Finite Difference Methods for Advection in Variable-Velocity Fields

David Scullen

Honours Thesis
1992

The University of Adelaide
Department of Applied Mathematics

I wish to thank Dr. John Noye for his supervision and assistance, and the many things that I have learned from him. In particular, I would like to thank him for his continual encouragement throughout the year.

I would also like to thank Jody Nicholls for typing the bulk of this text, and Shane Richards for his assistance with the LaTeX diagrams.

Thanks also to the staff and students of the Applied Mathematics department for their assistance and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 The Advection-Diffusion-Mortality Equation

1.1.1 What is the advection-diffusion-mortality equation?

The advection-diffusion-mortality (ADM) equation is a partial differential equation (PDE) that describes the spreading of material subject to advection, diffusion and mortality (or decay), usually when the substance is dissolved or suspended in a fluid.

Advection is the term used to describe how the substance is carried along by the fluid. For example oil is carried on the surface of sea-water due to the currents produced by winds and tides. This motion is sometimes also called convection.

Diffusion is the spreading of a substance due to random molecular collisions, or due to larger forces such as vorticity or turbulence; for example an oil slick on the sea surface spreads due to the turbulent action of the sea.

Mortality or decay describes the way in which a substance can be depleted, for example oil in a slick can coagulate and settle on the sea floor or may evaporate from the surface.

So in this way, the ADM equation can be used to model the fate of many situations involving transport and decay.

The ADM equation in one-dimensional space is the PDE:

$$\frac{\partial \hat{\tau}}{\partial t} + u\frac{\partial \hat{\tau}}{\partial x} - \alpha_x \frac{\partial^2 \hat{\tau}}{\partial x^2} + \lambda \hat{\tau} = f\left(x,t\right) \tag{1.1}$$

where $\hat{\tau}\left(x,t\right)$ is the concentration in the fluid of the substance in which we are interested, u is the fluid velocity in the x-direction, α_x is the diffusion coefficient, λ is the coefficient of decay, and $f(x,t)$ is a source or sink term.

In the above equations the term

$$\frac{\partial \hat{\tau}}{\partial t} \tag{1.2}$$

represents the time dependence of the concentration $\hat{\tau}$,

$$u\frac{\partial\hat{\tau}}{\partial x} \tag{1.3}$$

represents the advection,

$$-\alpha_x\frac{\partial^2\hat{\tau}}{\partial x^2} \tag{1.4}$$

represents diffusion, and

$$\lambda\hat{\tau} \tag{1.5}$$

represents decay or mortality.

This one-dimensional equation can easily be expanded to two-dimensional or three-dimensional space. For example in three-dimensional space it takes the form

$$\frac{\partial\hat{\tau}}{\partial t}+u\frac{\partial\hat{\tau}}{\partial x}+v\frac{\partial\hat{\tau}}{\partial y}+w\frac{\partial\hat{\tau}}{\partial z}-\alpha_x\frac{\partial^2\hat{\tau}}{\partial x^2}-\alpha_y\frac{\partial^2\hat{\tau}}{\partial y^2}-\alpha_z\frac{\partial^2\hat{\tau}}{\partial z^2}+\lambda\hat{\tau}=f\left(x,y,z,t\right) \tag{1.6}$$

where $\hat{\tau}(x,y,z,t)$ is now a function of x, y, z and t, v is the fluid velocity in the y-direction, w is the fluid velocity in the z-direction, α_y is the y component of diffusion, α_z is the z component of diffusion and all other terms are the same as they were for the one-dimensional situation.

1.1.2 What real life applications may be modelled using the advection-diffusion-mortality equation?

The questions should be asked: "Of what importance is the ADM equation?" and "Why should this equation be investigated?"

The answers are that the variety of applications which can be modelled by the ADM equation make it an important equation, and that the requirement for solutions to these problems justifies its study.

Real world applications of the ADM equation are abundant. They arise in oceanography, problems associated with rivers and estuaries, groundwater, the atmosphere and even in areas that would at first glance be considered unlikely, like physics and botany.

Oceanography

Naturally, oceanography is a field in which the ADM equation can be of great use.

Eurospill, a model for oil and chemical spills at sea has been developed as a user-friendly package that is designed to give rapid predictions for the fate of a spill, or for risk analysis, contingency planning and training. The processes which the program simulates are spreading, dispersion, reaction, evaporation and solution (Lunel, 1991). Spreading is due to advection, dispersion is due to diffusion, while reaction, evaporation and solution can be collectively called decay. The need to accurately predict the fate of a spill is obvious.

The ADM is also of importance in marine biology. Marine aquaculture can be seen as an enterprise that can contribute directly to socio-economic maturity (Harnett and Cawley, 1991). Water quality is an issue that must be addressed in fish farming, since fish survival is dependent upon the water being relatively unpolluted. The dispersion of wastes (ammoniacal nitrogen, suspended solids and other effluent) from the farm can be modelled by ADM, with the currents being determined from tide and wind conditions.

Similarly, the population dynamics of some marine organisms can be modelled using ADM (Possingham and Roughgarden, 1990). This applies particularly to organisms that have limited means of self locomotion such as pelagic larvae. The location of the organism is determined by the currents and eddy-diffusion. The numbers are also influenced by factors of mortality such as predation, or the availability of a suitable habitat (in the case of a settled organism).

Coastal engineering works include investigations of currents on coastal and estuarine morphology, siltation studies of harbours and reclamation projects. Cheong et al. (1992) investigate the movement of a silt slurry piped from quarries and discharged off the coast of Singapore due to the necessity of dredging to maintain the seaways. This is essentially an advection-diffusion problem.

Rivers and estuaries

Likewise, rivers and estuary problems are often solved using the ADM equation. The fate of pollutants introduced to river systems is an obvious application. Roldão et al. (1991) developed a computer system for modelling water quality with particular reference to the Paraíba do Sul river in Brazil. This river is the most important water supply for Rio de Janeiro. Of particular interest was the reach down stream of the industrial city of Volta Redonda due to the presence of a steelworks which produces ammonia, phenol and cyanide which are sometimes accidentally discharged into the river. The aim was to determine in real time if pollutant concentrations would rise above permitted levels as the result of an accident.

Salt water intrusion in estuaries (Savenije, 1989) is of interest, in particular when the estuary is used as water resource. The way in which saltwater flows upstream as the tide comes in is governed by the processes of advection and diffusion. The currents of interest are those due to the fresh water discharge of the stream, and the tidal velocity. Note that the velocity of the currents here may be considered to be sinusoidal in time due to the tides, that is the velocity of the fluid is the sum of a constant value (the fresh water discharge) and an oscillating component (the tide). The spreading of the salt in solution then becomes a problem in one-dimensional advection-diffusion.

Also, Bencala et al. (1990) studied the hydrology of the Snake River in Colorado due to the environmental impact of acid mine drainage and acid rain.

Groundwater

The ADM equation is used to assist in the safe disposal of nuclear wastes. Emplacement in deep geologic formations is a realistic and reliable solution to nuclear waste disposal. Patyn et al. (1989) used ADM to evaluate the ability of Boom clay formations in North Eastern Belgium to store the waste produced by Belgium reactors. They state that the processes by which the radionuclides contained in wastes could return to the human environment are largely determined by the hydrogeological situation. They considered the investigation of aquifers adjacent to potential host formations to be an important part of safety analysis.

