to be automated as a priority.

Adaptive Scheme 1-D

If men liked shopping, they'd call it
research.

Cythina Nelms

8.1 INTRODUCTION

Adaptive methods for solving Partial Differential Equations (PDE) have become very popular [see Hawken et al., 1991; Baines, 1998; Chang and Haworth, 1997; Pardhanani and Carey, 2000; Wouwer et al., 2001]. The basic concept in of adaptive schemes is that an appropriate spatial distribution of grid points with respect to the evolving numerical solution plays a key role in determining the quality of the computed solution. The assumption is that the use of nonuniform grids that have higher local resolution in regions where the numerical error is large improves the accuracy of the numerical solutions of the PDE. Adaptive Schemes that use this concept include the Continuous Dynamic Grid Adaptation (CDGA) method, Moving Grid methods and the Local Refinement technique. [see Hawken et al., 1991; Baines, 1998; Chang and Haworth, 1997; Pardhanani and Carey, 2000; Wouwer et al., 2001].

However, these adaptive methods usually require tuning to ensure that the automatic choice of the changing space nodes is properly controlled. Baines (1998) showed that automatic grid selection based on some tuning parameter is intrinsically difficult, as they are rather problem-dependent and do not lend themselves to automation. Hence considerably more expertise, as compared to a fixed-grid algorithm, is needed to successfully implement these adaptive-grid techniques to achieve the best

possible results in terms of efficiency, robustness and reliability. There are very few, if any, robust moving-grid software packages available even for the relatively simple 1-D case.

In this chapter, we will investigate the CDGA advocated by Dietachmayer and Droegemeier [1992], and duplicate it using MATLAB and its built-in libraries. In addition, some modifications will be implemented so that the vectorized ODE/MOL approach can be used. The Burgers' model with an analytical solution (in section 3.3) is used to fully assess advantages and disadvantages of this approach. Another purpose for using the Burgers' model is to use it as an example to explain the mechanics of the CDGA scheme, which is described in section 8.2.

8.2 ADAPTIVE SCHEME 1-D (CDGA)

The CDGA scheme may be broken down into 3 distinct stages- transforming the physical governing equations, generation of the grid, and solving the coupled equations generated by the two preceding stages.

8.2.1 TRANSFORMING THE PHYSICAL GOVERNING EQUATIONS

The transformation from the physical to computational space in one dimension may be written as

$$\begin{aligned} z &= z(\mathbf{x}, t), \\ \mathbf{t} &= \mathbf{t}(t). \end{aligned} \tag{8.1}$$

By applying the chain rule for the partial derivatives,

$$\frac{\partial \Omega}{\partial \mathbf{x}} = \frac{\partial \Omega}{\partial z} \frac{\partial z}{\partial \mathbf{x}}, \tag{8.2}$$

$$\frac{\partial^2 \Omega}{\partial z^2} = \left(\frac{\partial \Omega}{\partial z} \right)_z = \frac{1}{z_x} \left(\frac{1}{z_x} \Omega_{xx} - \frac{z_{xx}}{z_x^2} \Omega_x \right), \tag{8.3}$$

$$\frac{\partial \Omega}{\partial t} = \frac{\partial \Omega}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial t} + \frac{\partial \Omega}{\partial \mathbf{t}} \frac{\partial \mathbf{t}}{\partial t} = \frac{\partial \Omega}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial t} + \Omega_{\mathbf{t}}, \tag{8.4}$$

where $\Omega = \Omega(z, t)$, is a function of the two independent variables. Then the original partial differential equation can be transformed from the physical coordinates (z, t) to the computational coordinates (\mathbf{x}, \mathbf{t}) where $t = \mathbf{t}$. Thus, Burgers equation (see section 3.3) transforms to

$$\Omega_t = -\Omega_E \Omega_x + \mathbf{u} \left(\frac{1}{z_x^2} \Omega_{xx} - \frac{z_{xx}}{(z_x)^3} \Omega_x \right), \quad (8.5)$$

$$\text{where } \Omega_E = \left(\Omega - \frac{\partial z}{\partial \mathbf{t}} \right) \left(\frac{\partial z}{\partial \mathbf{x}} \right)^{-1},$$

is the effective convective velocity. The transformed boundary conditions are

$$t > 0, \quad z = 0, \quad \Omega_x = \Omega_z * z_x, \quad (8.6)$$

$$t > 0, \quad z = B, \quad \Omega_x = \Omega_z * z_x, \quad (8.7)$$

where B is the length of the domain. The transformed Burgers Equation and its boundary conditions are then solved on a uniform grid in the computational space.

8.2.2 GENERATION OF THE GRID

The grid generation determines the mapping that takes the grid points from the physical domain (z, t) to the computational domain (\mathbf{x}, \mathbf{t}) . Following Dietachmayer and Droegemeier [1992] the effective weight function is taken in the form

$$W(z, t) = W_1 \left| \frac{\partial \Omega}{\partial z} \right| + W_2 \left| \frac{\partial^2 \Omega}{\partial z^2} \right|, \quad (8.8)$$

where $W(z, t)$ is scaled within 0 to 1. W is a weighted average of the slope, $\frac{\partial \Omega}{\partial z}$, and

the curvature is represented by $\frac{\partial^2 \Omega}{\partial z^2}$, in the physical space. To construct the grids, they

used the variational approach of Brackbill and Saltzman [1982], and Thompson and Mastin [1985], i.e. seeking the solution of the variational problem,

$$\text{minimize } \int (1 + \mathbf{h}W^2) \left(\frac{\partial z}{\partial \mathbf{x}} \right)^2 d\mathbf{x}, \quad (8.9)$$

where \mathbf{h} is a user-specified constant that controls the degree of adaption, and where the maximum is taken over $z = z(\mathbf{x}, \mathbf{t})$.

The goal of the minimization of the integral (8.9) is to find the functional whose minimum is a transformation from \mathbf{x} to $z = z(\mathbf{x}, \mathbf{t})$ which depends on a given weight function $W > 0$. Note that $\frac{\partial z}{\partial \mathbf{x}}$ measures the derivative of the stretch of the distance between grid points, so that the minimum of (8.9) should result in a transformation where the distance between grid points varies smoothly. Hence, small values of \mathbf{h} yield very smooth grids with little response to the weight function, while larger values increase the impact of the weight function, but at the possible risk of introducing rapid changes in the grid spacing (and thus increased truncation error). The parameters W_1 and W_2 are user-defined constants, which for all computations are both set equal to one. This choice of value for W_1 and W_2 have been proven perfectly adequate in the results obtained by Dietachmayer and Droegemeier [1992]. This requirement will be discussed more fully in section (8.2.3.1)

Using the Calculus of Variation, the Euler Lagrange equation for (8.9) is found to be

$$(1 + \mathbf{h}W^2) \frac{\partial^2 z}{\partial \mathbf{x}^2} + \mathbf{h}W \frac{\partial W}{\partial \mathbf{x}} \frac{\partial z}{\partial \mathbf{x}} = 0, \quad (8.10)$$

or

$$\frac{\partial^2 z}{\partial \mathbf{x}^2} = \frac{-\mathbf{h}W \frac{\partial W}{\partial \mathbf{x}} \frac{\partial z}{\partial \mathbf{x}}}{(1 + \mathbf{h}W^2)}, \quad (8.11)$$

and its boundary conditions are

$$\begin{aligned}
t > 0, \quad \mathbf{x} = 0, \quad z = 0, \\
t > 0, \quad \mathbf{x} = BB - 1, \quad z = B.
\end{aligned}
\tag{8.12}$$

Note that (8.11) is the grid generator equation. The length of \mathbf{x} is arbitrary and is usually taken as a positive integer, except zero. In this chapter, BB is the number of discretized nodes in $0 \leq z \leq B$, and there is a ready correspondence between the physical and computational domains. Note that for $W_1 = W_2 = 0$, then $W = 0$ and (8.10) yields a linear relationship $z = a\mathbf{x} + b$, where a and b are integration constants. This represents a direct linear interpolation between the physical and computational spaces, and is the degenerative form from (8.10): this transform is independent of the properties of Ω .

The transformation in (8.10) reduces the high gradient regions in the physical domain, which are then carried over to the computational domain. Furthermore, standard (fixed-grid) finite difference techniques can be used to solve the transformed equation which results in conceptually simpler finite-difference schemes for the dependent physical variables.

8.2.3 SOLUTION OF EQUATIONS

The governing equations (8.5) and (8.11) are coupled in that (8.5) depends explicitly on z_x and z_{xx} . Further, (8.11) depends on W and hence on Ω_z , and Ω_{zz} , through (8.8). There are many ways of discretizing and solving the transformed governing equations and the coupled grid generator equations. In this chapter, the vectorized ODE/MOL approach is used to solve the transformed governing equation (8.5), and the shooting method is used for the coupled grid generator equation (8.11) [Lindfield and Penny, 1995]. To conduct these two processes, the weight function for the grid generator equation must first be created. The dependent physical variable in the

form of the initial conditions of the physical region is used to initialize the whole process.

8.2.3.1 Creation of the Weight function

The weight function (8.8) consists of 4 unknown variables, two of which are arbitrary parameter values, as mentioned in section (8.2.2), and two are derivative values of the solution variable Ω . Any form of derivative algorithm may be used to compute these two derivative functions. We choose to implement the piece-wise cubic spline interpolation on the dependent physical variable, to compute the derivatives required in (8.8). This method allows us to capture the finer details of the dependent physical variable to yield a better weight function approximation. Furthermore, several passes of a 1-2-1 filter also smooth the weight function before feeding it into the grid-generator (8.11), as recommended by Dietachmayer and Drogegier [1992]. They advocated that this smoothing would reduce the smearing effect of the weight function across several grid points to allow for any motion of significant features (for example shocks and fronts) over the course of the time step. Small high frequency wobbles in W may also arise from the differentiation. These can seriously affect the solution of (8.11), and hence the extensive use of smoothing. Finally, the weight function is scaled within 0 and 1. In Dietachmayer and Drogegier [1992], this scaling was advocated as an important part of the CDGA technique because it allows the user a simple method of controlling the degree of grid adaption via the parameter h and also provides a safeguard against solving problems that have discontinuous solutions. Without it, the appropriate values of h to achieve a given level of grid adaption can not be determined for the former case, and the weight function tends to infinity for the latter case.

8.2.3.2 Solving the Euler-Lagrange equation

The Euler-Lagrange equations associated with (8.9), along with its boundary conditions are given by (8.11) and (8.12) respectively. Note that (8.11) is a second order ODE that can be reduced to a system of two first order ODEs and solved using an ODE45 solver by implementing the shooting method. Details of the shooting method are available in many good numerical books, hence it will not be elaborated here. However note that the differentiation matrices in Lee et al. [1998] are applied to the weighting function and the physical grid system to compute the derivatives in the right hand side of (8.11): this is a further use of the template of section 4. With these values, solving the Euler-Lagrange equation yields a new transformed grid system, \mathbf{x} , based on the prior knowledge of the physical grid system, the weighting function and the dependent physical variable in each preceding time step. This computation of the new grid system is implemented for every successful time-step integration of the physical governing equations and at the start of the integration phase.

8.2.3.3 Solving the physical governing equations

Using the vectorized MOL and the PDE template in chapter 4, (8.5) is converted to a system of ODEs. An important detail in (8.5) is the evaluation of the time derivative of the transformed computational grid system $\partial z / \partial t$. This time derivative is obtained by the second order forward difference formula applied on the preceding values of z for each successful time-step. This implies that $\partial z / \partial t$ is maintained at a constant value for all trial functional calls by the ODE45 solver. The function that computes $\partial z / \partial t$ is shown in Table 8.1. Naturally $\partial z / \partial t$ will vary with successful time steps by the solver.

Table 8.1 $\partial z/\partial t$ function.

```

function status = timederv(t,y,flag)
% TIMEDERIV to compute dz/dt
global x X T ts dxdt Y z

if nargin < 3 | isempty(flag) % timederv(t,y)
    yy = y(:,size(y,2));      % pick the latest value in t, y.
    tt = t(size(t,2),1);

    [wt wtprime wtprimeprime]= weightfn(x,yy'); %
    x = shooting1;
    X = [X(:,size(X,2)) x];   % to shorten X & T
    T = [T(:,size(T,2)) tt]; % save memory space
    Y = [Y(:,size(Y,2)) yy];
    dxdt = (X(:,2)-X(:,1))./(T(:,size(X,2))-T(:,size(X,2)-1));
else
    switch(flag)
    case 'init' % timederv(tspan,y0,'init')
        T=ts;      % initialize the time derivative parameters at t=0.
        X=x;
        dxdt =0;
        Y=y;

    case 'done'      % timederv([],[],'done')
        fprintf('\n\n'); % dummy

    end
end
status = 0;

```

This “timederv.m” file which computes $\partial z/\partial t$ is specially designed for the ODE45 solver in MATLAB. With the following MATLAB commands,

```

Options = odeset('OutputFcn', 'timederv')
[t,y] = ode45('rhs',[starttime endtime], initial_y, Options)

```

the solver calls this function after each successful time step. The M-file is coded for the solver to call it with `timederv(tspan,y0,'init')` before beginning the integration so that the `timederv` function can be initialized. Then, the solver calls `status = timederv(t,y)` after each step, where the value of $\partial z/\partial t$ is computed. Note that the status output value of 1 instructs the integration to halt, and the value of 0, instructs the integration to proceed. When the integration is completed, the solver calls the output function with `timederv([],[],'done')`. The `timederv` has to be coded in this particular way so that it

interacts properly with the ODE solver. This ‘timederiv’ function file is important because modification of the original ODE45 files is complicated and not advisable.

With the above descriptions of the numerical implementation of the three major stages in the adaptive scheme, clearly the CDGA scheme is a complicated and tedious method. The scheme starts with the creation of the weight function, which is used by the grid generator to generate the new computational grid system, which in turn is utilized by the MOL to solve the physical governing equations for every successful time-step. However, most of the major numerical schemes used in the MATLAB CDGA, i.e. cubic spline interpolation, the smoothing filter and the ODE solver are built-in functions in MATLAB. Furthermore, the use of the vectorized MOL and the PDE template are similar to using built-in functions. They are user-friendly and versatile. Thus the above MATLAB CDGA setup benefits greatly from readily available robust software packages, templates, and algorithms.

8.3 NUMERICAL EXPERIMENTS

The Burgers’ Model (see section 3.2) is used in this chapter for the numerical models. The model was implemented in the CDGA scheme advocated by Dietachmayer and Droegemeier [1992]. The CDGA scheme was carried out as described in section 8.2- using the template approach, CDGA components and MATLAB’s built-in libraries. The models were first run for various values of the tunable parameter \mathbf{h} , W_1 and W_2 (for the CDGA component) to determine their optimal values. Then the optimal values were implemented in the Burgers’ model for various values of the tunable parameter, \mathbf{n} and the results are compared to those obtained via the ODE/MOL approach. The experiments were designed to test the performance of the CDGA scheme as compared to the ODE/MOL approach. All simulations were run to a model time of 0.5 and uses 21 discretised points and the 2nd order finite difference.

8.4 RESULTS AND DISCUSSION

Dietachmayer and Drogegémier [1992] showed that the CDGA method could suffer from the problem of an oscillating grid. This oscillation may be controlled, by a careful consideration of the finite-difference representation of the grid-generator equation, together with sufficient smoothing of the weight function and smoothing of the time-step of the time evolution.

In our MATLAB CDGA model, we have tried sufficient smoothing of the weight function and smoothing of the time-step of the time evolution, to help in reducing the oscillations in the numerical scheme. Moreover, we have tried different values of weighting W_1 and W_2 . Tables 8.2 and 8.3 show the effect of the values of weights W_1 and W_2 for $h=1$, which depicts a relatively uniform grid of 21 nodes. The case $W_1=W_2=0$ is not considered. There is a gain of one significant figure for the parabolic dominated BE, using $W_2=0$, or only the slope as the effective weight function. However, Table 8.3 shows that for the hyperbolic dominated BE, the effect of the values of the weighting W_1 and W_2 is marginal at best.

Tables 8.4 and 8.5 show the effect of the values of weighting W_1 and W_2 for $h=30$, where the grids are relatively clustered. The parabolic form still shows better accuracy for $W_2=0$, where the slope is the effective weight function. Again, the results show that the effect of W_1 and W_2 is marginal at best.

For convenience, the choice of the weighting, $W_1 = 1 = W_2$, has proven adequate in this cause, and is used in all subsequent numerical computations.

Table 8.2 Effect of W_1 and W_2 for $h=1$, $n=1$ and $tol=10^{-6}$.

		$W_1 = 0$	$W_1 = 1$	$W_1 = 2$
$W_2 = 0$	CPU (min)	N.A.	5.6	5.6
	MRE	N.A.	$1.8 \cdot 10^{-5}$	$1.8 \cdot 10^{-5}$
$W_2 = 1$	CPU (min)	7.1	7.5	7.6
	MRE	$2.7 \cdot 10^{-4}$	$2.9 \cdot 10^{-4}$	$2.2 \cdot 10^{-4}$
$W_2 = 2$	CPU (min)	6.9	6.2	7.5
	MRE	$2.7 \cdot 10^{-4}$	$2.9 \cdot 10^{-4}$	$2.9 \cdot 10^{-4}$

Table 8.3 Effect of W_1 and W_2 for $h=1$, $n=10^{-3}$ and $tol=10^{-6}$.

		$W_1 = 0$	$W_1 = 1$	$W_1 = 2$
$W_2 = 0$	CPU (min)	N.A.	3.0	2.9
	MRE	N.A.	0.9932	0.9932
$W_2 = 1$	CPU (min)	4.9	4.7	5.1
	MRE	0.9931	0.9921	0.9909
$W_2 = 2$	CPU (min)	4.9	4.6	5.0
	MRE	0.9931	0.9926	0.9921

Table 8.4 Effect of W_1 and W_2 for $h=30$, $n=1$ and $tol=10^{-6}$.