Abelink and Wheater (1990) state that the need to predict the movement of pesticides and fertilizers through soils to groundwater requires that the processes of advection, diffusion and mortality be studied.

Contractor and El-Didy (1989) discuss the effect of underground coal gasification on water quality. If underground coal gasification is to be adopted as a viable method of energy production, it must be seen to be environmentally safe. "During the gasification process, the water in the cavity is converted to steam and the high pressure that is maintained in the cavity drives water and gas back into the aquifer... Upon completion of the burn, the pressure in the cavity will be atmospheric and the water in the aquifer begins to seep back into the burn cavity which will now contain ash and rubble... The cavity fills up and becomes part of the confined aquifer system... A time will be reached when the [contaminants will be] dispersed and sorbed in the aquifer." This describes the process by which the contaminants are introduced into the groundwater system.

Miscellaneous

The ADM equation also has application in areas that are not so obvious.

For example Calder (1991) used the advection-diffusion equation when studying transpiration in trees. He uses the equation to derive two parameters, dispersivity and effective porosity that are of value in designing tracer experiments and may be useful in understanding flows of water as part of the transpiration process in trees.

Game and Stephenson (1979) discuss forced cooling of solid material in which heat is being generated, for example in stator and rotor windings in turbogenerators and high voltage underground cables. This is achieved by passing fluid through a duct within the solid.

In all of these examples either the ADM equation or advection-diffusion equation have actively modelled the situation — and we have only considered a few examples. So the importance of this equation is clear.

With so many applications of interest, it is imperative that accurate, stable and efficient methods for solving the ADM equation be found.

1.1.3 Techniques for solving the advection-diffusion-mortality equation

Various techniques have been used to solve the ADM equation. They include analytic techniques, finite-element methods, finite-difference methods (FDMs) and several other less commonly used techniques such as boundary element methods.

Analytic techniques

Analytic technique is often used for solving the ADM equation in problems associated with soil science and flow through porous media. Some of the important methods include separation of variables, complex variable techniques, Fourier and Laplace transformations, Green's functions and power series (Huyakorn et al. 1983).

Knopman and Voss (1988) use analytic techniques to determine exact solutions to their model of solute transport in a porous medium in one dimension. They have been used to solve the equation produced when modelling the transport of dissolved substances with second order reaction (Yates and Enfield, 1989), the fate of chemicals in aquifers (Lindstrom and Boersma, 1989), radial dispersion in a variable velocity flow field (Yates, 1988), hydrodynamic dispersion in bounded media (Batu, 1989), contaminant transport in a single fracture (Lowell, 1989), and contaminant transport from sources in saturated porous media (Dillon, 1989).

Analytic solutions are exact throughout the solution domain. However, they are specific to a given problem and often are only solutions to an approximation of the real life problems being considered. For instance, non-linear terms are often omitted, variable coefficients are generally taken to the constant, and the shape of the boundary and the nature of the boundary conditions are usually greatly simplified as solutions are difficult to obtain for problems with regions of irregular boundaries.

Finite-element methods

Another popular means for solving the ADM equation is the use of finite-element methods.

Finite-element methods have been used to determine numeric solutions when modelling water quality in a coal seam (Contractor and El-Didy, 1989), subsurface transport of volatile organic compounds (Culver et al., 1991), nitrogen transport in groundwater (Kaluarachchi and Parker, 1988), and multicomponent transport with microbial transformation in groundwater (Frind et al., 1990).

It has also been used extensively in problems involving advection and diffusion such as solute segregation during freezing of peatland waters (Kadlec et al., 1988), and groundwater flow and solute transport (Chiang et al., 1989, Herbert et al., 1988, Jinzhong, 1988, Leijtalk and Dane, 1991, Putti et al., 1990).

Finite-difference methods

Another useful technique is the FDM. These are generally robust and easy to program. Their merits will be discussed in greater detail in Section 1.2.1.

Bencala et al. (1990) used FDMs to solve the ADM equation when modelling solute transport in the Snake River, Colorado. FDMs have been used for the problems of solute transport (Cox and Nishikawa, 1991), groundwater flow (Fio and Deverel, 1991) when solving the Navier-Stoker and energy equations (Macarthur and Patankar, 1989), predicting the structure of tidal current and salinity in San Francisco Bay, California (Ford et al., 1990), brine transport (Hassanizadeh and Leijnse, 1988), the estimation of groundwater recharge (Solomon and Sudicky, 1991) and solute transport in the soil layer of a hill slope (Kinouchi et al., 1991).

Other techniques that have been used to solve the ADM equation include the least squares optimisation method (Bensabat and Zeitoun, 1990) optimal test functions (Kindred and Celia, 1989 and Celia et al., 1989) and the discrete kernel approach (Illangasekare and Döll, 1989).

So a wide variety of techniques are used to solve the ADM equation. Analytic solution, although exact, can be difficult to use for complex problems. Numerical methods (for example finite-element methods and finite-difference methods) do not produce exact solutions, but in many cases are highly accurate. They are also more easily adapted to different problems of a more complicated nature.

It is for this reason that the remainder of this thesis will discuss the use of FDMs in solving the ADM equation.

1.2 Finite-Difference Methods

1.2.1 Why are finite-difference methods used to solve the advection-diffusion-mortality equation?

FDMs are easy to understand, are reasonably simple in their concepts and so are simple to apply. (This is not to say that the search and application of a good FDM for an accurate and speedy solution is a simple task.) Like all numerical methods, FDMs are almost arbitrarily accurate — their accuracy is limited only by the number of decimal figures that the computer can store for a given value, and the speed at which they can be computed. They are robust, that is the results produced can by relied upon to be correct, provided the particular FDf being used is well understood (stability, peak shift and attenuation are key factors in fidelity of results). They are easily coded for — they are an iterative technique and require only a basic understanding of any computer language. And finally, much work has been devoted to the practise of developing and applying FDMs to a variety of problems.

One particularly attractive point is that a complicated problem (such as that of ADM) can be broken up into its constituents (advection, diffusion, and mortality) and solved sequentially, rather than having to deal with the entire problem simultaneously. Indeed, any one of

these constituents (for example advection) can be broken down further (to the x, y and z components say) so that the problem can be further simplified. The method that is referred to here is that of time splitting or fractional steps (D'Yankonov 1963). It will be discussed in greater detail in Chapter 5, but for now it to note that to solve the ADM equation it is necessary only to develop a FDM for each of the three constituents and then to combine them in application.

1.2.2 What is a finite-difference method?

FDMs are techniques for solving PDEs within a spatial region subject to initial value conditions in the case of time-dependent problems, and boundary conditions, by approximating the unknown value of the function at some time in the near future. This approximation is then used to determine the approximate value of the function at some later time. The process is repeated until an approximation for the exact value of the function at the desired time is determined.

Hence FDMs are an iterative procedure, with each step introducing only a small error. The final result, the value of the function everywhere within the region at the desired time is subject to these errors, and as such is only an approximation. However this approximation can be made as accurate as is required by making the steps small enough and is limited only by the computer used to solve the problem.

A finite-difference formula (FDf) is an equation that approximates the value of the function at a point in terms of points nearby in space and time.

To explain this, consider the non-dimensionalised function $\hat{\tau}(x,t)$ in which the parameters of the problem have been scaled so that $0 \leq x \leq 1$, $0 \leq t \leq T$ (T is order 1) and $\hat{\tau}$ has values typically around 1. For the one-dimensional advection equation we have

$$\frac{\partial \hat{\tau}}{\partial t} + u\frac{\partial \hat{\tau}}{\partial x} = 0 \tag{1.7}$$

with initial condition

$$\hat{\tau}(x,0) = f(x), \qquad 0 \leq x \leq 1, \tag{1.8}$$

and boundary condition

$$\hat{\tau}(0,t) = g_L(t), \qquad 0 \leq t \leq T. \tag{1.9}$$

We are interested in the values of $\hat{\tau}(x,T)$, the function at some time T.

We divide the solution domain $0 < x < 1$, $0 < t \leq T$ into a grid with spacings of Δx in the x-direction and Δt in the t-direction (see Figure 1.1). Grid lines parallel to the t-axis are given by

$$x = x_j = j\Delta x, \qquad j = 1\,(1)\,J-1, \tag{1.10}$$

where $\Delta x = 1/J$, and are labeled by the index j. Grid lines parallel to the x-axis are given by

$$t = t_n = n\Delta t, \qquad n = 1\,(1)\,N, \tag{1.11}$$

where $\Delta t = T/N$, and are labeled by the index n.

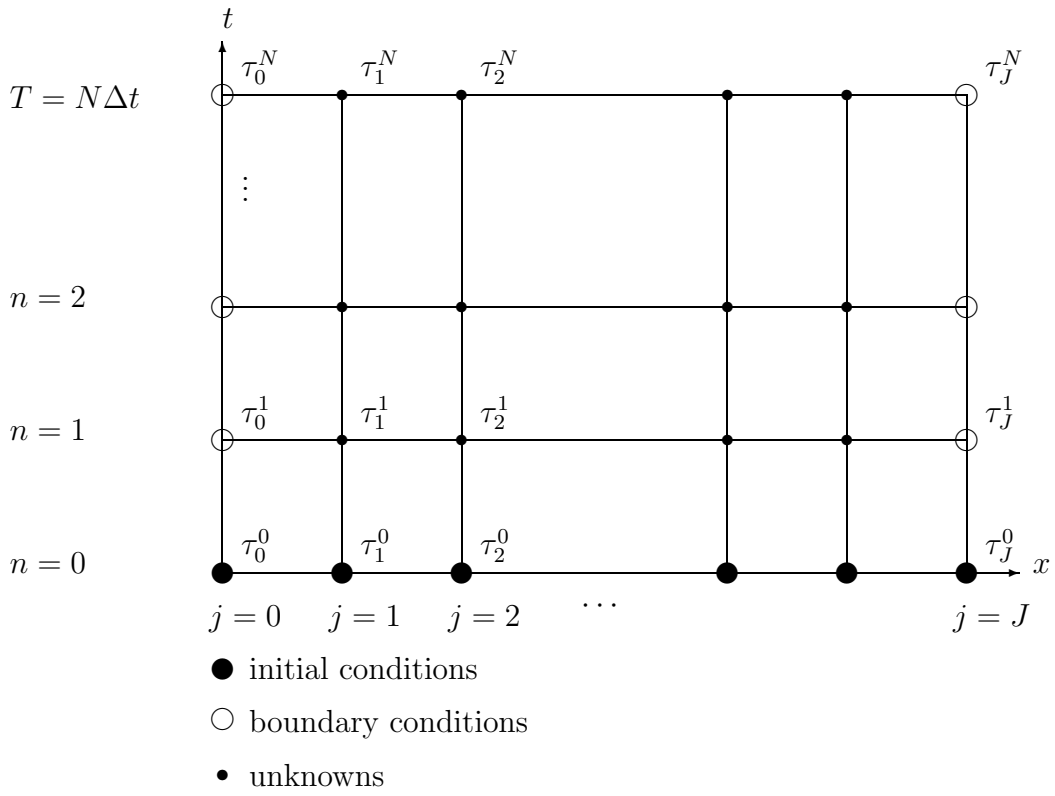

Figure 1.1: The one-dimensional FDM grid

The symbol τ represents the solution to the FDf and is an approximation to the function $\hat{\tau}$.

We will evaluate τ at each of the points at which these grid lines intersect. We refer to the point (x_j, t_n) that has position $(j\Delta x, n\Delta t)$ to be the $(j, n)^{th}$ grid point.

In this way the FDf describes the value of the function at a point by the value at points that are nearby on the grid. An example FDf is

$$\tau_j^{n+1} = a\tau_{j-1}^n + b\tau_j^n + c\tau_{j+1}^n \tag{1.12}$$

where a, b and c are known. In the above notation the superscript, n, is the time index, which we shall call the time level. The subscript, j, represents the spatial position in the x-direction, and we shall call this the "position". So τ_j^n represents the value of the function τ at the n^{th} time level at the j^{th} position.

Using the FDf it is possible to evaluate the function τ at each grid point at the $(n+1)^{th}$ time level if the values at the n^{th} time level are known. These new values can be used to determine τ at the $(n+2)^{th}$ time level and so on, until eventually the values at the N^{th} time level are known.

So, given an initial condition and a boundary condition, FDMs can be used to determine the value of a function at a later time T which can be arbitrarily chosen.

1.2.3 Where do finite-difference formulas come from?

FDfs are derived from the governing PDE using Taylor series expansions about each of the points in the FDf. Consider the one-dimensional advection equation

$$\frac{\partial \hat{\tau}}{\partial t} + u\frac{\partial \hat{\tau}}{\partial x} = 0, \tag{1.13}$$

which is a special case of the one-dimensional ADM equation with the coefficients of diffusion and decay set to zero.

The Taylor series of $f(t + \Delta t)$ about t is

$$f\left(t + \Delta t\right) = f\left(t\right) + \Delta t f'\left(t\right) + O\{(\Delta t)^2\}, \tag{1.14}$$

so rearranging we obtain

$$f'\left(t\right) \approx \frac{f\left(t + \Delta t\right) - f\left(t\right)}{\Delta t}. \tag{1.15}$$

So we can approximate $\partial\hat{\tau}/\partial t|_j^n$ by

$$\left.\frac{\partial \hat{\tau}}{\partial t}\right|_j^n \approx \frac{\tau_j^{n+1} - \tau_j^n}{\Delta t}, \tag{1.16}$$

where $\partial\hat{\tau}/\partial t|_j^n$ represents $\partial\hat{\tau}/\partial t$ evaluated at the $(j, n)^{th}$ grid point.