		$W_1 = 0$	$W_1 = 1$	$W_1 = 2$
$W_2 = 0$	CPU (min)	N.A.	23.9	24.0
	MRE	N.A.	$4.7 \cdot 10^{-4}$	$4.7 \cdot 10^{-4}$
$W_2 = 1$	CPU (min)	30.6	13.0	11.6
	MRE	$1.5 \cdot 10^{-3}$	$2 \cdot 10^{-3}$	$2.1 \cdot 10^{-3}$
$W_2 = 2$	CPU (min)	30.6	15.8	12.9
	MRE	$1.5 \cdot 10^{-3}$	$1.9 \cdot 10^{-3}$	$2.0 \cdot 10^{-3}$

Table 8.5 Effect of W_1 and W_2 for $h=30$, $n=10^{-3}$ and $tol=10^{-6}$.

		$W_1 = 0$	$W_1 = 1$	$W_1 = 2$
$W_2 = 0$	CPU (min)	N.A.	5.3	5.4
	MRE	N.A.	0.9954	0.9954
$W_2 = 1$	CPU (min)	10.2	10.9	10.6
	MRE	0.9856	0.9683	0.9555
$W_2 = 2$	CPU (min)	10.3	11.0	10.9
	MRE	0.9856	0.9783	0.9683

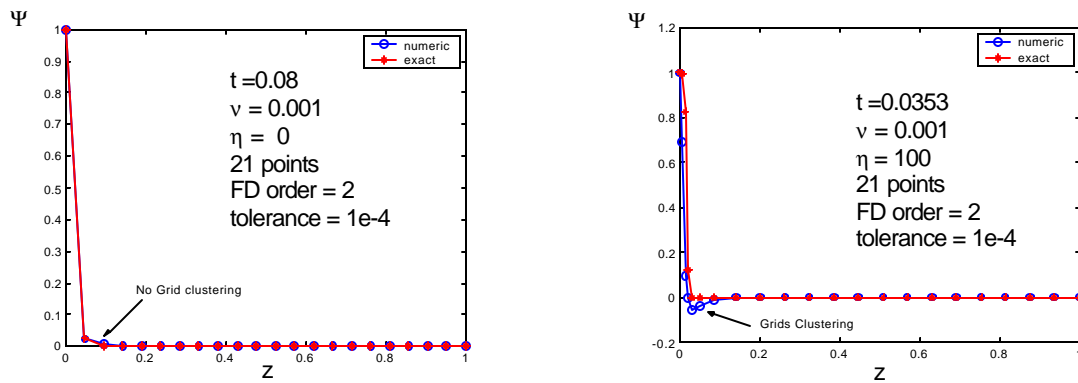


Fig. 8.1 Grid clustering in z with 21 points, for various h values.

Fig. 8.1 shows that the h value controls the distribution of the grid points in the system, a higher h value generates a greater clustering of grid points, and a value of $h=0$ creates a uniform grid system. Fig. 8.2 shows a system using $h=21$ with 21 discretization points and note that oscillations developed at $t = 0.1069$. Fig. 8.3 shows a better solution, using $h=0$, with 21 grid points, and $tol = 10^{-4}$. However, with a coarse grid, the numerical solution lags behinds the exact solution. Using more grid points remedies this ‘lag’ problem, those used in Tables 8.4 to 8.7. The MRE results for $n = 1$, show that clustering of the points produces less accurate answers based on the MRE values. The CPU times also increase as h is increased. For the hyperbolic case $n = 10^{-3}$ the accuracy is about the same, although h is increased significantly. The CPU time doubles in these examples.

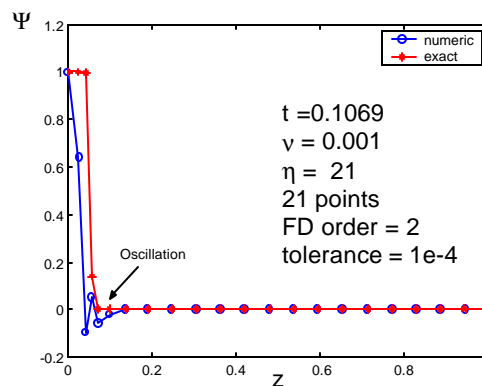


Fig. 8.2 Oscillations developed in system with $h=21$ and 21 points.

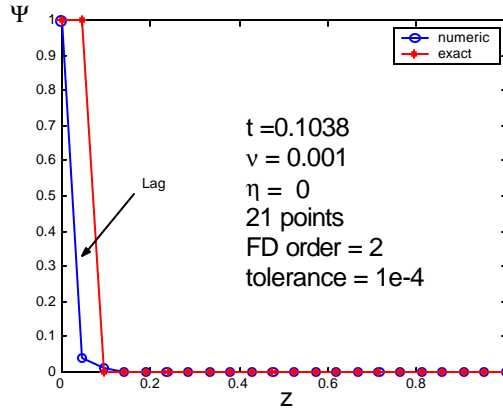


Fig. 8.3 Solution Profile for $h=0$.

Using the values for $W_1 = W_2 = 1$, numerical experiments were conducted to look at the effects of h and the tolerance in the ODE solver.

The results are displayed in Tables 8.6 and 8.7. For the parabolic case $n=1$, Table 8.6 indicates an optimal value of $h=0$. The errors for the $h=0$, case are two significant figures less, i.e. better, than the other cases of $h \geq 10$, for $tol = 10^{-6}$. The corresponding values for $n = 10^{-3}$, $h=1$ (Table 8.3) and $h=30$ (Table 8.4) also support and fit into the pattern in Table 8.6. Moreover, a higher value of h causes a reduction of 1 significant figure for the parabolic dominated BE and the CPU time is 4 to 6 fold longer. However, the magnitude of the tol value does not have much effect on the accuracy for the parabolic case.

This was rather surprising as Dietachmayer and Drogegier [1992] showed that with $h=0$, there are large oscillations in the dependent variable behind the shock. They noted that this is a well-documented feature of standard finite-difference techniques. Our model does not have any or small oscillations for $h=0$. It is likely that the reduction of the gradient does not affect the PDE system in an iterative setup as demonstrated by as Dietachmayer and Drogegier [1992]. A larger value of h yields

a slight improvement in the accuracy for the hyperbolic dominated BE, but at the expense of the CPU time that is 5 to 11 fold longer. Furthermore, a higher *tol* value does not improve the accuracy of the solution, while there is 2 to 3 fold increase in the CPU.

Table 8.6 Effect of h and tol for $n = 10^{-3}$.

		$h=0$	$h=10$	$h=50$	$h=100$	$h=150$	$h=200$
$tol=10^{-3}$	CPU (s)	211	730	882	1039	1052	1137
	MAE	2.4e-4	1.4e-3	2.3e-3	2.7e-3	2.8e-3	3.0e-3
$tol=10^{-6}$	CPU (s)	213	731	874	989	1067	1156
	MAE	1.1e-5	1.4e-3	2.3e-3	2.7e-3	2.8e-3	3.0e-3

Table 8.7 Effect of h and tol for $n = 10^{-3}$.

		$h=0$	$h=10$	$h=50$	$h=100$	$h=150$	$h=200$
$tol=10^{-3}$	CPU (s)	38	164	280	326	436	473
	MAE	0.9951	0.9683	0.9888	0.9880	0.9788	0.9667
$tol=10^{-6}$	CPU (s)	102	525	600	777	893	906
	MAE	0.9952	0.9683	0.9888	0.9880	0.9788	0.9665

The optimal value ($h=0$), with $W_1 = W_2 = 1$, was implemented in the CDGA scheme for the Burgers' model for various values of the tunable parameter, n and the results are compared to those obtained via the ODE/MOL approach on a fixed grid. Tables 8.8 to 8.12 shows the results of this comparison. Note that in the tables, there are two distinct groups of result. The left group is the results obtained via the CDGA scheme using $h=0$, and the right group is the results obtained via the vectorized ODE/MOL on a fixed grid. 'Failure' in the table means that the solver is unable to meet the integration tolerances without reducing the step size below the smallest value allowed (8.830442×10^{-17}) at time t . The 'order' signifies the specific order of the finite difference scheme used in the differentiation matrix.

Comparing the results for the two methods, it can be seen that the CDGA or the uniform grid-stretching scheme yields slightly better results than the vectorized

ODE/MOL approach, for the full range of n values. Note the failure of the integrator to use higher order schemes for $n = 10^{-3}$.

Table 8.8 $n = 1$, where integration is from 0 to 0.5 second for a spatial step of 1/20.

Order	CDGA, $h=0, tol=10^{-10}$					ODE/MOL, $tol=10^{-10}$				
	MRE	CPU Time (s)	Good steps	Failed steps	Derivative calls	MRE	CPU Time (s)	Good steps	Failed steps	Derivative calls
2	$1.1*10^{-5}$	162	141	3	865	$1.2*10^{-5}$	0.4	134	3	823
4	$1.6*10^{-9}$	214	186	2	1 129	$1.9*10^{-9}$	0.5	175	2	1063
6	$4.7*10^{-11}$	199	173	44	1 303	$5.1*10^{-12}$	0.6	161	19	1081
8	$3.6*10^{-12}$	246	211	75	1 717	$2.2*10^{-10}$	0.8	192	79	1627
10	$6.7*10^{-12}$	262	227	45	1 633	$8.7*10^{-11}$	0.7	209	49	1549
12	$1.9*10^{-11}$	284	245	18	1 579	$1.7*10^{-11}$	0.7	225	3	1369
14	$7.9*10^{-12}$	298	257	36	1 759	$6.4*10^{-11}$	0.7	233	3	1417
16	$3.1*10^{-11}$	322	278	53	1 987	$3.4*10^{-12}$	0.9	256	60	1897

Table 8.9 $n = 2$, where integration is from 0 to 0.5 second for a spatial step of 1/20.

Order	CDGA, $h=0, tol=10^{-10}$					ODE/MOL, $tol=10^{-10}$				
	MRE	CPU Time (s)	Good steps	Failed steps	Derivative calls	MRE	CPU Time (s)	Good steps	Failed steps	Derivative calls
2	$1.5*10^{-6}$	220	191	4	1 171	$1.7*10^{-6}$	0.6	176	4	1081
4	$6.6*10^{-11}$	305	262	9	1 627	$1.5*10^{-10}$	0.7	242	3	1471
6	$1.6*10^{-10}$	304	316	20	2 017	$3.5*10^{-11}$	0.9	285	55	2041
8	$4.4*10^{-11}$	438	378	95	2 839	$2.9*10^{-11}$	1.2	343	67	2461
10	$1.6*10^{-11}$	508	426	119	3 271	$5.2*10^{-12}$	1.3	386	85	2827
12	$9.2*10^{-12}$	527	454	74	3 169	$2.1*10^{-11}$	1.4	413	68	2887
14	$1.1*10^{-11}$	566	486	145	3 787	$3.7*10^{-12}$	1.7	441	129	3421
16	$3.9*10^{-11}$	637	547	128	4 051	$1.7*10^{-11}$	1.9	495	121	3697

Table 8.10 $n = 0.1$, where integration is from 0 to 0.5 second for a spatial step of 1/20.

Order	CDGA, $h=0, tol=10^{-6}$					ODE/MOL, $tol=10^{-6}$				
	MRE	CPU Time (s)	Good steps	Failed steps	Derivative calls	MRE	CPU Time (s)	Good steps	Failed steps	Derivative calls
2	$1.5*10^{-2}$	24	21	1	133	$1.7*10^{-2}$	0.3	107	1	649
4	$2.2*10^{-4}$	29	25	1	157	$2.8*10^{-4}$	0.4	132	1	799
6	$1.4*10^{-5}$	27	24	1	151	$2.6*10^{-5}$	0.4	137	1	829
8	$7.5*10^{-6}$	29	24	3	163	$4.9*10^{-5}$	0.4	131	1	793
10	$2.5*10^{-5}$	32	28	2	181	$4.7*10^{-5}$	0.4	114	1	691
12	$1.9*10^{-5}$	36	31	1	193	$1.4*10^{-5}$	0.3	101	1	613
14	$1.6*10^{-5}$	36	31	1	193	$4.9*10^{-5}$	0.4	124	1	751
16	$3.7*10^{-5}$	37	32	5	223	$7.9*10^{-5}$	0.5	138	1	835

Table 8.11 $n = 0.01$, where integration is from 0 to 0.5 second for a spatial step of 1/200.

Order	CDGA, $h=0, tol=10^{-6}$					ODE/MOL, $tol=10^{-6}$				
	MAE	CPU Time (s)	Good steps	Failed steps	Derivative calls	MAE	CPU Time (s)	Good steps	Failed steps	Derivative calls
2	$1.7*10^{-2}$	211	138	15	919	$1.7*10^{-2}$	0.6	137	15	913
4	$1.1*10^{-4}$	101	162	13	162	$1.1*10^{-4}$	0.7	161	10	1027
6	$1.3*10^{-6}$	284	182	28	1 261	$1.3*10^{-6}$	1.0	179	27	1237
8	$2.0*10^{-6}$	316	201	43	1 465	$2.1*10^{-6}$	1.1	198	38	1417
10	$4.6*10^{-7}$	355	225	29	1 525	$4.6*10^{-7}$	3.7	222	26	1489
12	$3.7*10^{-7}$	398	248	55	1 819	$2.8*10^{-7}$	1.8	245	57	1813
14	$1.7*10^{-7}$	449	278	58	2 017	$1.9*10^{-7}$	1.9	275	64	2035
16	$1.0*10^{-7}$	508	313	46	2 155	$1.7*10^{-7}$	2.3	312	42	2125

Table 8.12 $n = 0.001$ where integration is from 0 to 0.5 second for a spatial step of 1/200.

Order	CDGA, $h=0, tol=10^6$					ODE/MOL, $tol=10^6$				
	MAE	CPU Time (s)	Good steps	Failed steps	Derivative calls	MAE	CPU Time (s)	Good steps	Failed steps	Derivative calls
2	$9.9*10^{-1}$	466	307	2	1 855	$9.9*10^{-1}$	1.2	306	2	1849
4	$3.5*10^{-1}$	590	383	1	2 305	$4.3*10^{-1}$	1.5	381	1	2293
6 and above	Failure at $t=2*10^{-2}$					Failure at $t=2*10^{-1}$				

For a uniform grid-stretching scheme, the transformation parameters $z_x = dz$ (spatial step) and $z_{xx} = 0$, which simplify (8.5) to

$$\Omega_t = -\frac{\Omega}{z_x} \Omega_x + u \left(\frac{1}{z_x^2} \Omega_{xx} \right). \quad (8.13)$$

Table 8.13 shows the results of the uniform grid-stretching scheme for $n = 0.001$. Using this uniform grid-stretching scheme without the grid generation yields similar results to those obtained from the CDGA scheme. Note that the CPU time is drastically reduced, as CPU time is not used for the generation of grids.

Table 8.13 $n = 0.001$ where integration is from 0 to 0.5 second for a spatial step of 1/200.

Order	Uniform Grid Stretch, $tol=10^6$					CDGA, $h=0, tol=10^6$				
	MAE	CPU Time (s)	Good steps	Failed steps	Derivative calls	MAE	CPU Time (s)	Good steps	Failed steps	Derivative calls
2	1.0	1	306	0	1 849	$9.9*10^{-1}$	466	307	2	1 855
4	$4.7*10^{-1}$	2	381	1	2 293	$3.5*10^{-1}$	590	383	1	2 305
6 and above	Failure at $t=2.*10^{-2}$					Failure at $t=2*10^{-2}$				

8.3 CONCLUSIONS

The CDGA case study has clearly demonstrated that the assumption that “the use of nonuniform grids that have higher local resolution in regions where the numerical error is large, improves the accuracy of the numerical solutions of a PDE” must be used with caution. The efficiency of the overall grids, interactions and effects of various factors such as slope discontinuities, singularities, non orthogonalities and

skewness of the grid controlled by the optimal tuning parameters in the grid generator, is hard to determine. Furthermore, the monitoring function for the grid distribution is somewhat arbitrary and can involve arc-length, curvature, arc-curvature (optimal combination of both), smoothing, and regularization. In situations where analytical solutions are not available, the validity of the solution may be questionable. Another drawback of this method is that the CPU requirement is many orders (100-500) of magnitude larger than that required in a non-adaptive scheme. However, in this case study, CDGA yields slightly better results than the vectorized ODE/MOL approach but at the expense of the CPU time.

One source of difficulty may be the need to differentiate the solution of (8.5) numerically to obtain the coefficients in (8.10). Large amounts of smoothing were used by Dietachmayer and Droegemeies [1992] in an attempt to control the sensitivity of the solution of (8.10). Certainly the results in this chapter suggest that much more research work is needed before the CDGA technique can deliver the potential that others have seen.

Non-uniform Grid Stretching

If an elderly but distinguished scientist says that something is possible he is almost certainly right, but if he says that it is impossible he is very probably wrong.

Arthur C. Clarke

9.1 INTRODUCTION

The Grid Stretching method has regained its popularity, especially in the field of meteorology [Leslie et al., 1989; Dietachmayer and Droegemeier, 1992; Ly and Luong, 1999; Rabinovitz et al, 2000]. Grid stretching methods have their origins in astrophysics and their implementation has also gained popularity in other fields such as science and engineering [Bao et al., 2000; Mavriplis, 1998; Oosterlee, 1997; Arneborg and Hansen, 1998]. The introduction of a nonlinear grid transformation function that maps nonuniform lattice point coordinates into orthogonal, uniform computational nodes has greatly contributed to its popularity [Anthes, 1970]. This grid-stretching mapping provides increased spatial resolution in selected regions of the computational domain and at the same time reduces the high gradient regions in the physical domain. A weakness of this method is that one must know *a priori*, and for the duration of the calculation, which regions of the domain will require high spatial resolution [Rabinovitz et al, 2000]. The method has been widely applied in convective and mesoscale modelling studies and some good numerical solutions have been achieved [Wihelmsen and Chen, 1982].