Similarly, expansion of the Taylor series about the grid points $(j-1, n)$, (j, n) and $(j+1, n)$ shows that

$$g'\left(x\right) = \frac{g\left(x + \Delta x\right) - g\left(x - \Delta x\right)}{2\Delta x} + 0\{(\Delta x)^2\}, \tag{1.17}$$

and so we could write:

$$\left.\frac{\partial \hat{\tau}}{\partial x}\right|_j^n \approx \frac{\tau_{j+1}^n - \tau_{j-1}^n}{2\Delta x}. \tag{1.18}$$

We could then substitute the two approximations into Equation (1.13) to produce the FDf

$$\frac{\tau_j^{n+1} - \tau_j^n}{\Delta t} + u\left[\frac{\tau_{j+1}^n - \tau_{j-1}^n}{2\Delta x}\right] = 0, \tag{1.19}$$

or upon rearranging

$$\tau_j^{n+1} = \frac{c}{2}\tau_{j-1}^n + \tau_j^n - \frac{c}{2}\tau_{j+1}^n \tag{1.20}$$

where c is called the Courant number defined by

$$c = u\frac{\Delta t}{\Delta x}. \tag{1.21}$$

The FDf (1.20) is used in the FDM to determine τ at the new time level. A comparison shows that Equation (1.20) has the same form as Equation (1.12).

A FDf is said to be consistent with a PDE if the discretisation error (that is, the error introduced by the approximations when truncating the Taylor series) tends to zero as the

grid spacings Δx and Δt tend to zero. This will be discussed further in Section 2.1. This FDf is consistent with the one-dimensional advection-diffusion PDE.

However, it is important to note that the discretisation error is not the only source of error in FDMs. In particular, errors can be introduced by calculations done by the computer. These errors are due to round-off, as the computer only stores a limited number of significant figures in its calculations. If these errors grow as the FDf is iterated, the results will soon be dominated by these errors. This phenomenon is called instability, and will be discussed in detail in Section 2.3. For now, it is necessary only to note that the above FDf (1.20) is actually unstable and is of no use in practical application. As we shall see later it is important that any FDf developed by stable, as well as consistent.

1.3 Why Concentrate on Advection?

The content of this thesis is the study of the advection equation in a velocity field that varies in space or time, or both.

We have seen the importance of the ADM equation, and so accept the need to fully understand how to apply FDMs to its solution. Time splitting enables us to break this complex problem into three smaller problems: advection, diffusion and mortality.

Mortality is easily solved, even through analytic techniques, if the mortality coefficient λ is constant. It can be readily shown that if $\hat{\tau}(x,t)$ is a solution of the one-dimensional advection-diffusion equation then $e^{-\lambda t}\hat{\tau}(x,t)$ is a solution to the one-dimensional ADM equation. That is, if

$$\frac{\partial \hat{\tau}}{\partial t} + u\frac{\partial \hat{\tau}}{\partial x} - \alpha_x \frac{\partial^2 \hat{\tau}}{\partial x^2} = 0 \tag{1.22}$$

then, for constant λ,

$$\frac{\partial \hat{\tau}}{\partial t}\left(e^{-\lambda t}\hat{\tau}\right) + u\frac{\partial}{\partial x}\left(e^{-\lambda t}\hat{\tau}\right) - \alpha_x \frac{\partial^2}{\partial x^2}\left(e^{-\lambda t}\hat{\tau}\right) + \lambda\left(e^{-\lambda t}\hat{\tau}\right) = 0. \tag{1.23}$$

It is important to note that although we would often expect λ to be variable, it is difficult in most applications to determine λ as a function of space and time. Hence in most solutions to problems λ is assumed to be constant. For those cases where λ does vary, one could use existing packages which solve ordinary differential equations to cover this aspect of the problem, since at each grid point the mortality equation involves only a time-derivative.

This idea can be expanded to three dimensions as required. Indeed the solution of the mortality component is comparatively straight forward.

Diffusion is well understood. Much work has been done on solving the diffusion equation and accurate robust FDfs have been developed. Again, the diffusion coefficient α is often taken to be constant for the same reasons stated above for mortality.

In most applications it is necessary to consider the fact that the fluid velocities, u, v, and w are not constant. For example, currents due to tides are not constant as is also the velocity of a river of varying cross section. Also, these velocity components can often be measured

or calculated. So we would be restricting ourselves to a very limited class of problems if we were to to constrain these velocities to be constant. Hence it is necessary to understand the advection equation for variable velocity fields, and to have good methods for its solution available.

1.4 Summary

We have seen that the ADM equation has many varied and important applications, and as such is an equation that is of great interest. It is important to have good methods of solving problems in ADM, and FDMs are an attractive numerical technique as they are reasonably easy to understand, and are easy to program.

One particularly attractive aspect is that we may simplify complex problems by breaking them up into smaller problems. This is done by the technique called time splitting.

So the complex problem of three-dimensional ADM reduces to three simpler problems, each in one dimension. They are the separate problems of advection, diffusion and mortality.

The problem of mortality is relatively straight forward to handle. Diffusion has been well studied and good methods of solution are readily available. The coefficients for these problems are often considered to be constant.

Velocity components of currents are usually not constant, and can often be obtained (either through experimental measurement or through calculations), so it necessary to consider the advection equation in varying velocity fields.

It is to this task that this thesis is devoted.

Chapter 2

Development of a Higher-Order Finite-Difference Method for One-Dimensional Advection

2.1 The Order of Accuracy of a Finite Difference Method

One of the most important aspects to consider when using, developing or investigating a FDM is the accuracy of the results that it produces.

The differences between the solution to the governing PDE and the exact solution of the FDf is called the discretisation error. It is denoted e, where

$$e = \hat{\tau} - \tau. \tag{2.1}$$

It can be produced from either of two sources.

The first is the error introduced when approximating the PDE by the FDf. This error is called the truncation error and will be discussed in detail here.

The second source is the error introduced by the calculation done by the computer, and is called the "round-off" error. This error will be discussed in detail later in Section 2.3, in the context of stability of the FDf.

The truncation error is denoted $E\{\hat{\tau}\}$ and is the error introduced when approximating the PDE evaluated at the point $j\Delta x$ at the time $n\Delta t$, by the FDf.

It can be determined by substituting the exact solution to the PDE for each term in the FDf, and then taking the Taylor series expansion of each of these terms about the point $(j\Delta x, n\Delta t)$. If we consider Equation (1.20) written in the form

$$\mathcal{L}_\Delta \tau_j^n = \tau_j^{n+1} - \frac{c}{2}\tau_{j-1}^n - \tau_j^n + \frac{c}{2}\tau_{j+1}^n = 0, \tag{2.2}$$

then replacing τ_j^n by the exact solution $\hat{\tau}_j^n$ of the advection equation gives

$$\mathcal{L}_\Delta \left\{\hat{\tau}_j^n\right\} = \hat{\tau}_j^{n+1} - \hat{\tau}_j^n - \frac{c}{2}\left[\hat{\tau}_{j-1}^n - \hat{\tau}_{j+1}^n\right], \tag{2.3}$$

which will not be equal to zero.

Taking Taylor expansions about the point $(j\Delta x, n\Delta t)$ and collecting like terms gives

$$\mathcal{L}_\Delta \left\{\hat{\tau}_j^n\right\} = \Delta t \left[\frac{\partial \hat{\tau}}{\partial t} + u\frac{\partial \hat{\tau}}{\partial x} + \frac{\Delta t}{2}\frac{\partial^2 \hat{\tau}}{\partial t^2} + u\frac{(\Delta x)^2}{6}\frac{\partial^3 \hat{\tau}}{\partial x^3} + O\left\{(\Delta t)^2, (\Delta x)^4\right\}\right]_j^n, \tag{2.4}$$

which differs from the one-dimensional advection equation by the additional terms

$$E\left\{\hat{\tau}\right\} = -\frac{\Delta t}{2}\frac{\partial^2 \hat{\tau}}{\partial t^2}\bigg|_j^n - u\frac{(\Delta x)^2}{6}\frac{\partial^3 \hat{\tau}}{\partial x^3}\bigg|_j^n + O\left\{(\Delta t)^2, (\Delta x)^4\right\}, \tag{2.5}$$

which is the truncation error.

From the largest terms in the truncation error, the FDf is said to be of order

$$O\left\{(\Delta x)^2, \Delta t\right\} \tag{2.6}$$

in accuracy.

As noted in Section 1.2.3, this truncation error tends to zero as Δt and Δx tend to zero. So the FDf (1.20) is consistent with the one-dimensional advection equation.

2.2 Improving on an Existing Finite Difference Formula

One method of improving the accuracy of a finite-difference method is to remove the lower order error terms by replacing them by appropriate differencings, thereby increasing the order of accuracy of the method.

As was seen in Section 1.2.3, the PDE evaluated at the point (x_j, t_n) is approximated by a finite-difference function which relates the unknown value of the function at a point to the known values at points nearby. Each of these points can be described by their Taylor series expansions and when multiplied by their appropriate coefficient will sum to an approximate value of the PDE evaluated at the unknown point. (It is helpful to use a symbolic manipulator to achieve this. The author used the Maple package for this purpose.) The order of magnitude of errors associated with this approximation determine the order of accuracy of the method. To improve the order of accuracy of the method it is necessary to remove the lowest-order error terms, that is, the largest errors in the approximation.

This is achieved by incorporating in the finite-difference formula the finite difference forms of the desired accuracy that are the equivalent of these lower-order terms.

This then produces a finite-difference formula of higher accuracy which may (subject to restrictions of stability which will be discussed later) be used to approximate the PDE.

This process was completed for several cases, one of which was the improvement on Martin's method that is detailed below.

Martin had developed the FDf which was third order accurate for the case where the velocity of the fluid, u, is constant and is of order one for the generalised non-constant velocity case. His formula was :

$$\begin{aligned}\tau_j^{n+1} - \Big[\quad & \tau_{j+1}^n \left(-\tfrac{1}{3}c + \tfrac{1}{2}c^2 - \tfrac{1}{6}c^3\right) \\ & +\tau_j^n \left(1 - \tfrac{1}{2}c - c^2 + \tfrac{1}{2}c^3\right) \\ & +\tau_{j-1}^n \left(c + \tfrac{1}{2}c^2 - \tfrac{1}{2}c^3\right) \\ & +\tau_{j-2}^n \left(-\tfrac{1}{6}c + \tfrac{1}{6}c^3\right) \Big] = 0,\end{aligned} \tag{2.7}$$

where c is defined by Equation (1.21).

Substitution of the exact solution to the advection equation for the solution to the FDf and then expanding the Taylor series about the point $(j\Delta x, n\Delta t)$ as in the previous section, leads to the determination of the truncation error

$$E\left\{\hat{\tau}\right\} = \frac{1}{2}\Delta t \left(\frac{\partial^2 \hat{\tau}}{\partial t^2} - u^2 \frac{\partial^2 \hat{\tau}}{\partial x^2}\right) + O\left\{(\Delta t)^2\right\}, \tag{2.8}$$

which has higher-order derivatives with respect to time than $\partial\hat{\tau}/\partial t$ (that is $\partial^2\hat{\tau}/\partial t^2$, $\partial^3\hat{\tau}/\partial t^3$ etc.)

We note that

$$\frac{\partial \hat{\tau}}{\partial t} + u\frac{\partial \hat{\tau}}{\partial x} = 0 \quad \Rightarrow \quad \frac{\partial \hat{\tau}}{\partial t} = -u\frac{\partial \hat{\tau}}{\partial x}, \tag{2.9}$$

and differentiating with respect to time yields, for varying velocity u,

$$\frac{\partial^2 \hat{\tau}}{\partial t^2} = -\left(\frac{\partial u}{\partial t} - u\frac{\partial u}{\partial x}\right)\frac{\partial \hat{\tau}}{\partial x} + u^2 \frac{\partial^2 \hat{\tau}}{\partial x^2}. \tag{2.10}$$

Similarly, higher-order derivative involving $\partial/\partial t$ may be replaced, so that the modified discretised PDE (MDPDE) obtained is

$$\begin{aligned}\Bigg[& \tfrac{\partial \hat{\tau}}{\partial t} + u\left(x,t\right)\tfrac{\partial \hat{\tau}}{\partial x} \\ & + \left\{-\tfrac{\Delta t}{2}\left(\tfrac{\partial u}{\partial t} - u\tfrac{\partial u}{\partial x}\right) - \tfrac{(\Delta t)^2}{6}\left(\tfrac{\partial^2 u}{\partial t^2} + u\left(\tfrac{\partial u}{\partial x}\right)^2 - u\tfrac{\partial^2 u}{\partial x \partial t} + u^2\tfrac{\partial^2 u}{\partial x^2} - 2\tfrac{\partial u}{\partial t}\tfrac{\partial u}{\partial x}\right)\right\}\tfrac{\partial \hat{\tau}}{\partial x} \\ & + u\tfrac{(\Delta t)^2}{2}\left(\tfrac{\partial u}{\partial t} - u\tfrac{\partial u}{\partial x}\right)\tfrac{\partial^2 \hat{\tau}}{\partial x^2} \\ & + O\left\{(\Delta t)^3\right\}\Bigg]_j^n = 0.\end{aligned} \tag{2.11}$$

This does not contain any time derivatives of $\hat{\tau}$ except $\partial\hat{\tau}/\partial t$. The largest truncation error is of order Δt, and so (2.7) is an $O\{1\}$ approximation to the advection-equation for non-constant velocity fields. Note that if u is constant then the method is $O\{3\}$.

This is an important point. A method may be of high-order accuracy for constant velocity, u, but will usually deteriorate to first order accuracy for varying velocity u.

It is therefore highly desirable to develop methods that are of higher-order accuracy for varying velocity fields.

This can be done by eliminating the largest error terms in the MDPDE (2.11).

Section 1.2.3 showed that derivatives can be approximated by finite-difference forms that are derived from Taylor series expansion. It is not intended here to derive the forms that are used in this document, but merely to define them as they are introduced. It is worthwhile to note, however, that the finite-difference forms used in developing a FDf will largely determine the performance of the method.