In recent times, the use of non-uniform grid stretching to complement contemporary numerical techniques has been popular and good results have been

achieved. In Barakos et al. [1994], grid stretching was employed in the control volume method to guarantee accurate solutions, especially near the walls for high Rayleigh values. In Zhang et al. [1995], the technique was used to more accurately capture shock waves. Grid stretching has also been used in GCM (General Circulation Models). The stretched-grid GCM provides an efficient downscaling over the area of interest, that is, it properly simulates not only large-scale but also mesoscale features [Rabinovitz et al., 2000].

In this chapter, we investigate the use of the non-uniform grid stretching in conjunction with the vectorized ODE/MOL approach. Furthermore, the use of the exact solution for constant flux infiltration presented by Sander et al. [1988] enables rigorous testing of the computer-based numerical solutions to be carried out.

9.2 NON-UNIFORM GRID STRETCHING IN ODE/MOL

The non-uniform grid stretching is incorporated into the MOL by coordinate transformations which stretch out the region of high gradients so that, in the new coordinate the high gradient phenomenon is well resolved. One such transformation is Hoffmann and Chiang[1993]:

$$\begin{aligned} z &= Pt \left\{ 1 + \frac{\sinh[\mathbf{b}(x - A)]}{\sinh(\mathbf{b}A)} \right\}, \\ \mathbf{t} &= \mathbf{t}(t). \end{aligned} \quad (9.1)$$

where $A = \frac{1}{2\mathbf{b}} \ln \left[\frac{1 + (e^{\mathbf{b}} - 1)(Pt/H)}{1 + (e^{-\mathbf{b}} - 1)(Pt/H)} \right]$, \mathbf{b} is the clustering parameter, Pt is the point

of clustering in the z coordinate and H is the length of the z domain. The range of \mathbf{b} is from 0 to ∞ , with larger values of \mathbf{b} producing a denser clustering of grid points near Pt while no clustering occurs for $\mathbf{b} = 0$, i.e., a uniform grid.

By applying the chain rule for the partial derivatives,

$$\Psi_x = \Psi_z z_x, \quad (9.2)$$

$$J = z_x, \quad (9.3)$$

$$\Psi_{zz} = \frac{1}{J^2} \Psi_{xx} - \frac{z_{xx}}{J^3} \Psi_x, \quad (9.4)$$

the original partial differential equation is transformed from the physical coordinates (z,t) to the computational coordinates (\mathbf{x}, \mathbf{t}) where $t = \mathbf{t}$. Thus the Mixed Composed form of RE (5.10) becomes

$$\Theta_t = \frac{\left(K \frac{\Psi_x}{J} \right)_x}{J} - \frac{K_x}{J}, \quad (9.5)$$

and its transformed boundary conditions are

$$t = 0, \quad \mathbf{x} \geq 0, \quad \Theta(\mathbf{x}, \mathbf{t}) = \Theta(z, 0), \quad (9.6)$$

$$t > 0, \quad \mathbf{x} = 0, \quad \Psi_x = \left(1 - \frac{Q}{K} \right)^* J, \quad (9.7)$$

$$t > 0, \quad \mathbf{x} = 1, \quad \Psi_x = 1^* J, \quad (9.8)$$

where $\mathbf{x} = 1$, corresponds to $z = H$, and is the length of the transformed domain.

The transformed equation and its boundary conditions are then solved on a uniform grid in the computational space. In this particular grid stretching (9.1), only two tuning parameters, i.e. \mathbf{b} and Pt are involved, controlling the strength and location of the clustering in the physical domain respectively. Hence the tuning parameters are simple and sensitivity analyses of these parameters can easily be implemented. Furthermore, standard (fixed-grid) finite difference techniques can be used to solve the transformed equation which results in conceptually simpler finite-difference schemes.

Note that this non-uniform grid stretching method is similar to the first stage of the CDGA scheme described in the previous chapter. The difference lies in the manner in which the stretching is applied. In the CDGA, a further equation, the Euler-Lagrange, needs to be solved to minimize a variational. In this non-uniform stretching, the

stretching function is prescribed and two parameters are available to adjust the position and strength of stretching to the evolving solution.

9.3 NUMERICAL EXPERIMENTS

Model B (see section 3.3), with $D_0 = 2.75862$, is used for the numerical testing of the solution schemes. This model was solved using the nonadaptive stretching in (9.1), and solution runs were made for various values of Pt and \mathbf{b} . If an adaptive scheme were used, the clustering point, Pt , should follow the point of highest gradient in the domain, as our aim was to reduce the high gradient to a lesser one to yield more accurate results. However, there are many variables, i.e. K , Θ and Ψ , that can have a high gradient point with respect to the spatial dimension. It was found that these high gradient points, or values of Pt range from 0 to 6cm for the whole period of integration (not shown here). These limits corresponded to the short term, 0 to 0.3625 min, integration. And the Pt ranges from 0 to 35cm for the long term, 0 to 36.25 min, integration (not shown here). During the short-term integration, the front did not move very far from its initial position and the fixed stretching represented by (9.1) was adequate. However, during the long-term integration, the front has time to move significantly. Then different values of Pt may be used and the optimal choice depends on which value adequately represents the average.

Model B, with $D_0 = 2.75862$, was implemented in conjunction with (9.1) and solved using 2^d order difference representations. The solution runs were made for various values of the transform parameters \mathbf{b} and Pt . Solutions were also constructed on a fixed uniform grid in the physical space; i.e. by using the vectorized MOL discussed in section (4.2). For all cases, a tolerance of 10^{-6} was used in the MATLAB ODE45 solver routines. This relatively large error tolerance was used, as recommended in section (6.6). This value of the tolerance is sufficient for the spatial errors to dominate. Moreover, 501 nodes were used for all simulations.

9.4 RESULTS AND DISCUSSION

Table 9.1 shows the results for the short-term integration, $t = 0.3625$ minutes.

Table 9.1 Sensitivity Analysis of the Clustering Parameters, Pt and b at time = 0.3625 min.
The top left box is the results obtained using the vectorized ODE\MOL approach without grid stretching.

Vectorized MOL CPU = 0.99 s GME = 0.1030 % MRE = 6.1×10^{-4}		Clustering Strength, b						
		0.5	1	2	3	4	5	
Clustering point, Pt (cm from the origin)	10^{-10}	CPU	1.21	1.48	3.08	10.65	43.94	210.09
		GME	0.0964	0.0797	0.0412	0.0173	0.007	0.0032
		MRE	5.6×10^{-4}	4.5×10^{-4}	2.0×10^{-4}	7.3×10^{-5}	3.4×10^{-5}	2.9×10^{-5}
	10^{-5}	CPU	1.21	1.38	3.19	10.55	44.71	227.06
		GME	0.0964	0.0797	0.0412	0.0173	0.0071	0.0032
		MRE	5.6×10^{-4}	4.5×10^{-4}	2.0×10^{-4}	7.3×10^{-5}	3.4×10^{-5}	2.9×10^{-5}
	10^{-2}	CPU	1.21	1.43	3.02	10.66	47.12	239.7
		GME	0.0964	0.0798	0.0412	0.0174	0.0071	0.0032
		MRE	5.6×10^{-4}	4.5×10^{-4}	2.0×10^{-4}	7.3×10^{-5}	3.4×10^{-5}	2.9×10^{-5}
	10^{-1}	CPU	1.21	1.49	3.24	10.16	40.15	160
		GME	0.0964	0.0799	0.0415	0.0178	0.0075	0.0036
		MRE	5.6×10^{-4}	4.5×10^{-4}	2.0×10^{-4}	7.4×10^{-5}	3.5×10^{-5}	2.9×10^{-5}
1	CPU	1.20	1.48	3.13	7.75	18.95	37.24	
	GME	0.0968	0.0812	0.0449	0.0223	0.0123	0.0085	
	MRE	5.7×10^{-4}	4.6×10^{-4}	2.2×10^{-4}	9.2×10^{-5}	4.7×10^{-5}	3.4×10^{-5}	
2	CPU	1.26	1.53	2.86	6.20	11.32	43	
	GME	0.0972	0.0826	0.0487	0.0277	0.0189	0.0165	
	MRE	5.7×10^{-4}	4.7×10^{-4}	2.4×10^{-4}	1.2×10^{-4}	6.9×10^{-5}	5.8×10^{-5}	
4	CPU	1.20	1.48	2.47	3.90	5.17	22.69	
	GME	0.0980	0.0856	0.0568	0.0400	0.0355	0.0394	
	MRE	5.7×10^{-4}	4.9×10^{-4}	2.9×10^{-4}	1.7×10^{-4}	1.4×10^{-4}	1.4×10^{-4}	
5	CPU	1.26	1.43	2.41	3.24	10.6	16.15	
	GME	0.0985	0.0870	0.0610	0.0467	0.0452	0.0528	
	MRE	5.8×10^{-4}	5.0×10^{-4}	3.1×10^{-4}	2.1×10^{-4}	1.8×10^{-4}	2.0×10^{-4}	
6	CPU	1.26	1.48	2.25	3.02	9.18	25.87	
	GME	0.0989	0.0885	0.0653	0.0539	0.0555	16.286	
	MRE	5.8×10^{-4}	5.0×10^{-4}	3.4×10^{-4}	2.4×10^{-4}	2.3×10^{-4}	3.6×10^{-3}	
10	CPU	1.26	1.32	1.54	1.59	5.82	9.56	
	GME	0.1005	0.0946	0.0838	0.0860	0.1039	0.1348	
	MRE	5.9×10^{-4}	5.4×10^{-4}	4.4×10^{-4}	4.3×10^{-4}	5.0×10^{-4}	6.4×10^{-4}	

This table also depicts the sensitivity analysis of the clustering parameters, Pt and b for this short-term integration. Note that in each box of the numerical results, the first row contains the CPU time, the second row contains the GME and the last row contains the MRE in the domain. Furthermore, the top left box contains the results obtained using the vectorized ODE\MOL approach without grid stretching.

From Table 9.1, it can be seen that the optimal Pt value is in the range 10^{-2} to 10^{-10} . The errors also show some optimal values with respect to \mathbf{b} , at least for values of $Pt \geq 4$. Such behaviour will doubtless repeat itself for $Pt < 4$, when calculations are done for larger values of \mathbf{b} . This corresponds to strong clustering of the grid points near the emerging front at the left hand boundary. This is to be expected for the short integration time solutions, where the cluster point is moved further away from the emerging front. Then a smaller value of \mathbf{b} is needed to spread the grid away from Pt , in an attempt to provide some coverage of the emerging front near the left hand boundary.

Compared with the vectorized MOL, this stretching scheme is able to produce better accuracy (a gain of 1 significant figure) but at a CPU cost. The computational cost increases by a factor of 200 for a small increase of only one significant figure in MRE. However, the Global Mass Balance (GME) can be improved by a factor of 1/100 or by 2 significant figures; a far better improvement given the importance placed on mass balance in the literature (Celia et al., 1992).

Table 9. 2 shows the results for the long-term integration at model time, $t = 36.25$ minutes and also depicts the sensitivity analysis of the clustering parameters, Pt and \mathbf{b} for this long-term integration. The results did not show much improvement in MRE, CPU time or FME. The reason for this poor results might be due to the high gradient for the different variables (i.e. K , Θ and Ψ) moving in time. These high gradient regions are the regions requiring high spatial resolution. Thus an optimal Pt value is not able to compensate for the movement of the high gradient regions in time.

Table 9.2 Sensitivity Analysis of the clustering parameter, Pt at time = 36.25 min. The top left box is the results obtained using the vectorized ODE\MOL approach without grid stretching.

Vectorized MOL CPU = 12.9 min GME = 1.2×10^{-3} % MR E = 4.5×10^{-4}		Clustering Strength, b						
		0.5	1	2	3	4	5	
Clustering point, Pt (cm from the origin)	1e-10	CPU	26.8	43.2	85.3	370		
		GME	1.2×10^{-3}	9.8×10^{-4}	4.2×10^{-4}	5.7×10^{-5}		
		MRE	4.4×10^{-4}	3.8×10^{-4}	3.4×10^{-4}	4.7×10^{-4}		
	5	CPU	29.3	39.3	74.0	176	362	
		GME	1.2×10^{-3}	1.0×10^{-3}	5.4×10^{-4}	7.5×10^{-5}	3.3×10^{-4}	
		MRE	4.4×10^{-4}	3.8×10^{-4}	3.3×10^{-4}	4.2×10^{-4}	6.4×10^{-4}	
	15	CPU	30.3	39.1	59.0	84.8	110	
		GME	1.2×10^{-3}	1.1×10^{-3}	7.3×10^{-4}	3.9×10^{-4}	6.8×10^{-5}	
		MRE	4.4×10^{-4}	3.9×10^{-4}	3.1×10^{-4}	3.2×10^{-4}	4.4×10^{-4}	
	25	CPU	29.8	36.6	44.9	51.4	49.4	
		GME	1.2×10^{-3}	1.1×10^{-3}	9.4×10^{-4}	7.7×10^{-4}	6.3×10^{-4}	
		MRE	4.4×10^{-4}	4.0×10^{-4}	3.0×10^{-4}	2.7×10^{-4}	3.0×10^{-4}	
	35	CPU	30.8	32.4	34.7	45	28	46
		GME	1.2×10^{-3}	1.2×10^{-3}	1.2×10^{-3}	1.2×10^{-3}	1.4×10^{-3}	1.7×10^{-3}
		MRE	4.4×10^{-4}	4.0×10^{-4}	3.1×10^{-4}	3.1×10^{-4}	2.1×10^{-4}	2.2×10^{-4}
	45	CPU	30.0	31.1	31.0	23.0	16.0	11.4
		GME	1.2×10^{-3}	1.3×10^{-3}	1.4×10^{-3}	1.8×10^{-3}	2.5×10^{-3}	3.7×10^{-3}
		MRE	4.4×10^{-4}	4.1×10^{-4}	3.2×10^{-4}	2.4×10^{-4}	5.35×10^{-4}	1.7×10^{-4}

9.5 CONCLUSIONS

The implementation of non-uniform grid stretching in MOL is able to produce better accuracy for short-term integration as compared to the vectorized MOL, but at a CPU cost which about 10 to 300 times longer in duration. For long-term integration, the non-uniform grid stretching did not show much difference in results, as compared to the vectorized ODE\MOL approach. The reason is the single optimal tuning parameter, Pt , in the stretching scheme value is not able to compensate for the large variation of the optimal P values for each time step (if an adaptive scheme were used).

Most importantly, this chapter indicates that grid stretching in MOL is able to reduce the steep moving front to a lesser one that yields slightly more accurate numerical solution, provided that the variation of the highest gradient point does not vary too much. The stretching parameters used in this chapter is maintained at a constant value. However, a varying stretching parameters with respect to the change in

the steep moving front may produce better integration. Thus, this non-uniform grid stretching method in MOL deserves more investigation.

Vertical Scaling 1-D

I believe that a scientist looking at nonscientific problems is just as dumb as the next guy.

Richard Feynman

10.1 INTRODUCTION

The nonlinear Burgers' equation has been of interest to research for many years and has led to the development of efficient and accurate methods for solving steep moving gradient advection-diffusion problems. Usually these methods involve highly complex mathematics and are difficult to implement [Hong and Mao, 1998; Park and Jang, 2000].

Other researchers have used simpler methods, i.e. the Method of Lines (MOL), with an ODE integrator, to solve this nonlinear equation with some success. The MOL is relatively simple and easy to implement, and has been proven to be capable of solving a mildly viscous Burgers' equation (as shown in chapter 6). However, one major drawback with this technique is that it may pose significant restrictions on the time step of integration. In some cases, even the use of a stiff integrator is not able to solve the equation efficiently and accurately [Lee, 1996].

In this chapter, a simple method has been devised that overcomes the stability constraint which restricts the size of the marching increment, Δt , for a given space increment, Δz . The new method consists of the application of Vertical Scaling on the dependent variable and the use of the ODE/MOL approach. Due to the sophisticated heuristics in using a stiff solver, neither the DAE/MOL approach nor a stiff solver is

considered for implementation with Vertical Scaling in MOL. An explicit solver is used because of its simpler heuristics in time-step and error control and it also permits comparison with other methods discussed in this thesis.

10.2 MAXIMIZED TIME STEPPING FOR HIGHLY VISCOUS BE

The stability and accuracy requirement of an explicit ODE solver influences the choice of the time stepping when solving an ODE system. In practice, this time stepping is directly influenced by the step length formula, which is in turn controlled by the tolerated error bound and the error approximation formulae of the explicit algorithm in the solver. One of the simplest forms of these three formulae can be found in the MATLAB ODE45 (in MATLAB 4.2) routine. The widely accepted formula [Gear, 1971; Gustafsson, 1991; Hairer et al., 1989] for predicting a step-size Δt_{n+1} , following a successful step of size Δt_n , is

$$\Delta t_{n+1} = \Delta t_n \left(\frac{\text{TE}}{\|e_{n+1}\|_{\infty}} \right)^{\frac{1}{p+1}}, \quad (10.1)$$

where $\|e_{n+1}\|_{\infty}$ is the infinity norm of the estimate of the local error e_{n+1} of the vector component of the solution Ψ , TE is the tolerated error bound for step $n+1$ and p is the order of the method. More sophisticated schemes have been used for step-size control [Butcher and Chen, 1998; Curtiss and Hirschfelder, 1952; Gustafsson, 1991; Morrison, 1962; Utumi et al., 1996] and they are usually based on error estimates at two or more successive time steps.

While the step size control is based on (10.1), the tolerated error bound, TE, is

$$\text{TE} = \text{tol} * \max(\|\Omega\|_{\infty}, 1), \quad (10.2)$$

which is a very basic form of control for TE. The parameter *tol* is the user-prescribed tolerance and $\|\Omega\|_{\infty}$ is the infinity norm of the vector component of the solution. It can be seen that TE changes its magnitude depending on the size of Ω . If Ω is large, the

tolerated bound will be greater than tol . But when Ω is small, TE will remain constant at the user-prescribed tolerance.