Using centred-space differencing for the first derivative

$$\left.\frac{\partial\hat{\tau}}{\partial x}\right|_j^n = \frac{\hat{\tau}_{j+1}^n - \hat{\tau}_{j-1}^n}{2\Delta x} + O\left\{(\Delta x)^2\right\}, \tag{2.12}$$

and defining

$$d_j^n = \frac{(\Delta t)^2}{2\Delta x}\left(\frac{\partial u}{\partial t} - u\frac{\partial u}{\partial x}\right)\Bigg|_j^n, \tag{2.13}$$

we can replace the largest error in (2.11) by the expression

$$-\frac{d_j^n}{2}\left(\hat{\tau}_{j+1}^n - \hat{\tau}_{j-1}^n\right). \tag{2.14}$$

So, subtracting this from the FDf would remove the largest error from the MDPDE, thus producing the more accurate FDf:

$$\begin{aligned}\tau_j^{n+1} - \Big[\quad & \tau_{j+1}^n\left(-\tfrac{1}{3}c_j^n + \tfrac{1}{2}(c_j^n)^2 - \tfrac{1}{6}(c_j^n)^3 - \tfrac{1}{2}d_j^n\right) \\ & +\tau_j^n\left(1 - \tfrac{1}{2}c_j^n - (c_j^n)^2 + \tfrac{1}{2}(c_j^n)^3\right) \\ & +\tau_{j-1}^n\left(c_j^n + \tfrac{1}{2}(c_j^n)^2 - \tfrac{1}{2}(c_j^n)^3 + \tfrac{1}{2}d_j^n\right) \\ & +\tau_{j-2}^n\left(-\tfrac{1}{6}c_j^n + \tfrac{1}{6}(c_j^n)^3\right)\Big] = 0\end{aligned} \tag{2.15}$$

which is of order two for non-constant velocity, and will have the MDPDE

$$\begin{aligned}&\left[\frac{\partial\hat{\tau}}{\partial t}+u\left(x,t\right)\frac{\partial\hat{\tau}}{\partial x}\right.\\&-\frac{\left(\Delta t\right)^{2}}{6}\left(\frac{\partial^{2}u}{\partial t^{2}}+u\left(\frac{\partial u}{\partial x}\right)^{2}-u\frac{\partial^{2}u}{\partial x\partial t}+u^{2}\frac{\partial^{2}u}{\partial x^{2}}-2\frac{\partial u}{\partial t}\frac{\partial u}{\partial x}\right)\frac{\partial\hat{\tau}}{\partial x}\\&+u\frac{\left(\Delta t\right)^{2}}{2}\left(\frac{\partial u}{\partial t}-u\frac{\partial u}{\partial x}\right)\frac{\partial^{2}\hat{\tau}}{\partial x^{2}}\\&\left.+O\left\{\left(\Delta t\right)^{3}\right\}\right]_{j}^{n}=0.\end{aligned}\tag{2.16}$$

This method has second order accuracy, so we will denote this Martins $O\left\{2\right\}$ method.

Similarly we can remove the error terms of order two.

If we define

$$h_{j}^{n}=\frac{\left(\Delta t\right)^{3}}{6\Delta x}\left[\frac{\partial^{2}u}{\partial t^{2}}+u\left(\frac{\partial u}{\partial x}\right)^{2}-u\frac{\partial^{2}u}{\partial x\partial t}+u^{2}\frac{\partial^{2}u}{\partial x^{2}}-2\frac{\partial u}{\partial t}\frac{\partial u}{\partial x}\right]_{j}^{n}\tag{2.17}$$

and note that

$$c_{j}^{n}d_{j}^{n}\left(\Delta x\right)^{2}=u\frac{\left(\Delta t\right)^{3}}{2}\left(\frac{\partial u}{\partial t}-u\frac{\partial u}{\partial x}\right)\Bigg|_{j}^{n}\tag{2.18}$$

we see the MDPDE of Martins FDf is

$$\begin{aligned}&\left[\frac{\partial\hat{\tau}}{\partial t}+u\left(x,t\right)\frac{\partial\hat{\tau}}{\partial x}+\left\{-d_{j}^{n}-h_{j}^{n}\right\}\Delta x\frac{\partial\hat{\tau}}{\partial x}\right.\\&\left.+c_{j}^{n}d_{j}^{n}\left(\Delta x\right)^{2}\frac{\partial^{2}\hat{\tau}}{\partial x^{2}}+O\left\{\left(\Delta t\right)^{3}\right\}\right]_{j}^{n}=0,\end{aligned}\tag{2.19}$$

evaluated at the point$(j\Delta x,\ n\Delta t)$.

Using first-order upwind differencing for the first spatial derivative, namely,

$$\frac{\partial\hat{\tau}}{\partial x}\Bigg|_{j}^{n}=\frac{\hat{\tau}_{j}^{n}-\hat{\tau}_{j-1}^{n}}{\Delta x}+O\left\{\Delta x\right\},\tag{2.20}$$

and second-order centred-space differencing for the second derivative, namely,

$$\frac{\partial^{2}\hat{\tau}}{\partial x^{2}}\Bigg|_{j}^{n}=\frac{\hat{\tau}_{j+1}^{n}-2\hat{\tau}_{j}^{n}+\hat{\tau}_{j-1}^{n}}{\left(\Delta x\right)^{2}}+O\left\{\left(\Delta x\right)^{2}\right\},\tag{2.21}$$

we can replace the truncation errors by

$$\left(-d_{j}^{n}-h_{j}^{n}\right)\left(\hat{\tau}_{j}^{n}-\hat{\tau}_{j-1}^{n}\right)+c_{j}^{n}d_{j}^{n}\left(\hat{\tau}_{j+1}^{n}-2\hat{\tau}_{j}^{n}+\hat{\tau}_{j-1}^{n}\right),\tag{2.22}$$

and so subtracting this from Martin's formula we get the FDf

$$\begin{aligned}\tau_j^{n+1} - \Big[\quad & \tau_{j+1}^n \left(-\tfrac{1}{3}c_j^n + \tfrac{1}{2}(c_j^n)^2 - \tfrac{1}{6}(c_j^n)^3 + c_j^n d_j^n\right) \\ & +\tau_j^n \left(1 - \tfrac{1}{2}c_j^n - (c_j^n)^2 + \tfrac{1}{2}(c_j^n)^3 - d_j^n - h_j^n - 2c_j^n d_j^n\right) \\ & +\tau_{j-1}^n \left(c_j^n + \tfrac{1}{2}(c_j^n)^2 - \tfrac{1}{2}(c_j^n)^3 + d_j^n + h_j^n + c_j^n d_j^n\right) \\ & +\tau_{j-2}^n \left(-\tfrac{1}{6}c_j^n + \tfrac{1}{6}(c_j^n)^3\right)\Big] = 0,\end{aligned} \tag{2.23}$$

which is nearly of order three. We call this Martins $O\{\sim 3\}$ formula. This FDf is of order three everywhere except that the approximation for the first derivative when multiplied by d_j^n gives a second order accurate result.

Thus we are able to produce higher order FDfs that are consistent with the PDE based upon existing lower order methods.

This process of modification was repeated for several existing FDfs. These methods are listed below in Tables 2.1, 2.2 and 2.3. The FDfs of several others tested later will be given at that time.