The final control mechanism is the estimate of the local error e_{n+1} of the explicit Runge-Kutta algorithm,

$$\|e_{n+1}\| = \left\| \Delta t_n * \frac{\partial \Omega}{\partial t} * \mathbf{g} \right\|_{\infty}, \quad (10.3)$$

where γ is the MATLAB ODE45 (version 4.2) Runge-Kutta Coefficient for the error approximation. Note that (10.1) to (10.3) can be found in MATLAB ODE45 (version 4.2) subroutine and that $\partial \Omega / \partial t$ it is a vector quantity that is equivalent to the right hand side of (10.4). Clearly, $\partial \Omega / \partial t$ plays a major role in the approximation of the error, as the vector γ is fixed (in MATLAB ODE45 (version 4.2)), and the values of the elements range from -0.0364 to 0.0299 , and Δt_n is the previously determined step size.

From the control equation (10.1), it can be seen that to maximize the time step-size, keep the TE constant, and hold the error approximation at less than TE, we need to have the norms of the Ψ and $\partial \Omega / \partial t$ vectors kept small in the solution process. This can be achieved by scaling the dependent variable of the ODE system by setting $\Omega = Mf * \Phi$, where Mf is the scaling factor. The transformed Burgers equation (3.1) becomes

$$\Phi_t = \mathbf{n} * \Phi_z - Mf * \Phi(\Phi_z), \quad (10.4)$$

where we note that the vertical scaling factor occurs only on the nonlinear convective term. Thus the relative contributions of the convective and diffusive terms can be adjusted using the numerical parameter Mf . The boundary and initial conditions are given by

$$\Phi_z = \Omega_z / Mf, \quad (10.5)$$

$$\Phi(z, 0) = \Omega(z, 0) / Mf. \quad (10.6)$$

The ODE45 (MATLAB 4.2) algorithm that uses the above simple control mechanism along with a specified tolerance $tol = 10^{-10}$ and the 4th order FD were used to solve the ODE system specified in (10.4). In this case, the above control mechanism formulae are implemented on Φ and $\partial\Phi/\partial t$. The effect of the Vertical Scaling can be seen by stepping through the Runge-Kutta algorithm, i.e., taking a single time step, and observing the variation in the values of the three control variables. These step-through statistics are shown in Tables 10.1 and 10.2.

From Tables 10.1 and 10.2, it can be seen that the vertical scaling does permit larger time step-sizes Δt_{n+1} , while still maintaining the tolerated error bound TE.

Table 10.1 The effect of Vertical Scaling on the time step-size for $n = 0.01$.

Statistics	$\Delta z = 0.01, \Delta t_n = 0.001$		
Mf	$1*10^{-3}$	1	1e3
TE	$5*10^{-8}$	$1*10^{-10}$	$1*10^{-10}$
$\ \Phi\ _{\infty}$	722	0.72	$7.2*10^{-4}$
$\ e_{n+1}\ $	0.0017	$1*10^{-8}$	$1*10^{-11}$
$ \partial\Phi/\partial t $	$2.76*10^{-4}$	37	0.037
Δt_{n+1}	$4.94*10^{-4}$	$3.07*10^{-4}$	$1.2*10^{-3}$

Table 10.2 The effect of the Vertical Scaling on the time step-size for $n = 1$.

Statistics	$\Delta z = 0.01, \Delta t_n = 0.001$		
Mf	$1*10^{-3}$	1	$1*10^3$
TE	$5*10^{-8}$	$1*10^{-10}$	$1*10^{-10}$
$\ \Phi\ _{\infty}$	500	$5*10^{-4}$	$5*10^{-4}$
$\ e_{n+1}\ $	0.0033	$3.3*10^{-6}$	$3.35*10^{-9}$
$ \partial\Phi/\partial t $	$2.1*10^3$	2	0.0039
Δt_{n+1}	$8.6*10^{-5}$	$9.95*10^{-5}$	$3.9*10^{-4}$

When the scaling factor is greater than 1, the tables show that TE is maintained at the user specified tolerance. The approximated error is reduced by the apparent reduction in the norm of $(\partial\Phi/\partial t)$ resulting in a larger possible time step-size. Note that both these values of n does not pose a problem for the ODE/MOL approach and good results have been achieved in Table 6.4 and 6.1. However, the value of $n = 10^{-3}$ does pose a problem as shown in Table 6.5, and we hope to improve the ODE/MOL approach to handle this highly hyperbolic dominated BE using the vertical scaling.

10.3 NUMERICAL EXPERIMENTS

The Burgers' Model (see section 3.2) is used in this chapter for the numerical model. The model was implemented and solved using the ODE/MOL approach in conjunction with the Vertical Stretching technique described in the preceding sections. The experiments were designed to test the performance of the Vertical Scaling technique on both the parabolic and hyperbolic dominated Burgers' equation. Moreover, the effect of the user-prescribed tolerance for the ODE solver and the order of spatial finite difference on the proposed Vertical Scaling will also be investigated. For all cases, the 4th order finite difference and a tolerance of 10^{-10} in the MATLAB ODE45 solver were used. Tight error tolerances have been selected so that the spatial error dominates.

10.4 RESULTS AND DISCUSSION

Tables 10.3 and 10.4 show the statistics for the integration of a parabolic and a hyperbolic form of Burger's equation, where several different spatial step sizes have been used. These calculations were undertaken without the use of the vertical scaling. Table 10.3 shows very impressive results for a parabolic dominated Burgers' equation where a coarse spatial step of 0.02 achieved an exceptionally high efficiency and accuracy. It also shows that the benefit of using a smaller spatial step is marginal and also as the system becomes stiffer, the computational cost becomes more expensive. A

remarkably good result of 9×10^{-12} for the MRE was obtained for a spatial step size of $\Delta z = 0.01$.

Table 10.4 shows poor results for a viscous Burgers' equation. The MAE is large compared to the values obtained in the mildly viscous case (Table 10.3). For a small spatial step of $\Delta z = 0.001$, lower efficiency and accuracy were achieved. It also shows that the system is not stiff for a large system of ODE's (1000 equations), a contradiction of the normal belief that MOL always yields a stiff system. This is reflected in the low number of failed attempts by the solver. It has been noted that when advection dominates, the ODE's may not be stiff [Shampine and Reichelt, 1994]. These two tables demonstrate that the conventional MOL is not suitable for a viscous or essentially hyperbolic Burgers' equation, but is highly effective for the parabolic forms of Burgers' equation.

Table 10.3 Effect of the step-size on parabolic dominated Burgers' equation.

Statistics	$\Delta z = 0.02$	$\Delta z = 0.01$	$\Delta z = 0.001$
<i>n</i>	1	1	1
Successful Steps	737	2937	293551
Failed Attempts	105	699	53398
Function Evaluations	5053	21817	2.1e6
CPU time (s)	3	13	4.7e3
MRE	6.3×10^{-11}	9.0×10^{-12}	8.1×10^{-12}

Table 10.4 Effect of the step-size on hyperbolic dominated Burgers' equation.

Statistics	$\Delta z = 0.02$	$\Delta z = 0.01$	$\Delta z = 0.001$	$\Delta z = 5 \times 10^{-4}$
<i>n</i>	0.001	0.001	0.001	0.001
Successful Steps	2231	1527	3810	4121
Failed Attempts	2	1	1	2
Function Evaluations	13399	9169	22867	24739
CPU time (s)	7	5	51	110
MAE	1.40	0.28	0.01	0.0009

Table 10.5 and Table 10.6 shows the effect of the Vertical Scaling on the parabolic dominated Burgers' equation and the hyperbolic dominated Burgers' equation respectively. There is a drop in the computational efficiency arising from deterioration in the MRE, of the order of 100, for the parabolic form of Burgers' equation (Table 10.5

and 10.3). The other computational statistics are approximately the same. Table 10.6 also shows that the hyperbolic form shows improvement in the solution statistics, and there is an optimal scaling factor at $Mf = 10^5$, for $\Delta z = 0.001$. This optimal value can easily be obtained by trial and error as implemented in Table 10.6. Also, there is a ten-fold improvement in the CPU time. Usually, there is an optimal value of Mf that yields the best results. Not any $Mf > 1$ will automatically give better results. However, once a Mf value shows sign of giving better results, there is a monotonic improvement in the results with respect to the value of Mf .

Table 10.5 Effect of Vertical Scaling on the parabolic dominated Burgers' equation.

Statistics	$\Delta z = 0.02$	$\Delta z = 0.01$	$\Delta z = 0.001$
n	1	1	1
Mf	10^2	10^2	10^5
Successful Steps	733	2925	293522
Failed Attempts	121	769	53317
Function Evaluations	5125	22165	$2.1 \cdot 10^6$
CPU time(s)	3	13	$3.4 \cdot 10^3$
MRE	$5.3 \cdot 10^{-9}$	$1.3 \cdot 10^{-10}$	$7.8 \cdot 10^{-7}$

Table 10.6 Effect of Vertical Scaling on the hyperbolic dominated Burgers' equation.

Statistics	$\Delta z = 0.001$					
Integration TIME	0.5	0.5	0.5	0.5	0.5	0.5
Mf	10^{-8}	10^{-5}	1	10^5	10^8	10^{10}
n	0.001	0.001	0.001	0.001	0.001	0.001
Successful Steps	4705	4705	3810	519	509	881
Failed Attempts	0	0	1	41	29	284
Function Evaluations	28231	28231	22867	3361	3229	6991
CPU time (s)	63	63	52	8	7	16
MAE	0.01	0.01	0.01	0.01	0.21	1.57

Note that in Table 6.5, we had used a spatial step of 0.005, the 4th order FD and tol of 10^{-6} to solve the same hyperbolic dominated BE ($n = 10^{-3}$), yielding poor accuracy. Obviously better accuracy can be achieved by using a smaller spatial step as used in Table 10.4. A finer spatial step can resolve the higher steep front in a highly hyperbolic dominated PDE and achieve an accurate result, but at the expense of the CPU. Usually a very small spatial step is needed and this leads to very long CPU times. With Vertical

Scaling, we have eradicated this deficiency of large CPU time in the application of fine spatial grids to handle a highly hyperbolic dominated PDE.

Table 10.7 shows the effect of varying the tolerance tol , using the optimal value of Mf , applied to the solution of the hyperbolic dominated form of Burgers' equation. Here, both the relative tolerance and the absolute tolerance are the same, and are represented by the parameter tol . It can be seen that the tolerance is not as effective for increasing the accuracy of the integrator. The increase in accuracy is compensated by the higher computational cost. Obviously, the initial increase of the tolerance from 10^{-6} to 10^{-10} shows the most improvement in accuracy and CPU time. Note that the spatial truncation error will limit the accuracy that can be obtained.

Table 10.7 Effect of the tolerance, using the optimal Mf , applied on hyperbolic dominated Burgers' equation.

Statistics	$\Delta z = 0.001, Mf = 1e5, \mathbf{n} = 0.001, 4^{\text{th}}$ order FD			
Tolerance, T	10^{-6}	10^{-10}	10^{-16}	10^{-20}
Successful Steps	521	519	6005	25225
Failed Attempts	173	41	6	0
Function Evaluations	4165	3361	36067	151351
CPU time (s)	9	7	81	332
MAE	0.44	0.01	0.01	0.01

The use of higher order finite difference representations of the derivatives is simple and easy to implement with the automated differentiation matrix as used in the vectorized form of the MOL. Table 10.8 shows the effect of the order of the finite difference scheme used in the differentiation matrix. The table has been compiled using the optimal Mf , applied on the hyperbolic dominated Burgers' equation. As the order of the finite differencing increases, MAE increases and then decreases. The 14th order is the optimal order for the differentiation matrix for these parameter settings. Similar behaviour and the existence of an optimal order, was found for other parameter settings; the results are not given here. There is an increase of 4.5 times in the computational CPU time and 304 times improvement in the MAE, as compared to those obtained for the second order differentiation matrix. Thus, the optimal order of the differentiation matrix

is crucial in the application of Vertical Scaling to achieve the solution with the best efficiency and accuracy.

Table 10.8 Effect of the order of the spatial finite differencing using the optimal Mf , applied on the hyperbolic dominated Burgers' equation.

	$\Delta z = 0.001, Mf = 10^5, \mathbf{n} = 0.001, tol = 10^{-10}$							
Order	2	4	6	8	10	12	14	16
Successful Steps	538	519	551	583	628	755	869	1034
Failed Attempts	35	41	118	139	144	139	81	106
Function Evaluations	3439	3361	4015	4333	4633	5365	5701	6841
CPU time (s)	6	8	11	14	17	23	27	35
MAE	0.14	0.01	0.006	0.002	0.003	0.003	$4.6 \cdot 10^{-4}$	0.005

10.5 CONCLUSIONS

The above results clearly show that the Vertical Scaling technique in conjunction with the ODE/MOL approach is effective in combating a steep-moving front problem, such as the Burgers' equation. The objective of maximizing the time steps to gain computational efficiency while maintaining accuracy and minimizing the number of function evaluations has been achieved. Note that this method does not improve the accuracy, but it does maintain the accuracy and maximize the time steps that led to a short CPU time. Hence, the accuracy of the scheme is still largely determined by the spatial step used in the numerical scheme.

However, the weakness of the method is that the optimal statistics for the integration are obtained by trial and error. Fortunately, these tasks are easy as shown in section 4 and can possibly be automated. Then an adaptive Vertical Scaling scheme depending on the evolution of the solution can also possibly be created. These improvements need further investigations.

Unfortunately, the Vertical Scaling scheme is not applicable to the Richard's equation, where the PDE has a complex interdependence relationship of fluid pressures, saturation and relative permeability. These complex relationships complicate the effect of

Vertical Scaling on the dependent physical variable, which does not kept the norms of the scaled dependent variable and its time derivative vector small in the solution process. Hence, the application of Vertical Scaling is not advocated for the Richard's model.

2-D Infiltration Models

To affect the quality of the day, that is
the highest of arts

Henry David Thoreau

11.1 INTRODUCTION

In the preceding chapters, some success has been shown in the ability of low order finite differencing, Vertical Scaling, and the composed forms of RE, in conjunction with the ODE/MOL approach, to simulate the 1-dimensional steep moving front problem or water infiltration into very dry soils. Vertical Scaling overcomes the time stepping constraint for the integration of the highly viscous Burgers' equation, resulting in a large improvement in computational efficiency and accuracy. However, Vertical Scaling does not work for all forms of Richard's equation. In addition, the results also show that for a very steep moving front in a dry initial condition, the composed form of RE, as compared to the decomposed form of RE, yields better accuracy and a shorter computational time. Furthermore, the use of a 2nd order finite difference approximation for the first order derivative is found to be the most effective order for simulation of a steep moving front in a very dry soil.

In this chapter, we investigate the capability of the ODE/MOL approach for a 2-D numerical model. For a 2-D infiltration model, spurious oscillation solutions or numerical difficulties near the edge of the inflow region on the top boundary is a common problem. To minimize this problem, usually finer spatial increments are

implemented near the right edge of the ponding area where sharp gradients were expected to develop [Huang et al., 1996, Yeh et al., 1994]. We propose a method of reducing or eliminating this spurious oscillation near the edge of the inflow, in conjunction with the ODE/MOL approach, the use of 2nd order finite difference approximation and the composed form of Richards' equation. Additionally, we investigate whether the proposed method is able to model a 2-D infiltration in very dry soils, effectively and accurately.

11.2 NUMERICAL EXPERIMENTS

We consider two test cases; the first being the Sander et al. (1988) model in (3.6) with the constitutive relation as described in (3.7) to (3.10). These are applied on the physical region in Fig. 3.1. This will be termed the Sander C model ($\mathbf{u}=0.9999$). Obviously the Sander et al. (1988) analytic solutions are no longer applicable to this problem. However, for $g \gg L$, the flow near the line of symmetry, will be approximately the same with that described by the analytic solution of Sander et al. (1988). Symmetry was not applied in this case, and the numerical solutions were calculated for the full region shown in Fig. 3.1.

The second test case is Richards equation applied to the region described in Fig. 3.1, with the boundary conditions given by (3.18 – 3.22). The model is completed by the VGM relations given in (3.24 – 3.27). Here, symmetry was imposed about $x = 0$, to reduce the computation times. This will be termed the Huang model. With the same soil properties and constitutive relationship, Huang et al. [1996] calculated a 4 h infiltration event using 54.5 min of CPU time with a 33Hz computer, which is about 26 times slower than the machine used in this thesis (866 Hz computer). Huang et al. [1996] uses a new convergence criterion for the modified Picard iteration method to solve the flow

equation. The above statistics clearly shows that the constitutive relations and the physical soil properties that we use represent a difficult numerical model.

There are two numerical studies. The first study looks at spurious oscillation solutions near the edge of the inflow region on the top boundary. At the edge of the inlet region, $x = g$ (see (3.20) and Fig. 3.1), and the imposed flux is still q . Where $x > g$, the imposed flux is zero (see (3.19)), and there is a discontinuity in the boundary condition. This discontinuity creates difficulties for the numerical solution and usually results in local oscillations. We attempted to control these oscillations by using a piecewise continuous approximation across the discontinuity.

Let $N_g (x = g, z = 0)$ be the node index at the edge of the input flux region. Similarly let N_{g-1} and N_{g+1} be the indices of the discretization nodes just before and just after $x = g$. (see Fig. 11.1).

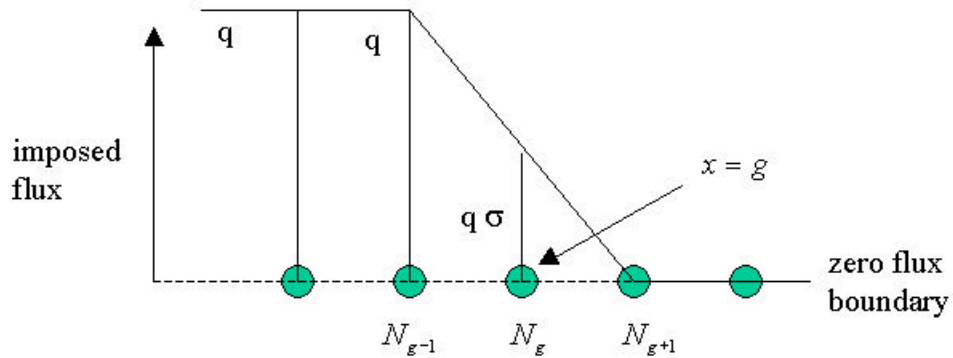


Fig. 11.1 Showing node numbering and flux approximation near discontinuity.

For boundary nodes before $x = g$, the imposed flux is taken as q (which is a constant).

$$Flux_{N_{g-1}} = q. \quad (11.1)$$

At $x = g$, the imposed flux is taken as

$$Flux_{N_g} = \mathbf{s} q, \quad (11.2)$$

where \mathbf{s} is a constant whose value will be tested by numerical experiment. The flux at N_{g+1} , and at other nodes on the no flow part of the boundary, the imposed flux is zero. This arrangement provides the piece wise linear approximation at the edge of the input region.

The properties of the oscillations were investigated by varying the flux distribution at the edges of the inlet, i.e., by varying \mathbf{s} , and the spatial step sizes of the discretization. Our task is to find an optimal value for this reduction of the influx at the edges of the inlet that leads to better accuracy, faster computation time and a reduction or elimination of the spurious oscillation. Moreover, the soil initially has a uniform matric pressure of -10^3 cm of water pressure with a fixed residual volumetric content of 0.05 and is integrated to a model time of 2 days.

The second study looks at the capability of the ODE/MOL approach, with an optimal influx value at the edges of the inlet, to handle a 2-D numerical model. The aim is to ascertain whether the composed form of RE with a 2nd order finite differencing for the first order derivative approximation and an optimal influx value is able to model a 2-D infiltration in very dry soils, effectively and accurately. The model was run for a uniform matric pressure of -10^3 cm, -10^4 cm or -10^5 cm of water pressure for the initial dry condition with a fixed residual volumetric content of 0.05 and integrated out to a model time of 2 days. Comparisons of results were also made at spatial steps of 1cm, 2 cm, and 4 cm to examine the effect of truncation errors.

11.3 RESULTS AND DISCUSSION

11.3.1 Oscillation Investigations.

The Sander C model was implemented using the MATLAB template, and solutions are shown in Fig. 11.2, for $\mathbf{s} = 1$.

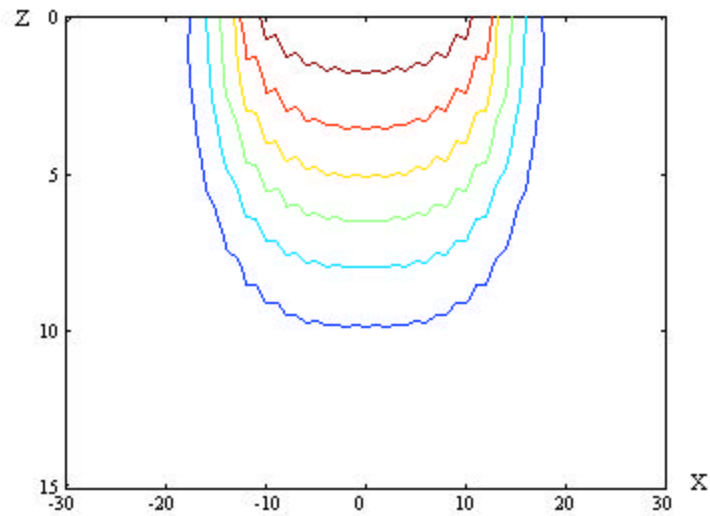


Fig. 11.2 Solution oscillations for $S = 1$ (before modification) of the boundary condition, at 4 minute in time, using a spatial step of 1 cm for both axes.

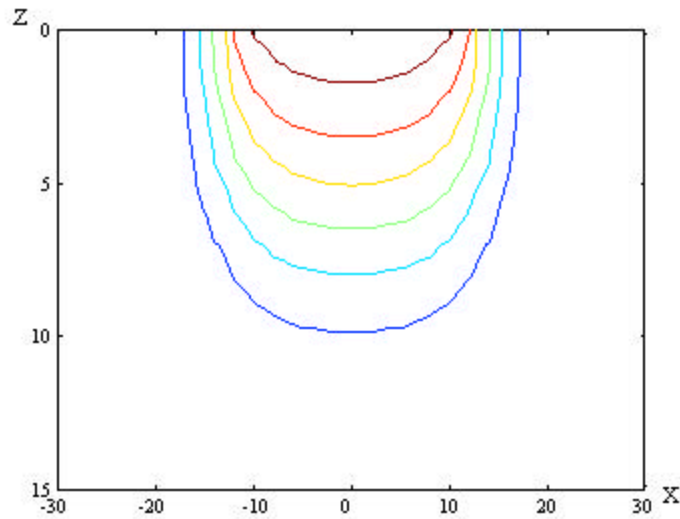


Fig. 11.3 Solution oscillations for $S = 0.5$ (before modification) of the boundary condition, at 4 minute in time, using a spatial step of 1 cm for both axes.

In Fig. 11.2, there is a clear high frequency oscillation in the contour of Ψ . In Fig. 11.3, the same contour is shown where the discontinuity at the edges of the input flux region, have been smoothed out. Close examinations of the initial time steps in the numerical solution indicate that these oscillations arise at the corners of the input flux region. As solution time increases, these oscillations propagate and affect the full

solution. These results clearly show the beneficial effects of smoothing the input flux. Use of this smoothing did not change the CPU time required.

The Huang model was run over the full region in Fig. 3.1. Symmetry was imposed in the Huang model. Again the MATLAB template was used to implement the numerical solutions, and some of these are shown in Fig. 11.3. Note that the plane of symmetry is at $x = 0$, and the left hand side is shown. Fig. 11.3a shows an early time solution with no smoothing of the input flux.

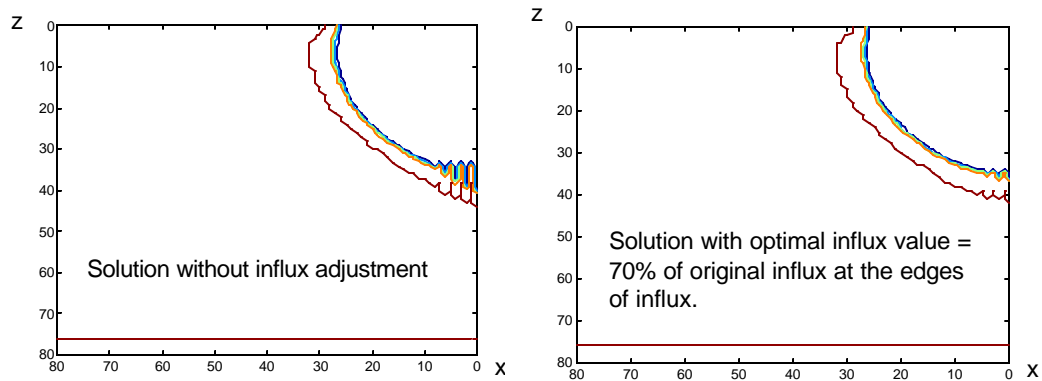


Fig. 11.3 Oscillation at the influx before and after modification of influx conditions, reflected via the pressure contour of the domain at 2 days in time, using a spatial step of 1cm.

The contour lines of Ψ show numerical oscillations, which are particularly severe near the plane of symmetry. The edge of the input flux region seems to have little effect on the generation of the numerical oscillations in the contours. The only real difference between the Sander C and the Huang models, is in the constitutive relations, and in the nonuse/use of symmetry.

A technique similar to the piecewise linear approximation in Fig. 11.1 was used at the line of symmetry. This served to adjust the input flux at the boundary made at the origin; the plane of symmetry. The results are shown in Fig. 11.3b for $s = 0.7$, and these indicate significant amplitude control of the oscillation. One feature of this experiment was a significant reduction in the CPU time required.

Tables 11.1 to 11.3 show the efficiencies of the numerical algorithm for various spatial steps and values of \mathbf{s} . The quantity FME (see (3.30)) measures mass balance errors relative to the accumulated net influx across the space in Fig. 3.1. Thus

$$Q = \int_0^t \left(\int_0^s \text{input flux} dx \right) dt ,$$

for the Huang model, and this obviously depends on the value of \mathbf{s} . Thus FME is essentially a measure of the mass balance errors. In each of the tables, FME takes an optimum or minimum value. The CPU time shows marked changes, relating to the value of \mathbf{s} used in the calculations. Most importantly, the results for a fine grid simulation depict an optimal value of $\mathbf{s} = 0.7$, and this corresponds to the smallest CPU time. These tables also show that the CPU time is correlated to the statistics of the ODE solver, i.e. the number of successful steps, failed attempts and derivative function call increases with larger CPU time and decreases with smaller CPU time.

For a coarse grid of 4 cm (Table 11.1), the optimal value of \mathbf{s} is 0.5, which corresponds to fastest CPU time. Comparing this optimal case with the \mathbf{s} value case (no alteration of the influx), there is about 100 times improvement in the CPU time and 4 times reduction in the accuracy of the FME value.

Table 11.1 Efficiencies of the numerical algorithm at spatial step of 4 cm (21*21 nodes).

\mathbf{s}	1	0.9	0.7	0.5	0.3	0.1
Successful steps	31 114	22 864	3 839	273	267	288
Failed attempts	2 074	1 482	261	1	0	3
Derivative Function Call	199 129	146 077	24 601	1 645	1603	1 747
CPU Time (s)	1 183	909	150	8	11	11
FME (%)	20.0	15.1	5.6	3.6	24.8	21.5

Table 11.2 shows the efficiencies of the numerical algorithm at a spatial step of 2 cm. The optimal value for this spatial step is $s = 0.7$. Comparing with the $s = 1$ case, there is about a 26-fold improvement in the CPU time and 9 times improvement in its FME value.

Table 11.2 Efficiencies of the numerical algorithm at s patial step of 2 cm (41*41nodes).

s	1	0.9	0.7	0.5	0.3	0.1
Successful steps	35 860	13 857	529	1 170	92 521	289 438
Failed attempts	2 322	848	0	58	6 133	19 284
Derivative Function Call	229 093	88 231	3 175	7 369	591 925	1 852 330
CPU Time (s)	4 210	1 640	63	142	14 973	34 840
FME (%)	14.87	4.84	1.61	21.35	23.89	7.97

Table 11.3 Efficiencies of the numerical algorithm at spatial step of 1 cm (81*81nodes).

s	1	0.9	0.7	0.5	0.3
Successful steps	146 387	60 805	1 070	3 140	213 818
Failed attempts	9 597	3 943	5	169	14 129
Derivative Function Call	935 905	388 489	6 451	19 855	1 367 680
CPU Time (s)	67 028	244 300	457	1 427	97 118
FME (%)	3.00	2.26	0.75	0.77	2.27

Table 11.3 shows the results for a fine grid (spatial step of 1 cm) simulation. The optimal value of s is 0.7. Comparing with the $s = 1$ case, there is about 150-fold improvement in the CPU time and 4-fold improvement in its FME value.

11.3.2 Capability of the proposed numerical scheme.

Tables 11.4 and 11.5 show the results of the simulation for various values of initial dryness with a fixed residual volumetric content of 0.05. The integration were to a model time of 2 days with a fixed optimal $S = 0.7$ value.

Table 11.4 Efficiencies of the numerical algorithm at spatial step of 4 cm (21*21nodes), with an optimal $S = 0.7$ value.

Initial Water Pressure (cm)	-10^3	-10^4	-10^5
Successful steps	273	485	740
Failed attempts	1	8	11
Derivative Function Call	1 645	2959	4507
CPU Time (s)	8	14	22
FME (%)	3.6	3.6	3.6

Table 11.5 Efficiencies of the numerical algorithm at spatial step of 2 cm (41*41nodes), with an optimal $S = 0.7$ value.

Initial Water Pressure (cm)	-10^3	-10^4	-10^5
Successful steps	529	820	1 138
Failed attempts	0	1	7
Derivative Function Call	3 175	4 927	6 871
CPU Time (s)	63	91	128
FME (%)	1.61	1.61	1.61

Table 11.6 Efficiencies of the numerical algorithm at spatial step of 1 cm (81*81nodes), with an optimal q values of $0.7*q$ at edges of the inlet.

Initial Water Pressure (cm)	-10^3	-10^4	-10^5
Successful steps	1 070	3 513	32 201
Failed attempts	5	169	2 042
Derivative Function Call	6 451	22 093	205 459
CPU Time (s)	457	1 587	14 787
FME (%)	0.75	0.75	0.75

The results indicate that spatial discretization is the most important factor controlling the accuracy of numerical solutions in the simulation of water flow, i.e. a particular spatial step will yield similar value of FME, regardless of the initial condition in the soils. However, the initial condition influences the CPU time, which rises for the drier soils.

11.4 CONCLUSIONS

The results show that the ODE/MOL approach, in conjunction with the use of 2nd order finite difference approximation, the composed form of Richards' equation and an optimal s value is able to model a 2-D infiltration in very dry soils and do so effectively and accurately. For a coarse spatial step of 4cm (21*21nodes), for all initial conditions (-10^3 cm, -10^4 cm or -10^5 cm of water pressure), an accuracy of FME =1.61% is obtained and the integration time is within a few minutes (Table 11.5). A 2-fold improvement in the FME could be obtained by using a spatial step of 2 cm, but at the expense of the CPU time that is 10 to 100 times longer (Table 11.6). A 5-fold increase in accuracy of the FME needs 10 to 100 times CPU time, using a spatial step of 0.5 cm. In fact, the longest CPU time needed for a 5-fold increase in accuracy is about 5 hours for an integration model time of 2 days (Table 11.6).

The oscillations near the plane of symmetry (see Fig. 11.3) were not expected. These probably arise from the corner point at $x = 0$, where a vertical flux condition and a horizontal flux condition need to be applied. This is the probable cause of stress in the numerical solution near the centre line. The trick of using a piecewise linear approximation seems to smooth out the oscillations near the plane of symmetry, but it does not eliminate them.

The presence of these numerical oscillations poses problems for including the effects of hysteresis in any model for solving Richard's equation. Braddock et al. [2001] developed the Parlange [1976] model for hysteresis, to include multiple switching between wetting and drying phases. These algebraic relations can be included in the Huang model. The inclusion requires that the switching history at each lattice point in the problem domain needs to be retained in the computer memory. This does require that considerable storage space be available to evaluate the $\mathbf{q} - \Psi$ relationship.

Such a model has been developed and run for the problem domain in Fig. 3.1. The estimation of each switch point requires that the ODE integrator be modified to determine when $\partial \mathbf{q} / \partial t$, or $\partial \Psi / \partial t$, changes sign, and hence switches phase. Such estimates can be built into the 'events feature' of ODE45.

However, we have seen that oscillation occur in the numerical solutions. These serve to provide 'artificial' tripping of the event feature, and provide false switch points to the hysteresis model. The hysteresis effect in the runs using Huang model, quickly filled the computer memory with erroneous phase switches. All arose early in the calculation and the machine ran out of memory. It was not possible to determine the effects of hysteresis on the flow field.

Conclusions and Recommendations

A conclusion is the place where you
got tired of thinking.

Anon

12.1 CONCLUSIONS

This thesis presents the vectorized ODE/MOL approach to solve time-dependent PDEs, in particular, the Richards' equation for modelling infiltration in very dry soils. Due to the steep pressure head or water volumetric content and the rapid advancement of the wetting front, accurate numerical solutions for infiltration into a dry soil are usually difficult to obtain. Additionally, such problems usually require very small time steps and large computation times. Some success has been shown in the ability of the ODE/MOL approach to simulate such problems in 1-Dimensional cases, while the 2-Dimensional cases need significant research and improvement before their full potential in routine applications for difficult nonlinear problems, such as RE, can be realized. However, we have achieved good results for a particular case of 2-Dimensional water flow simulation– a finger flow (without hysteresis).

For the 1-Dimensional cases, we have shown that (in conjunction with the ODE/MOL approach):

- 1) The composed form of RE yields better accuracy and a shorter computational time than the decomposed form of RE. Moreover, the decomposed form of RE was shown to fail in the case of a highly steep front problem with very high initial water pressure.

- 2) For a parabolic dominated PDE or a less steep moving front problem, the use of a high order scheme is beneficial, there is more rapid convergence to high accuracy and an optimal order for the finite differencing. The results also show that the use of an odd order scheme in ODE/MOL approach could cause an instability problem, which leads to the termination of the ODE integrator due to the minimum integration step not being able to satisfy the tolerance. Furthermore, the 2nd order finite difference scheme is the most effective scheme for the numerical solution of Richard's equation or highly steep moving-front problem.
- 3) Varying the different order of finite difference stencils in the FODM does not improve the accuracy and no optimal varying FODM is found, as there were no correlation between the error indicators and the varying FODM. However, the use of a higher order FODM immediately after a lower order FODM scheme for all nodes led to a controlled numerical error without oscillation.
- 4) The moving grid algorithm, CDGA, is shown to be complicated and the tuning parameters in the grid generator are hard to determine. The CPU requirement for the moving grid algorithm is many orders of magnitude larger than that required in a non-adaptive scheme.
- 5) The implementation of non-uniform grid stretching in conjunction with the ODE/MOL approach is able to produce better accuracy for short-term integration (where the region requiring high spatial resolution is known) as compared to the vectorized ODE/MOL approach alone, but at a computational CPU cost that is 10 to 30 times longer in duration.

- 6) The Vertical Scaling in conjunction with the ODE/MOL approach has been proven to be effective in combating a steep-moving front problem, such as the Burgers' model. The objective of maximizing the time steps to gain computational efficiency while maintaining accuracy and minimizing the number of function evaluations have been achieved. Unfortunately, it is not applicable to Richard's equation.