These tables give the name of the method which was modified, and the finite-difference forms used to produce the modified method.

The first table refers to modifications done to the FDf called "Upwind 2" which is of order two for constant velocities, but reverts to order one for varying velocities. Its FDf is

$$\tau_j^{n+1} = \left(-\frac{1}{2}c_j^n + \frac{1}{2}(c_j^n)^2\right)\tau_{j-2}^n + \left(2c_j^n - (c_j^n)^2\right)\tau_{j-1}^n + \left(1 - \frac{3}{2}c_j^n + \frac{1}{2}(c_j^n)^2\right)\tau_j^n. \tag{2.24}$$

The entries $(1,3)$ and (1,5) refer to the stencil used by the FDf. The first number indicates the number of grid points at the time level $n+1$, and the second number represents the number of grid points used at the time level n. Hence, the Upwind 2 FDf has a (1,3) stencil.

When the fluid velocity is variable, this FDf has truncation error

$$E\left\{\hat{\tau}\right\} = \frac{\Delta t}{2}\left(\frac{\partial u}{\partial t} - u\frac{\partial u}{\partial x}\right)\frac{\partial \hat{\tau}}{\partial x} + O\left\{(\Delta x)^2, (\Delta t)^2\right\}, \tag{2.25}$$

which we see is of order two for constant velocities and of order one for varying velocities.

The differencing used to produce this FDf is based on first-order forward-time differencing, and second-order backward-spaced differencing.

This produces the basic FDf

$$\tau_j^{n+1} = -\frac{1}{2}c_j^n\tau_{j-2}^n + 2c_j^n\tau_{j-1}^n + \left(1 - \frac{3}{2}c_j^n\right)\tau_j^n \tag{2.26}$$

which has the truncation error

$$E\{\hat{\tau}\} = -\frac{1}{2}\Delta t\frac{\partial^2\hat{\tau}}{\partial t^2} + O\left\{(\Delta x)^2, (\Delta t)^2\right\}. \tag{2.27}$$

Now making the substitution of Equation (2.10) yields the MDPDE

$$\left[\frac{\partial\hat{\tau}}{\partial t} + u\frac{\partial\hat{\tau}}{\partial x} + \frac{1}{2}\Delta t\left[-\left(\frac{\partial u}{\partial t} - u\frac{\partial u}{\partial x}\right)\frac{\partial\hat{\tau}}{\partial x} + u^2\frac{\partial^2\hat{\tau}}{\partial x^2}\right] + O\left\{(\Delta x)^2, (\Delta t)^2\right\}\right]_j^n = 0. \tag{2.28}$$

The formula can be improved upon by removing the largest error term. Of course, a number of finite-difference forms can be used for both derivatives that are to be removed in Equation (2.28), and those experimented with are listed in Table 2.1.

Method	$\partial\hat{\tau}/\partial x$	$\partial^2\hat{\tau}/\partial x^2$
(1,3) Upwind 2 $O\{2\}$	uw1	uw2
(1,5) Upwind 2	uw2	cs
(1,5) Upwind 2	uw1	cs2
(1,5) Upwind 2	cs2	bs1

Table 2.1: Modifications made to the standard Upwind 2 method

In a similar fashion, Noye's (3,3) $O\{4\}$ method (Noye 1992) was modified to be made $O\{3\}$ for varying velocities. See Table 2.2.

$\partial\hat{\tau}/\partial x$	$\partial^2\hat{\tau}/\partial x^2$
cs	cs2
uw1	cs2

Table 2.2: Noye's $O\{4\}$ (3,3) modified to become $O\{3\}$ for non-constant coefficients

We saw in Section 2.2 that the third-order Martins method reverts back to first order for varying velocities. We also saw in detail how this method could be improved upon to order two or indeed order three accuracy.

Many variations were experimented with using its basic MDPDE (2.11) and finite difference forms. Table 2.3 is a summary of those methods developed.

2.3 Stability of the Finite Difference Method

Although a modified FDf may more closely approximate the PDE, there is no guarantee that it will be of use in numerical computations.

Specifically, errors associated with round-off processes during computations may grow exponentially as the process is iterated in time. If the computational error introduced at one time level is amplified by the computation at the next time level, the errors may quickly

Accuracy	d	h	cd
Order 2:			
	cs2		
	uw1		
Order 3:			
	cs2	cs2	cs2
	cs2	uw1	cs2
	uw2	uw1	cs2
	uw2	uw1	uw2
	uw2	uw1	dw2
	uw2	uw1	bs2
	uw2	uw1	fs2
Approximately Order 3:			
	uw1	uw1	cs2
	uw1	uw1	dw2
	uw1	uw1	uw2
	uw1	uw1	fs2
	uw1	uw1	bs2

Table 2.3: Modified methods developed for Martins formula

grow, and eventually any original information will have been destroyed. This phenomenon is termed instability.

A FDf is considered to be stable if the errors introduced through computations decay with successive iterations of the method. Noye (1992) describes several methods by which the stability of FDfs can be analysed. In particular, he discusses von Neumann stability analysis and defines the amplification factor G to be the amount by which errors can grow in one time step. Hence

$$|G| \leq 1 \tag{2.29}$$

for all possible wave numbers of component waves, is the criterion for stability. The author was given access to a computer program capable of testing FDfs for stability based upon von Neumann stability analysis.

An example of the output from the stability analysis program is attached (see Figure 2.1). It is for Noye's (3,3) method for constant coefficients modified to be of order three for the variable coefficient case, using first-order upwind differencing to replace the first derivative in the error, and centred-space differencing to replace the second derivative. The output shows that the method is stable for all values of c_j^n and d_j^n small enough — the region of stability is closely approximated by the ellipse:

$$(c_j^n)^2 + \left(3d_j^n\right)^2 \leq 1. \tag{2.30}$$

This can be achieved by scaling the parameters of the problem. For example, choosing a small enough time step would satisfy constraints on c_j^n as:

$$c_j^n = u_j^n \frac{\Delta t}{\Delta x}. \tag{2.31}$$

All of the methods listed in Tables 2.1, 2.2 and 2.3 were tested for stability.

The results for many of the methods indicated that the methods were unstable for very large regions, and in particular near the origin (that is, where the velocities are small but not zero). This is in general undesirable because any method that is required to model fluid flow should be suitable for small velocities. However, it is possible that errors that grow during time steps for which the formula is unstable may be reduced by subsequent calculations at time steps for which the formula is stable. So, it is possible that a FDf that is unstable for small velocities may still be useful in practice.

One objective was to improve upon Martin's method to produce a method for non-constant velocity fields with a third order accuracy.

As a result the third order accurate methods listed in Table 2.3 were developed. However, their stability analyses all suffer the same flaw. The methods are not stable for values of $d_j^n \neq 0$ when $h_j^n = 0$. An example of one such method is displayed in Figure A.1 in the Appendix.

It was thought that the second-order upwind differencing of the derivative whose coefficient is d, might be the cause of this lack of stability. Using first-order upwind differencing on that derivative might restore some stability, although this would produce a method that was not truly third order accurate — that particular term would be only second order accurate.