For the 2-Dimensional cases, we have shown that the ODE/MOL approach, in conjunction with 2nd order finite difference approximation, the composed form of Richards' equation and optimal values of \mathbf{s} in the ranges 0.5 to 0.7, is able to model a 2-D infiltration in a dry soil with a very high initial water pressure (of -10^3 cm, -10^4 cm or -10^5 cm of water pressure), and to do so effectively and accurately. By reducing the influx at the edges of the inlet (with an optimal \mathbf{s} value) and maintaining constant flux elsewhere in the inlet facilitated a smooth transition from the region of influx to a no flow region on the boundary. This smooth transition reduced the oscillation to a minimal level, which leads to better accuracy and faster computation time.

In this thesis, it has been shown that the MATLAB PDE template is very suitable for numerical modelling of PDEs. The plug and play mode of modifying the PDE template for solving time-dependent PDEs is a major factor contributing to the success of the numerical experiments, in contrast to more conventional approaches using Pascal, Fortran, C or C++. When developing a new problem, we need only to change the initial and boundary conditions and the structure of the new domain. The template will automatically discretize the region and create the differential matrix for the problem. A MATLAB built-in ODE integrator is then used to advance the solution in time. This liberates the programmer from the tedium and distraction of programming techniques per se. Also the extensive built-in MATLAB numerical library, packages of ODE integrators, and graphics facilities are extremely useful. The template approach is

versatile and efficient for solving initial value PDE problems in both 1-D and 2-D spatial dimensions. Furthermore the template can be extended for an irregular shaped domain.

Once one is familiar with the PDE template in chapter 4, many PDE problems can be solved relatively quickly. In contrast, it would take days or weeks to write and debug a finite difference program in Pascal or Fortran to solve a particular problem. Good results for a 2-Dimensional finger flow (without hysteresis) and some successes for a 1-Dimensional flow problem were obtained. It should be noted that the ODE/MOL approach is not totally suitable for highly steep moving-front problems in very dry soils for 1-Dimensional cases. Even for the 2-Dimensional case, more research is required to improve this ODE/MOL approach to achieve the aims stated in this thesis. More numerical experiments on different physical soil property and water flow models are needed to ascertain its effectiveness and usefulness. Due to time constraint, we set this task for future exploration and analysis.

However, the foundation of the ODE/MOL or DAE/MOL has been laid, and an extension or improvement is not really a problem. The only difference between the ODE/MOL and DAE/MOL approach is the integrator used to solve the system of ODEs i.e. ODE solver or DAE solver. The next section proposes extensions or improvements that can be made to the ODE/MOL or DAE/MOL approach. Most of these improvements are theoretically involved and thus more complex. This is the reason why they are not carried out in this thesis.

12.2 FUTURE RESEARCH

1) The Different Forms of Richards' Equation.

Chapter 5 illustrated that the different forms of RE (constant air phase equation) play an important role in the success of the numerical scheme, especially in the

simulation of very steep moving front in very dry soils. The results have shown that the composed form of RE, both Ψ based and mixed form, outperform the decomposed form of RE. Moreover, Forsyth et al., [1995] found that the two phase (non-constant air phase pressure) equations were always easy to solve numerically even for very dry, heterogeneous problems. This contrasts with the supposedly simplified constant air phase equations, which can be very difficult to solve unless special care is taken. Hence it makes sense to investigate the different forms of RE, and multi-phase flow conditions.

2) **Method of Characteristics**

Nieber [1996] suggested that some variation of the method of characteristics can be used to treat the convection part of the governing equations, and this facilitates the accurate treatment of convection dominated flows. Such schemes include those by Huang et al. [1994], and Mulder and Gmelig-Meyling [1991]. The former scheme is for unsaturated water flow and the latter for multiphase flow.

3) **ENS Scheme.**

Hilden and Steinebach [1998] stated that classical upwinding finite difference schemes using upwind scheme in MOL give poor resolutions, oscillations or no results for problems with changing behaviour and other special effects, e.g., sharp gradients. They advocated the use of high-resolution schemes, i.e. ENO (Essentially Non-oscillating) schemes which applied to smooth solutions, reaches high order and, when applied to shocks and discontinuities, avoids oscillations and numerical diffusion. The ENO scheme uses a FD stencil of fixed size, which steered away from discontinuities and adapts to the upwind direction. In this way, oscillation-free results that are of high order in smooth solution regions are obtained.

4) **Finite Volume Method.**

Taylor-series expansions work well when the requirement of differentiability is met. In fact, the series will not converge when the function that is approximated is not smooth, i.e. shock wave or other discontinuities are present. Aubert et al. [1995] overcome this requirement of differentiability by using a finite-volume approximation in solving the conservation equations and evaluating the flux terms at the cell interfaces by the solution of a Riemann problem. This method leads to a numerical technique that treats the problem of discontinuous solutions in a very specific way. Hence, the finite volume method may be incorporated in the ODE/MOL approach to yield better numerical solutions, especially for shock wave type problem.

5) **Robustness to Nonnormality**

Nonnormal matrices can exhibit spectral instability [Henrici, 1962, van der Sluis, 1975, Chatelin, 1993, Chaitin-Chatelin, 1997]. While a normal matrix has a diagonal Schur form, a nonnormal matrix has a triangular Schur form. In finite precision arithmetic computation, Chaitin-Chatelin and Gratton [1996] pointed out that the presence of nonnormality decreases the reliability and the convergence quality of numerical methods for the evolution equations. Hence there is a need to look at the characteristics of the matrices, i.e. nonnormality, in the vectorized ODE/MOL approaches.

6) **Hysteresis Switches**

Numerical oscillations provide trigger mechanisms for the switching between wetting and drying phases. In the extended Parlange model of hysteresis, this produces storage problems for the $q - \Psi$ relation. Research is needed so that the features of the numerical oscillations can be filtered out. Then the physical features of proper switching of phases can be studied and incorporated correctly into the hysteresis.

7) Theoretical Aspect.

Finally, a better understanding of the mathematical aspects of the MOL, ODE and DAE solvers will lead to more effective use of the method, and help to assess its error properties. Understanding the characteristics and properties will invariably lead to more effective use of existing integrators. This will involve an analysis of the error properties of the matrix differential operators and how this interacts with the ODE/DAE solvers.

Appendix A Subroutines for 1-D PDE's Template

Subroutine DOMAIN

Purpose Create and number the grid points in a 1-D region.

Synopsis $[G,k,z,dz,m,mm,b1,b2] = \text{domain}(A,B,\text{Division})$

Parameters

Inputs:

A A real number denoting the start of the discretized region in the z coordinate.
B A real number denoting the end of the discretized region in the z coordinates.
Division An integer containing the total number of discretized points in the region.

Outputs:

G A matrix containing all the discretized points with values equal to the numbering or node index scheme and zeros for points located external to the discretized region.
k A vector containing the indices of the non-zero elements in G.
z A vector denoting the physical co-ordinate system in the physical region.
dz The spatial step of the numerical scheme, which is a constant number.
m, mm The size of the G matrix- m rows and mm columns.
b1 The node index of the grid point at the start of the discretized region.
b2 The node index of the grid point at the end of the discretized region.

Description

Create the physical region in the form of a vector, z, denoting the physical co-ordinate system. Using logical operator and the z vector, the discretized region is selected and numbered. These discretized points and the physical co-ordinate system are then manifested as matrices and vectors to be used in the FODM subroutine.

Subroutine STENCIL

Purpose To compute a subset of the first order differential matrices (FODM), derived from a single one-dimensional FD stencil applied on the grid G. It is the main engine in the automatic differencer or FODM subroutine.

Synopsis $D = \text{stencil}(G,\text{aloc},\text{coef},\text{operator})$

Parameters

Inputs:

G A matrix containing all the discretized points with values equal to the numbering or node index scheme and zeros for points located external to the discretized region.
aloc A vector denoting the nodal index number of a discretized point or a batch of discretized points where the derivatives are approximated. This batch of points uses the same finite difference (FD) stencil to compute its derivative.
Operator A vector e.g. [1 2 3 4] for a FD stencil to denote the relative position of a particular FD stencil in relation to its i,j location. i,j is the location where the derivative is approximated. Hence vector [1 2 3 4] denotes that the neighboring FD stencil's points are 1, 2, 3 and 4 points away from i,j .
coef A string matrix containing the coefficients of the FD stencil, whose first row is designated for the differencing point(i,j) and subsequent rows for its neighbouring points.

Outputs:

D A sparse matrix containing a subset of the first order differential matrices, derived from a single one-dimensional FD stencil applied on the grid G.

Description

Create a subset of the first order differential matrices (FODM), derived from a single one-dimensional FD stencil applied on the grid G. The knowledge that
a.)The discretized numbers or nodal index numbers of the grid points give the row's indices of FODM,

b)The discretized grid points' neighbor nodal index number gives the column's indices of FODM, is the main essence of the subroutine. Hence based on the nodal index number of the discretized grid points, coef and the operator, the subset of FODM is easily created. The sum of the different subset of FODM yields the full FODM for the required FD stencils. Please note that the structure of delsq in MATLAB, by Cleve Moler, contributed greatly to the creation of this subroutine.

Subroutine FODM

Purpose Create the 4th order FODM, Az and Azup.

Synopsis [Az , Azup]=fodm(G,dz,z,m,b1,b2)

Parameters

Inputs:

- G A matrix containing all the discretized points with values equal to the numbering or node index scheme and zeros for points located external to the discretized region.
- dz The spatial step of the numerical scheme, which is a constant number.
- z A vector denoting the physical co-ordinate system in the physical region.
- m The number of row in the G matrix- m rows.
- b1 The node index of the grid point at the start of the discretized region.
- b2 The node index of the grid point at the end of the discretized region.

Outputs:

- Azup The upwind FODM for handling the convective terms.
- Az The symmetric FODM for handling the diffusive terms.

Description

Using b1 and b2, the discretized region is categorized into 6 batches of points for a fourth order representation of a first order derivative. Each batches of points are assigned different type of 4th FD stencils to compute its first order derivatives, depending on its physical location in the discretized region. These batches of points are then fed into the "stencil" subroutine to derive the rows of the FODM, whose sum yield the full FODM matrices.

Appendix B Subroutines for 2-D PDE's Template

Subroutine DOMAIN

Purpose Create and number the grid points in a 1-D region.

Synopsis [G,k,x,z,dx,dz,m,mm,b1,b2,b3,b4] = domain(dxn,dzn,A1,B1,A2,B2)

Parameters

Inputs:

- A1 A real number denoting the start of the discretized region in the x coordinate.
- B1 A real number denoting the end of the discretized region in the x coordinates.
- A2 A real number denoting the start of the discretized region in the z coordinates.
- B2 A real number denoting the end of the discretized region in the z coordinates.
- dxn An integer containing the total number of discretized points row-wise in the x matrix.
- dzn An integer containing the total number of discretized points column-wise in the z matrix.

Outputs:

- G A matrix containing all the discretized points with values equal to the numbering or node index scheme and zeros for points located external to the discretized region.
- k A vector containing the indices of the non-zero elements in G.
- x A matrix denoting the x co-ordinate system in the physical region.
- z A matrix denoting the z co-ordinate system in the physical region.
- dx The x spatial step of the numerical scheme, which is a constant number.
- dz The z spatial step of the numerical scheme, which is a constant number.
- m, mm The size of the G matrix- m rows and mm columns.
- b1 The node index of the grid points at the start of the discretized region in the x coordinate.
- b2 The node index of the grid points at the end of the discretized region in the x coordinate.
- b3 The node index of the grid points at the start of the discretized region in the z coordinate.
- b4 The node index of the grid point at the end of the discretized region in the z coordinate.

Description

Create the physical region in the form of x and z matrix. The x and z matrix denotes the x and z physical co-ordinate system respectively. Using logical operator and these coordinate matrices, the discretized region is selected and numbered. These discretized points and the physical co-ordinate systems are then manifested as matrices and vectors to be used in the FODM subroutine.

Subroutine FODM

Purpose Create the 4th order FODM, Ax, Axup, Az and Azup.

Synopsis [Ax,Axup,Az,Azup] = fodm(G,dx,dz,x,z,m,mm,b1,b2,b3,b4);

Parameters

Inputs:

- G A matrix containing all the discretized points with values equal to the numbering or node index scheme and zeros for points located external to the discretized region.
- x A matrix denoting the x co-ordinate system in the physical region.
- z A matrix denoting the z co-ordinate system in the physical region.
- dx The x spatial step of the numerical scheme, which is a constant number.
- dz The z spatial step of the numerical scheme, which is a constant number.
- m, mm The size of the G matrix- m rows and mm columns.
- b1 The node index of the grid points at the start of the discretized region in the x coordinate.
- b2 The node index of the grid points at the end of the discretized region in the x coordinate.
- b3 The node index of the grid points at the start of the discretized region in the z coordinate.
- b4 The node index of the grid point at the end of the discretized region in the z coordinate.

Outputs:

- Axup The upwind FODM for handling the convective terms in the x-axis.
- Ax The symmetric FODM for handling the diffusive terms in the x-axis.
- Azup The upwind FODM for handling the convective terms in the z-axis.
- Az The symmetric FODM for handling the diffusive terms in the z-axis.

Description

Using b1 and b2, the discretized region is categorized into 6 batches of points for a fourth order representation of a first order derivative in the x-axis. Each batches of points are assigned different type of 4th FD stencils to compute its first order derivatives, depending on its physical location in the discretized region. These batches of points are then fed into the "stencil" subroutine to derive the rows of

the FODM, whose sum yield the full FODM matrices. The above procedures are repeated for the z-axis to create the FODM's in the z-axis; using b3 and b4 to categorize another 6 batches of points for a fourth order representation of a first order derivative in the z-axis.

Appendix C CONTENT IN CD-ROM

Note : All problems or examples can be solved by typing 'main' in the MATLAB CONSOLE.

Global Directory (common subroutines share by other directories)

Domain1d-	discretize, number and create the boundary points in a 1-D domain.
Domain2d-	discretize, number and create the boundary points in a 2-D domain.
Endvalues-	compute the analytical solution for the 1-D Richard's equation.
Fodmeve1d-	create the even order FODM, Az and Azup, for a 1-D domain.
Fodmeve2d-	create the even order FODM, Az, Azup, Ax and Axup for a 2-D domain.
Fodmodd1d-	create the odd order FODM, Az and Azup, for a 1-D domain.
Fodmodd2d-	create the odd order FODM, Az, Azup, Ax and Axup for a 2-D domain.
Mapped-	to map a vector into a 2-D domain defined by G.
Mole-	call various order of computational FD stencils and its error coefficient obtained by numerical method.
Mole_sys-	call various order of computational FD stencils and its error coefficient obtained by symbolic method.
Stenc11-	compute a subset of the FODM, derived from a single 1-D FD stencil applied on the grid G. It is the main engine in the automatic differencer or FODM subroutine.
Stencilcoef30	finite difference formulas for the 1 st to 30 th order generated by symbolic computation.

Code Directory (codes used in each chapter of the thesis)

Chapter 4 –Template (self-contain for each directory, using the 4th order FD)

1. 1Dburger- solving 1D Burgers' equation.
2. 2Dburger- solving 2D Burgers' equation.
3. 2pde- solving a couple 1-D PDEs.
4. Heat- solving a simple 1-D heat equation.
5. Hyperbolic- solving a 1-D hyperbolic equation.
6. Laplacian- solving a 2-D Laplacian model with a circular domain.

Chapter 5 – Forms_RE

1. h_composed- solving the 1-D pressure composed form of RE.
2. h_decomposed- solving the 1-D pressure decomposed form of RE.
3. Mix_composed- solving the 1-D mixed composed form of RE.
4. Mix_decomp- solving the 1-D mixed decomposed form of RE.
5. Theta_c&d- for solving the 1-D theta composed and decomposed form of RE.

Chapter 6 - HigherOrder

1. Burger- solving the 1D Burgers' equation using various order FD scheme
2. H_composedRE- solving the pressure composed form of RE using various order FD scheme.

Chapter 7 – VayingFODM

Note: The GUI for solving the heat equation can be activated by typing ‘main’ in the MATLAB CONSOLE. To continue the integration of the PDE at each successful step, type ‘return’ in the MATLAB CONSOLE and hit the ‘Enter’ key.

Main-	the PDE template described in chapter 4.
Goodstep-	the output function invoked for each successful step of the integration.
Manvfodm-	define and activate the graphical user interface (GUI).
Manvfodm_sw-	the switching function file that associate tasks with each component in the GUI.
Rhs-	function that define the right hand side of the PDE.

Chapter 8 – Adaptive

1. AnalyticAdapt- implementing the CDGA scheme to solve the BE with analytical solution.
2. AnalyticNorm- solving the BE with the vectorized ODE/MOL approach.
3. AnalyticUniform- solving the BE with uniform stretching, in conjunction with the vectorized ODE/MOL approach

Chapter 9 – Stretch

Main-	the PDE template described in chapter 4, modified to implement the non-uniform stretching technique to solve the BE.
Rhs-	function that define the right hand side of the PDE.

Chapter 10 - VerticalScale

Main-	the PDE template described in chapter 4, modified to implement the vertical scaling to solve the BE.
Ode45v4-	ode45 version 4 found in MATLAB 4.2.
Rhs-	function that define the right hand side of the PDE.

Chapter 11 - Final2D

Main-	using the 2-D PDE template to solve an infiltration problem with constant influx.
Rhs-	function that define the right hand side of the PDE.
Simp2v-	Simpson’s rule for repeated integrals adapted from Lindfield and Penny, used to calculated the numerical mass in the system.

References

- Abarbanel, S., and A. Ditkowski, 1997.** Asymptotically stable fourth-order accurate schemes for the diffusion equation on complex shapes. *Journal of Computational Physics* 133: 279-288.
- Abarbanel, S. and A. Ditkowski, 1999.** Multi-dimensional asymptotically stable finite difference schemes for the advection-diffusion equation. *Computers and Fluids* 28: 481-510.
- Ainsworth, M and B. Senior, 1998.** An adaptive refinement strategy for hp-finite element computations. *Applied Numerical Mathematics* 26: 165-178.
- Anthnes, R.A., 1970.** Numerical experiments with a two-dimensional horizontal variable grid. *Monthly Weather Review*. 98: 810-822.
- Arneborg, L., and E. Hansen, 1998.** Numerical modelling of advection-dispersion equation in a stretched curvilinear grid using the quickest scheme. I. *Journal for Numerical Methods in Fluids* 28: 1033-1052.
- Aubert S, L. Hallo, P. Ferrand and M. Buffat, 1995.** Numerical simulations of a three-dimensional nozzle, from inviscid to turbulent flows. *International journal of numerical methods for heat and fluid flow* 5: 889-905.
- Babajimopoulos, C., 2000.** Revisiting the Douglas-Jones method for modelling unsaturated flow in a cultivated soil. *Environmental Modelling & Software* 15: 303-312.
- Baca, R.G., J.N. Chung, and D.J. Mulla, 1997.** Mixed transform finite element method for solving the non-linear equation for flow in variably saturated porous media. *International Journal for Numerical Methods in Fluids* 24: 441-455.
- Baca, R.G., I. P. King, and W.R. Norton, 1978.** Finite element models for simultaneous heat and transport in unsaturated soils, in *Proceedings of the Second International Conference on Finite Elements in Water Resources*, pp. 1.19-1.35, Pentech, London.
- Baines, M.J., 1994.** *Moving Finite Elements*, Oxford Univ. Press, Oxford.
- Baines, M.J., 1998.** Grid adaptation via node movement. *Applied Numerical Mathematics* 26: 77-96.
- Barakos, G., E. Mitsoulis, and D. Assimacopoulos, 1994.** Natural convection flow in a square cavity revisited: laminar and turbulent models with wall functions. *International Journal for Numerical Methods in Fluids* 18: 695-719.
- Bao, X.W., J. Yan and W. X. Sun, 2000.** A Three-dimensional Tidal Model in Boundary-fitted Curvilinear Grids Estuarine. *Coastal and Shelf Science* 50: 775-788.

Barrett, R., M. Berry, T.F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine and H. van der Vorst 1994. *Templates for the solution of linear systems : building blocks for iterative methods.* Philadelphia, SIAM.

Berg, P., 1999. Long-term simulation of water movement in soils using mass-conserving procedures. *Advances in Water* 22 : 419-430.

Bickley, W.G., 1941. Formulae for Numerical Differentiation, *The Mathematical Gazette* 25: 19-35.

Biswas, R., J.E. Flaherty and D.C. Arney, 1993. An adaptive mesh-moving and refinement procedure for one-dimensional conservation laws. *Applied Numerical Mathematics* 11: 259-282.

Bouloutas, E. T., 1989. *Improved numerical approximations for flow and transport in the unsaturated zone*, Ph.D. thesis, Dep. Of Civ. Eng., Mass. Inst. Of Technol., Cambridge.

Brackbill, J.U., and J.S. Saltzman, 1980. Numerical Grid Generation Techniques. *NASA Conference Publication* 2166: 193.

Brackbill, J.U., and J.S. Saltzman, 1982. Adaptive zoning for singular problems in two dimensions. *Journal of Computational Physics* 46: 342-368.

Braddock, R.D., J.-Y Parlange, and H. Lee, 2001. Application of a Simple Soil Water Hysteresis Model to the Power Models of Soil Water Relationships. *Transport in Porous Media* 44: 407-420.

Broadbridge, P., and I. White, 1988. Constant rate rainfall infiltration: A versatile nonlinear model, 1, Analytic solution, *Water Resources Research* 24: 145-154.

Brooks, R.H., and A. J. Corey, 1964. Hydraulic properties of porous media. *Hydrology Paper* 3, Colo. State Univ., Fort Collins.

Brutsaert, W.F., 1971. A functional iteration technique for solving the Richards equation applied to two dimensional infiltration problems. *Water Resources Research*. 7: 1583-1596.

Butcher, J.C. and D.J.L. Chen, 1998. ESIRK methods and variable stepsize. *Applied Numerical Mathematics* 28: 193-207.

Campbell, G. S., 1985. *Soil Physics with BASIC*, Elsevier Sci, New York.

Cao, W., W. Huang, and R.D. Russell, 1999a. An r-Adaptive Finite Element Method Based upon Moving Mesh PDEs. *Journal of Computational Physics* 149: 221-244.

Cao, W., W. Huang, and R.D. Russell, 1999b. A study of monitor functions for two-dimensional adaptive mesh generation. *SIAM Journal of Scientific Computing* 20: 1978-1994.

Canuto, C., M.Y. Hussaini, A. Quarteroni, and T.A. Zang, 1988. *Spectral Methods in Fluid Dynamics.* Springer-Verlag, Berlin.

Carlson, N., and K. Miller, 1994. Design and application of a gradient-weighted moving finite element code. Part 1 in 1D, *Technical Report 236*, Purdue University, West Lafayette, IN.

Carpenter, M.H., D. Gottlieb, and S. Abarbanel, 1997. Time Stable Boundary Conditions for Finite Difference Schemes Solving Hyperbolic Systems: Methodology and Application to High Order Compact Schemes. *ICASE Report 93:9*.

Carroll, J., and S. Stewart, 1996. A comparison of some adaptive space mesh solvers for the numerical solution of parabolic partial differential equations. *Computers and mathematical Applications* 31: 97-115.

Carslaw, H.S. and J.C. Jaeger, 1959. Conduction of Heat in Solids. Oxford Univ. Press, London and New York.

Celia, M. A., L.R. Ahuja, and G.F. Pinder, 1987. Orthogonal collocation and alternating-direction procedures for unsaturated flow problems. *Advanced Water Resources* 10: 178-187.

Celia, M. A. and Binning, P., 1992. A mass-conservative numerical solution for two-phase flow in porous media with application to unsaturated flow. *Water Resources Research* 28: 2819-2828.

Celia, M.A., E. T. Bouloutos, and R.L. Zarba, 1990. A general mass conservative numerical solution for the unsaturated flow equation, *Water Resources Research* 26: 1483-1496.

Chaitin-Chatelin, F., and S. Gratton, 1996. Convergence in finite precision of successive iteration methods under high nonnormality, *Journal of BIT Numerical Mathematics* 36: 455-469.

Chaitin-Chatelin, F., 1997. Is nonnormality a serious computational difficulty in practice. In Ronald F. Boisvert, editor, *Quality of Numerical Software, Assessment and Enhancement*: 300-314, London, Chapman & Hall.

Chatelin, P., 1993. The influence of nonnormality on matrix computations, in *Linear Algebra, Markov Chains and Queuing Models*, R.J. Plemmons and C.D. Meyer, eds. Springer, New-York, 13-19.

Chang, W.L., W. Biggar, and D.R. Nielsen 1994. Fractal description of wetting front instability in layered soils, *Water Resources Research* 30: 125-132.

Chang, S., and D.C. Haworth, 1997. Adaptive grid refinement using cell-level and global imbalances. *International Journal for Numerical Methods in Fluids* 24: 375-392.

Clement, R.S., R. De Jong, H.N. Hayhoe, W.D. Reynolds and M. Hares, 1994. Testing and comparison of three unsaturated soil water flow models. *Agricultural Water Management* 25: 135-152.

Cooley, R. L., 1983. Some new procedures for numerical solution of variably saturated flow problems. *Water Resources Research* 19: 1271-1285.

Cowell, W.R., 1984. *Sources and Development of Mathematical Software*. Prentice-Hall, Englewood Cliffs, N.J.

Cox, C. L., W. F. Jones, V. L. Quisenberry and F. Yo, 1994. One-dimensional infiltration with moving finite elements and improved soil water diffusivity. *Water Resources Research* 30: 1431-1438.

Crank, 1956. *The Mathematics of Diffusion*. Oxford Univ. Press, London, and New York.

Curtiss, C.F., and J.O. Hirschfelder, 1952. Integration of stiff equations. *Proceedings of the National Academy of Science* 38, 235-243.

Dane, J. H., and Mathis, F.H., 1981. An adaptive finite difference scheme for the one-dimensional water flow equation. *Soil Science Society of American Journal* 45: 1048-1054.

Dane, J.H. and P.J. Wierenga, 1975. Effect of Hysteresis on the prediction of infiltration, redistribution and drainage of water in a layered soil. *Journal of Hydrology* 25: 229-242.

Dennis, J. E. and R. B. Schnabel, 1996. Numerical Methods for Unconstrained Optimization and Nonlinear Equations, no. 16 in Classics in *Applied Mathematics*, SIAM., Philadelphia.

Dietachmayer, G. D., and K.K. Droegemeier, 1992. Application of Continuous Dynamic Grid Adaption Techniques to Meteorological Modelling. Part I: Basic Formulation and Accuracy. *Monthly Weather Review* 120: 1675-1706.

Dullien, F.A.L., 1979. *Porous Media Fluid Transport and Pore Structure*. Academic Press, San Diego, CA.

El-Kadi, A. I. and G. Ling, 1993. The Courant and Peclet number criteria for the numerical solution of the Richards equation. *Water Resources Research* 29: 3485-3494.

Eliassi, M. and R. Glass, 2001. On the continuum scale modelling of gravity driven fingers in unsaturated porous media. The inadequacy of Richards Equation with standard monotonic constitutive relations and hysteretic equations of state, *Water Resources Research* 37: 2019-2035.

Everett, D.H. and W.I. Whitton, 1952. A general approach to hysteresis: 1. *Transaction of Faraday Society* 48: 749-752.

Fornberg, B., 1988. Generation of Finite Difference Formulas on Arbitrarily Spaced Grids, *Mathematics of Computation* 51: 699-706.

Fornberg, B., 1996. *A Practical Guide to Pseudospectral Methods*. Cambridge University Press, Cambridge.

Forsyth, P. A., Y. S. Wu, and K. Pruess, 1995. Robust numerical methods for saturated-unsaturated flow with dry initial conditions in heterogeneous media. *Advanced Water Resources* 18: 25-38.

Furzeland, R. M., J. G. Verwer, and P.A. Zegeling, 1990. A Numerical Study of Three Moving Grid Methods for One-Dimensional Partial Differential Equations which are based on the Method of Lines. *Journal of Computational Physics* 89: 349-388.

Gear, C.W., 1971. *Numerical Initial Value Problems in Ordinary Differential Equations*, Automatic Computation, Prentice-Hall, Englewood Cliffs, NJ.

Gee, G.W., T. Kincaid, R.J. Lenhard, and C.S. Simmons, 1991. Recent studies of flow and transport in the vadose zone, *U.S. Natl. Rep. Int. Union Geod. Geophys.* 1987-1990, Rev. Geophy 29: 277-239.

Gerald, C.F., and P.O. Wheatley, 1989. *Applied Numerical Analysis*. Fourth Edition, Addison-Wesley Publishing Company.

Glass, R.J., J.-Y. Parlange, and T.S. Steenhuis, 1989a. Wetting front instability, 2, Experimental determination of relationships between system parameters and two-dimensional unstable flow field behaviour in initially dry porous media, *Water Resources Research* 25: 1195-1207.

Glass, R.J., J.-Y. Parlange, and T.S. Steenhuis, 1991. Immiscible displacement in porous media: Stability analysis of three-dimensional, axisymmetric disturbances with application of gravity-driven wetting front instability, *Water Resources Research* 27:1947-1956.

Glass, R.J., and M.J. Nicholl, 1996. Physics of gravity fingering of immiscible fluids within porous media: An overview of current understanding and complicating factors, *Geoderma* 70: 133-163.

Gray, W. G., and S. M. Hassanizadeh, 1991a. Paradoxes and realities in unsaturated flow theory. *Water Resources Research* 27: 1847-1854.

Gray, W. G., and S. M. Hassanizadeh, 1991b. Unsaturated flow theory including interfacial phenomena, *Water Resources Research* 27: 1855-1863.

Gui, W. and I. Babuska, 1986. The h, p and h-p versions of the finite element method in one dimension, Part 1: The error analysis of the p-version; Part 2: The error analysis of the h and h-p version; Part 3: The adaptive h-p version, *Numerical Mathematics* 40: 577-612, 613-657, 659-683.

Gustafsson, K., 1991. Control theoretic techniques for stepsize selection in explicit Runge-Kutta methods. *ACM Transaction in Mathematical Software* 17: 533-534.

Hairer,E., C. Lubich, and M. Roche, 1989. The Numerical Solution of Differential-Algebraic Systems by Runge-Kutta Methods, no. 1409 in *Lecture Notes in Mathematics*, Springer-Verlag New York.

Hanks, R.J. and S.A. Bowers, 1962. Numerical solution of the moisture flow equation for infiltration into layered soils. *Soil Science Society American Proceedings* 26: 530-534.

- Haverkamp, R., and Vaculin, M., 1981.** A Comparative Study of Three Forms of the Richard Equation Used for Predicting One-Dimensional Infiltration in Unsaturated Soil. *Soil Science American Journal* 45: 13-20.
- Haverkamp, R., M. Vaculin, J. Touma, P. J. Wierenga, and G. Vachaud, 1977.** A comparison of numerical simulation models for one-dimensional infiltration. *Soil Science American Journal* 41: 285-294.
- Hawken, D. F., Gottlieb, J. J., and J. S. Hansen, 1991.** Review of some adaptive node-movement techniques in finite-element and finite difference solutions of partial differential equations. *Journal of Computational Physics* 95, 254.
- Henrici, P., 1962.** Bounds for iterates, inverses, spectral variation and field of values of nonnormal matrices, *Numerical Mathematics* 4: 24-40.
- Hilden, M. and G. Steinebach, 1998.** ENO-discretizations in MOL-applications: some examples in river hydraulics. *Applied Numerical Mathematics* 28: 293-308.
- Hillel, D., 1971.** *Soil and Water, Physical Principles and Processes*, Academic, New York.
- Hillel, D., 1980.** *Fundamentals of Soil Physics*, Academic, San Diego, California.
- Hillel, D., and R.S. Baker, 1988.** A descriptive theory of fingering during infiltration into layered soil, *Soil Science* 146: 51-56.
- Hills, R. B., D. B. Hudson, I. Porro, and P. J. Wierenga, 1989.** Modelling one-dimensional infiltration into very dry soils, 1, Model development and evaluation. *Water Resources Research* 25: 1259-1269.
- Hills, R.G., and A. W. Warrick, 1992.** Burgers' Equation: A Solution for Soil Water Flow in a Finite Length. *Water Resources Research* 29: 1179-1184.
- Ho, N. T., R. Gaudu, and C. Thirriot, 1977.** Influence of the hysteresis effect on transient flows in saturated-unsaturated porous media. *Water Resources Research* 13: 992-996.
- Hoffmann, K.A., and Chiang, S.T., 1993.** *Computational Fluid Dynamics*, Volume 1, Academic Press, Inc, New York.
- Hong, Y.C., and X.Z. Mao, 1998.** An efficient numerical scheme for Burgers' equation. *Applied Mathematics and Computers* 95: 37-50.
- Huang, K., B.P. Mohanty, and M.Th. van Genuchten, 1996.** A new convergence criterion for the modified Picard iteration method to solve the variably saturated flow equation. *Journal of Hydrology*. 178: 69-91.
- Huang, W., Y. Ren and R. D. Russell, 1994.** Moving mesh methods based on moving mesh partial differential equations. *Journal of Computational Physics* 113: 279-290.

- Huang, W. and R.D.Russell, 1999.** Moving mesh strategy based upon a gradient flow equation for two dimensional problems. *SIAM Journal of Scientific Computings* 20: 998-1015.
- Huyakorn, P. S. and G.F. Pinder, 1983.** *Computational Methods in Subsurface Flow*. Academic Press, Orlando, FL.
- Jameson, L., 1995.** On the spline-based wavelet differentiation matrix. *Applied Numerical Mathematics* 17: 33-45.
- Jury, W.A., and H. Fluhler, 1992.** Transport of chemicals through soil: Mechanisms, models, and field applications, *Advance Agron.* 47: 141-201.
- Kelley, C.T., C.T. Miller and M.D. Tocci, 1998.** Termination of Newton/Chord iterations and the method of lines. *Siam Journal of Scientific Computing* 19: 280-290.
- Keller, H.B. and V. Pereyra, 1978.** Symbolic Generation of Finite Difference Formulas. *Mathematics of Computation* 32: 955-971.
- Kirkland, M.R., R. G. Hills, and P. J. Wierenga, 1992.** Algorithms for solving Richards' equation for variably saturated soils. *Water Resources Research* 28: 2049-2058.
- Kladivko, E.J., G.E. Van Scoyoc, E.J. Monke, K.M. Oates, and W. Pask, 1991.** Pesticide and nutrient movement into subsurface tile drains on a silt loam soil in Indiana, *Journal of Environmental Quarterly* 20: 264-270.
- Köckler, Norbert, 1994.** Numerical methods and scientific computing : using software libraries for problem solving. Oxford : Clarendon Press.
- Kool, J. B., J. C. Parker, and M. T. van Genuchten, 1985.** Determining soil hydraulic properties from one-step outflow experiments by parameter estimation. I, Theory and numerical studies. *Soil Science Society American Journal.* 49: 1348-1354.
- Lakin, W.D. 1986.** Differentiating matrices for arbitrarily spaced grid points. *International Journal of Numerical Methods in Engeering* 23: 209-218.
- Lee, H. 1996.** *Method of Lines in Soil Physics- A Numerical Study*, Honours Thesis, Griffith University.
- Lee, H.S., R.D. Braddock and G.C. Sander, 1998.** Automating the Method of Lines for Modelling Moisture Flow in the Unsaturated Zone. *Computational Techniques and Applications*: CTAC-98, Pages 361-368, editors B.J. Noye, M.D. Teubner and A.W. Gill, World Scientific Publishing Co.
- Lehmann, F, and P.H. Ackerer, 1998.** Comparison of iterative methods for improved solutions of the fluid flow equation in partially saturated porous media *Transport in Porous Media* 31: 275-292
- Leslie, L. M., G.S. Dietachmayer, and G.J. Holland, 1989.** On the dynamics of multiple vortex interaction. Extended Abstracts 18th Conf. On Hurricanes and Tropical Meteorology, San Diego. *American Meteorological Society* 102-103. ?

- Liakopoulos, A.C. 1966.** Theoretical prediction of evaporation losses from groundwater. *Water Resources Research* 2: 227-240.
- Lindfield, G. and J. Penny, 1995.** *Numerical Methods using MATLAB*. Ellis Horwood, Great Britain.
- Ly, L.N. and P. Luong, 1999.** Numerical multi-block grids in coastal ocean circulation modelling. *Applied Mathematical Modelling* 23: 865 – 879.
- Madsen, N.K. and R.F. Sincovec, 1976.** Software for Partial Differential Equations in *Numerical Methods for Differential Systems*, L. Lapidus and W.E. Schiesser, eds., Academic Press, New York.
- Mandelbrot, B.B., 1982.** *The fractal Geometry of Nature*, W.H. Freeman, New York.
- Matthews, C., 2002.** *Modelling a multi-layered Wasted Damp using the MOL*, Phd Thesis, Griffith University, to appear.
- MathWorks, 2001,** Inc. 24 Prime Park Way, Natick, Mass. 01760-1500.
- Mavriplis, D.J. 1998.** Multigrid Strategies for Viscous Flow Solvers on Anisotropic Unstructured Meshes. *Journal of Computational Physics* 145: 141–165.
- Miller, C. T., G. Christakos, P. T. Imhoff, J. F. McBride, J. A. Pedit, and J. A. Trangenstein, 1998a.** Multiphase flow and transport modelling in heterogeneous porous media: Challenges and approaches. *Advanced Water Resources* 21: 77-120.
- Miller, K. and R.N. Miller, 1981.** Moving finite elements, I. *SIAM Journal on Numerical Analysis* 18: 1019-1032.
- Miller, C. T., G. A. Williams, C. T. Kelley, and M. D. Tocci, 1998b.** Robust solution of Richards' equation for nonuniform porous media. *Water Resources Research* 34: 2599-2610.
- Milly, P. C. D. 1985.** A mass-conservative procedure for time-stepping in models of unsaturated flow. *Advanced Water Resources* 8: 32-36.
- Molz, F.J., and I. Remson, 1970.** Extraction term models of soil moisture use by transpiring plants, *Water Resources Research* 6: 1346-1356.
- Morrison, D. 1962.** Optimal mesh size in the numerical integration of an ordinary differential equation. *Journal of Associated Computing* 9: 98-103.
- Mulder, W.A. and R.H.J. Gmelig-Meyling, 1991.** Numerical simulation of two-phase flow using locally refined grids in three-space dimensions, SPE 21230. *Proceeding of Society of Petroleum Engineers Symposium on Reservoir Simulation*, February 1991, Anaheim, CA, pp 299-306.
- Mulholland, L.S., Y. Qiu, and D.M. Sloan, 1997.** Solution of evolutionary partial differential equations using adaptive finite differences with pseudospectral post-processing. *Journal of Computational Physics* 131: 280-298.

- Mualem, Y. 1976.** A new model for predicting the hydraulic conductivity of unsaturated porous media. *Water Resources Research* 12: 513-522.
- Neuman, S.P., 1973.** Saturated-unsaturated flow seepage by finite element. *Proceedings of ASCE, J. Hydraul. Div.*, 99(HY12).
- Nieber, J.L., 1996.** Modelling finger development and persistence in initially dry porous media. *Geoderma* 70: 209-229.
- Nieber, J.B., T.W.J. Bauters, T.S. Steenhuis, and J.-Y Parlange, 2000.** Numerical simulation of experimental gravity-driven unstable flow in water repellent sand. *Journal of Hydrology* 231: 295-307.
- O'Neill, K., 1981.** Highly efficient, oscillation free solution of the transport equation over long times and large spaces. *Water Resources Research*, 17: 1665-1675.
- Oden, J.T., 1989.** Progress in Adaptive Methods in Computational Fluid Dynamics, Chapter 15, *Adaptive Methods for Partial Differential Equations* (Ed. Flaherty, J.E., Paslow, P.J., Shephard, M.S. and Vasilakis, J.D.), pp. 206-252, Society for Industrial and Applied Mathematics, Philadelphia.
- Oden, J.T. and L. Demkowicz, 1988.** Advances in adaptive improvements: a survey of adaptive methods in computational fluid mechanics, in: A.K. Noor and J.T. Oden, eds., *State of the Art, Surveys in Computational Mechanics* (ASME, New York,).
- Olson, P., 1993.** Summation by Parts, Projections and Stability. *RIACS Technical Report* 93.04.
- Oosterlee, C.W., 1997.** A GMRES-Based Plane Smoother in Multigrid to Solve 3D Anisotropic Fluid Flow Problems. *Journal of Computational Physics* 130: 41-53.
- Pan, L. and P. J. Wierenga, 1995.** A transformed pressure head-based approach to solve Richards' equation for variably saturated soils. *Water Resources Research* 31: 925-931.
- Pan, L. and P.J. Wierenga, 1997.** Improving numerical modelling of two-dimensional water flow in variably saturated, heterogenous porous media. *Soil Science society of America Journal* 61: 335 346.
- Paniconi, C., and M. Putti, 1994.** A comparison of Picard and Newton iteration in the numerical solution of multidimensional variably saturated flow problems. *Water Resources Research* 30: 3357-3374
- Pardhanani, A.L., and G.F. Carey, 2000.** Mapped discretization strategies for curvilinear adaptively redistributed grids in semiconductor device modelling. *Computational Methods in Applied Mechanical Engineering* 181: 365-379.
- Park, H.M., and Y.D. Jang, 2000.** Control of Burgers equation by means of mode reduction, *Internal Journal of Engineering Science* 38: 785-805.

- Parlange, J.-Y., 1976.** Capillary hysteresis and the relationship between drying and wetting curves. *Water Resources Research* 12: 224-228.
- Phillip, J.R. , 1969.** Theory of infiltration. *Advance Hydroscience* 5: 215-296.
- Phillip, J.R. 1970.** Flow in porous media. *Annual Review in Fluid Mechanics* 2: 177-204.
- Pikul, M.F. , R.L. Street, and I. Remson, 1974.** A numerical model based on coupled one-dimensional Richards and Boussineq equations. *Water Resources Research* 10: 295-302.
- Poulovassilis, A., 1969.** The effect of hysteresis of pore water on the concept of independent domains. *Soil Science* 93: 405-412.
- Raats, P.A.C., and W.R. Gardner, 1974.** Movement of water in the unsaturated zone near a water table, in *Drainage for Agriculture: Agronomy*, edited by J. Van Schilfgaarde, *ASA Special Publication* 17: 311-405.
- Rabinovitz, F.M., G.L. Stenchikov, M.J. Suarez, L.L. Takacs, and R.C. Govindaraju, 2000.** A uniform- and variable-resolution stretched-grid GCM dynamical core with realistic orography. *Monthly Weather Review* 128: 1883-1898.
- Rathfelder, K., and L. M. Abriola, 1994.** Mass conservative numerical solutions of the head-based Richards' equation. *Water Resources Research* 30: 2579-2586.
- Redinger, G. J., G. S. Campbell, K. E. Sexton, and R. I. Papendick, 1984.** Infiltration rate of slot mulches: Measurement and numerical simulation. *Soil Science of America Journal* 48: 982-986.
- Rice, R.J. 1993.** *Numerical Methods, Software and Analysis*. 2nd edition, Boston : Academic Press.
- Richards, L.A., 1931.** Capillary conduction of liquids through porous mediums. *Physics I*: 318-333.
- Ritsema, C.J., Dekkar, L.W., Nieber, J.L., and T.S. Steenhuis, 1998.** Modelling and field evidence of finger formation and recurrence in a water repellent sandy soil, *Water Resource Research* 34: 555-567.
- Roache, P.J. 1976.** *Computational Fluid Dynamics*. Hermosa Publishers.
- Roberts, F.J., and B.A. Carbon, 1972.** Water repellency and its role in forming preferred flow paths in soils, *Australian Journal of Soil Research* 10: 35-42.
- Ross, P. J., 1990.** Efficient numerical methods for infiltration using Richards' equation. *Water Resources Research* 26: 279-290.
- Ross, P. J., 1992.** Cubic approximation of hydraulic properties for simulations of unsaturated flow. *Water Resources Research* 28: 2617-2620.

Ross, P. J. and K.L. Bristow, 1990. Simulating water movement in layered and gradational soils using the kirchoff transform. *Soil Science Society of America Journal*.54: 1519-1524.

Rothe, E., 1930. Zweidimensionale parabolische Randwertaufgaben als Grenzfall eindimensionaler Randwertaufgaben. *Mathematische Annalen* 102: 650-670.

Rubin, J 1968. Theoretical analysis of two-dimensional, transient flow of water in unsaturated and partly unsaturated soils. *Soil Science Society of America Proceeding*. 32: 607-615.

Rubin, J., and R. Steinhardt, 1963. Soil water relations during rain infiltrations: I. Theory. *Soil Science Society of America Proceeding* 27: 246-251.

Saffman, P.G. and G. Taylor, 1958. The presentation of a fluid into a porous medium or Hele-Shaw cell containing a more viscous liquid, *Proc. R. Soc. Lond. Ser. A* 245: 312-331.

Sander, G.C., J.-Y. Parlange, V. Kuhnelt, W.L. Hogarth, D. Lockington, and J.P.J. O' Kane, 1988. Exact nonlinear solution for constant flux infiltration. *Journal of Hydrology* 97: 341-346.

Schiesser, W.E., 1991. *The Numerical Method of Lines: Integration of Partial Differential Equations: ODEs, DAEs, and PDEs*. Academic Press, San Diego, 1991.

Schiesser, W.E., 1994. *Computational Mathematics in Engineering and Applied Science*, CRC Press, Boca Raton, Florida.

Schnabel, R.R. and E.B. Richie, E. B., 1984. Calculation of internodal conductances for unsaturated flow simulations: a comparison. *Soil Science Society of America Journal* 48: 1006-1010.

Shampine, L.F., and M.W. Reichelt, 1994. The MATLAB ODE, *Suite Reptort*. 94-6, Math. Dept., SMU, Dallas, TX.

Si, B.C. and R.G. Kachanoski, 2000. Unified solution for infiltration and drainage with hysteresis: Theory and field test. *Soil Science Society of America Journal* 64: 30-36.

Simunek, J., K. Huang, M. Sejna, and M. Th. van Genuchten, 1997. *The HYDRUS-1D Software Package for Simulating the One-Dimensional Movement of Water Heat, and Multiple Solutes Variably Saturated Media*, Version 1.0. US Salinity Laboratory, USDA/ARS, Riverside, CA.

Simunek, J. and M. Th. van Genuchten, 1994. The CHAIN-2D code for simulating the two-dimensional movement of water, heat, and multiple solutes in variably-saturated porous media. *Technical Report* 136, US Salinity Laboratory, USDA/ARS, Riverside, CA.

- Simunek, J., T. Vogel, and M. Th.van Genuchten, 1994.** The SWMS-2D Code for Simulating Water Flow and Solute Transport in Two-Dimensional Variably Saturated Media. Version 1.21. *Technical Report 132*, US Salinity Laboratory, USDA/ARS, Riverside, CA.
- Smith, R.E., 1990.** Analysis of infiltration through a two-layer soil profile, *Soil Science American Journal* 54: 1219-1227.
- Steenhuis, T.S., and J.-Yves Parlange, 1990.** Preferential Flow in Structured and Sandy Soils, *Engineering Cornell Quarterly* 25: 7-14.
- Strad, B. 1991.** *Summation by parts for finite difference approximations for d/dx.* Department of Scientific Computing, Upsala University, Upsala, Sweden.
- Thompson, J. F., 1985.** A survey of dynamically-adaptive grids in the numerical solution of partial differential equations. *Applied Numerical Mathematics* 1: 3?
- Thompson, J.F., and C.W. Mastin, 1985.** Order of difference expressions in curvilinear coordinate systems. *Journal of Fluids Engineering* 107: 241-250.
- Tocci, M.D., C.T. Kelley, and C.T. Miller, 1997.** Accurate and economical solution of the pressure head form of Richards' Equation by the method of lines. *Advances in Water Resources* 20: 1-14.
- Tzimas, E., 1979.** The measurement of soil water hysteretic relationships on a soil monolith. *Journal of Soil Sciences* 30: 529-534.
- Utumi, M., R. Takaki and T. Kawai, 1996.** Optimal Time Step Control for the Numerical Solution of Ordinary Differential Equations. *SIAM Journal on Numerical Analysis* 33: 1644-1653.
- van der Sluis, A. 1975.** Perturbations of eigenvalues of nonnormal matrices. *Communications of the ACM* 18: 30-36.
- van Genuchten, M. T., 1978.** Calculating the unsaturated hydraulic conductivity with a new closed form analytical model, *Report 78-WR-08*, Water Res. Program, Dep. of Civ. Eng., Princeton Univ., Princeton, N. J.
- van Genuchten, M. T., 1980.** A closed-form equation for predicting the hydraulic conductivity of unsaturated soils. *Soil Science of America Journal* 44: 892-898.
- van Genuchten, M. T., 1991.** Recent progress in modeling water flow and chemical transport in the unsaturated zone. In: Kienitz, G., Milly, P.C.D. et al. (Eds.), *Hydrological Interactions Between Atmosphere, Soil and Vegetation*. IAHS.
- Vaculin, M., D. Khanji and G. Vachaud, 1976.** Etude experimentale et numerique du drainage et de la recharge des nappes a surface libre, avec prise en compte de la zone non saturee, *J. de Mec.* 15: 307-340.
- Vauclin, M., D. Khanji, and G. Vachaud, 1979.** Experimental and numerical study of a transient, two-dimensional unsaturated-saturated water table recharge problem. *Water Resources Research* 15: 1089-1101.

- Vogel, T. and M. Cislerova, 1988.** On the reliability of unsaturated hydraulic conductivity calculated from the moisture retention curve. *Transport in Porous Media* 3: 1-15, 1988.
- Vogel, T., M. Cislerova, and J. W. Hopmans, 1991.** Porous media with linearly variable hydraulic properties. *Water Resources Research* 27: 2735-2741.
- Warren, G.P., W.K. Anderson, J.L. Thomas and S.L. Krist, 1993.** Grid convergence for adaptive methods, in: M.J. Baines and K.W. Morton, eds., Oxford University Press, *Numerical Methods for fluid Dynamics* 4: 317-328.
- Warrick, A.W., 1991.** Numerical approximations of Darcian flow through unsaturated soil. *Water Resources Research* 27: 1215-1222.
- Warrick, A.W. and G.W. Parkin, 1995.** Analytical solution for one-dimensional drainage: Burgers' and simplified forms. *Water Resources Research* 31: 2891-2894.
- Watson, K.K., V.A. Sardana and G.C. Sander, 1995.** Comparison of analytical and numerical results for constant flux infiltration. *Journal of Hydrology* 165: 101-112.
- Weideman, J.A. and S.C. Reddy, 2000.** A MATLAB Differentiation Matrix Suite. *ACM Transaction on Mathematics and Software* 1: 1-49.
- Whisler, F.D., and A. Klute, 1965.** The numerical analysis of infiltration considering hysteresis into a vertical soil column at equilibrium under gravity. Soil Science Society of America Proceeding 29: 489-494.
- Wihelmson, R. B., and C.S. Chen, 1982.** A simulation of the development of successive cells along a cold outflow boundary. *Journal of Atmospheric Science* 39: 1466-1483.
- Wilson, H.B. and L.H. Turcotte, 1994.** *Advanced Mathematics and Mechanics Applications Using MATLAB*, CRC Press, Boca Raton.
- Williams, G. A., and C. T. Miller, 1999.** An evaluation of temporally adaptive transformation approaches for solving Richards' equation. *Advances in Water Resource* 22: 831-840.
- Williams, G.A., C.T. Miller, and C.T. Kelley, 2000.** Transformation approaches for simulating flow in variably saturated porous media. *Water Resource Research* 36: 923-34.
- Wouwer, A.V., P. Saucez, and W.E. Schiesser, 2001.** *Adaptive Method of lines*. CRC Press.
- Wouwer, A.V., P. Saucez and W.E. Schiesser, 1998.** Some user-oriented comparisons of adaptive grid methods for partial differential equations in one space dimension, *Applied Numerical Mathematics* 26: 49-62.

Yeh, G. T., 1987. *FEMWATER: A Finite Element Model of Water Flow through Saturated-Unsaturated Porous Media – First Revision.* Oak Ridge National Laboratory, Oak Ridge. TN, ORNL-5567/RI edition.

Yeh, T.J, A. Guzman, R. Srivastava, and P. E. Gagnard, 1994. Numerical Simulation of the Wicking Effect in Linear Systems. *Ground Water* 32: 2-11.

Zegeling, P., 1992. *Moving-grid methods for time-dependent partial differential equations.* Ph. D. thesis, Univ. of Amsterdam, Amsterdam.

Zegeling, P. A., 1993. Moving-grid methods for time-dependent partial differential equations, *CWI Tract No. 94.* Centre for Mathematics and Computing Science, Amsterdam.

Zegeling, P.A. 1998. R-refinement for evolutionary PDEs with finite elements or finite differences. *Applied Numerical Mathematics* 26: 97-104.

Zaidel, J., and D. Russo, 1992. Estimation of finite difference interblock conductivities for simulation of infiltration into initially dry soils. *Water Resources Research* 28: 2285-2295.

Zhang, H.L., A. Rottgermann, and S. Wagner, 1995. Field panel method with grid stretching technique for solving transonic potential flow around arbitrary airfoils, *Computational Mechanics* 15: 384-393.