Hence the several nearly third order methods listed above were developed and analysed for stability. The results showed there were still some regions near the origin for which the methods were unstable.

The nearly third order method derived earlier (Equation 2.23) was considered suitable for further investigation as its stability analysis, although not perfect, was promising. Representative output from the stability programs for this formula are included in Figures A.2 and A.3 in the Appendix for the readers reference.

2.4 Addendum

When the nearly third order Martin's FDf (2.23) discussed above was tested for accuracy, it showed no tendency to be unstable.

This leads us to ask the question:"Are any of the other methods listed above suitable for use as an accurate FDf despite the fact that von Neumann stability analysis indicates they are unstable near the origin?"

This will be the subject of a later study.

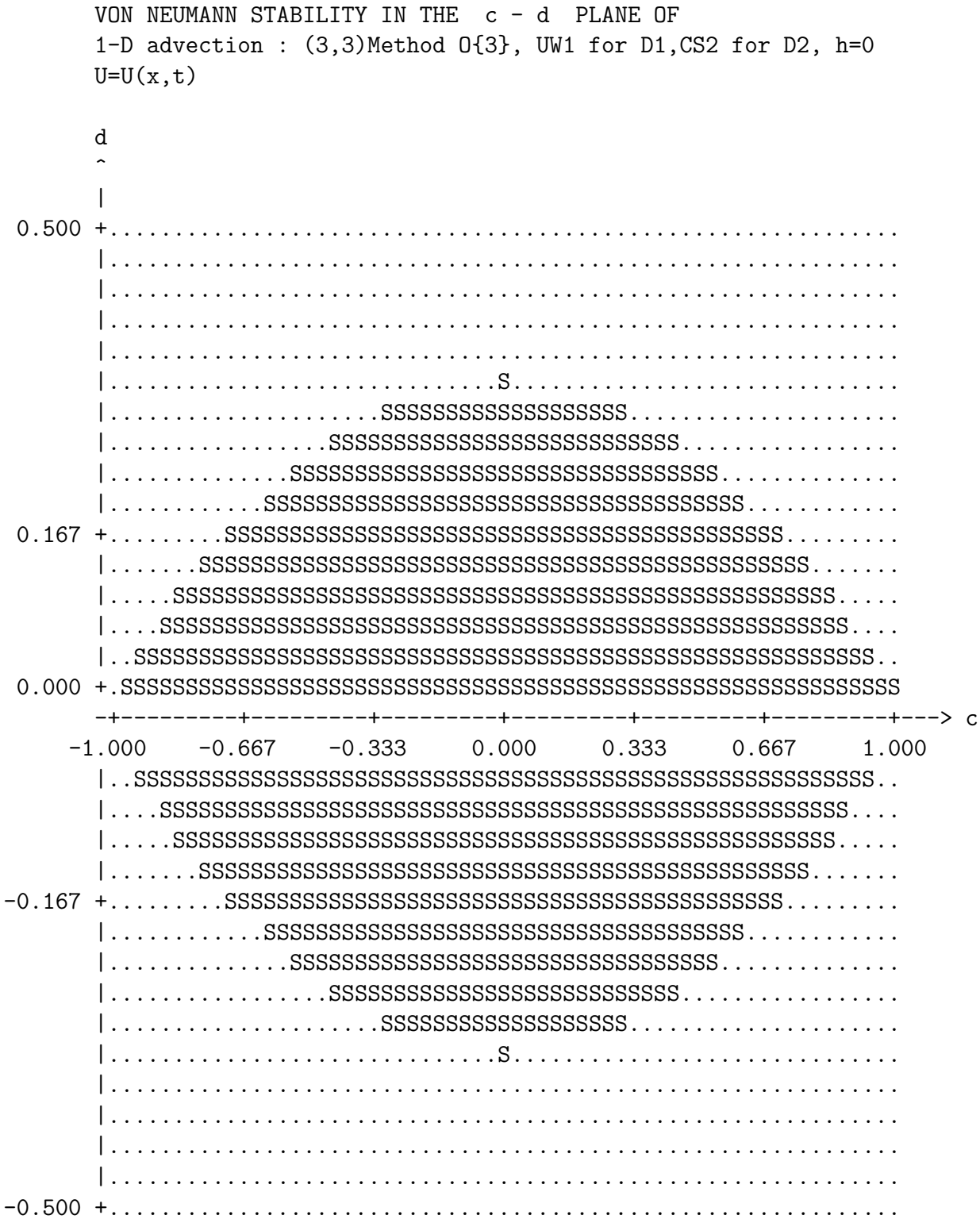

Figure 2.1: Stability analysis for Noye's (3,3) modified method.

Chapter 3

One-Dimensional Advection Tests

Before using a FDf in application, it is necessary to have some knowledge of the accuracy of the results that it will produce.

In practice, FDMs are used where the governing equation has no analytic solution, and the FDf may be the best method available — yet we do not know how close the solution of the FDf is to the exact solution of the partial differential equation.

One method of gaining an understanding of the accuracy of the FDf is to use it to simulate a situation in which an exact analytic solution is known. Thus the FDf may be compared to the exact solution at every point at which the FDf is evaluated, and some numerical comparison can be made.

We are interested in the accuracy of FDMs for velocity fields that vary in space and time, as opposed to being constant (work on constant velocities is well documented).

3.1 Space-Varying Velocities

One velocity field that has a space-varying component is

$$u(x) = m(x + a) \tag{3.1}$$

for which the one-dimensional advection equation

$$\frac{\partial \hat{\tau}}{\partial t} + m(x + a)\frac{\partial \hat{\tau}}{\partial x} = 0 \tag{3.2}$$

has the exact solution

$$\hat{\tau}(x, t) = e^{-k\left[(x+a)e^{-mt} - x_0 - a\right]^2}. \tag{3.3}$$

Hence this velocity field can be used as a suitable test for the accuracy of a given FDf in velocity fields that vary in space.

This does not provide knowledge of the accuracy of the method for arbitrary space-varying velocity fields, but can be used as a general guide.

A similar situation applies for time-varying velocity fields, which will be discussed in detail later in Section 3.2.

By setting $t = 0$ in Equation (3.3), we see that these equations represent the advection of a Gaussian peak initially centered at x_0 in a velocity field that varies linerly in space. That is,

$$\hat{\tau}\,(x, 0) = e^{-k[x-x_0]^2}. \tag{3.4}$$

To determine the results produced by the FDf it is necessary to write a computer program that evaluates the FDf at a later time, and compares it to the exact solution at that time, producing some report. Graphical representation of the results is invaluable for easy interpretation.

The author modified an existing program developed by Waluyo (1991) to achieve this. Coding of the programs is included, but a brief description is given here.

The program stores two arrays of values called $Tauold$, and $Taunew$, each of length $bigJ$, where $bigJ$ is the number of grid points in the x-domain. $Tauold$ contains the values of the solution to the FDf at time level n, and $Taunew$ contains the values of the solution to the FDf at time level $n + 1$. The program uses the values of $Tauold$ and the FDf to compute the values of $Taunew$ for each grid point $j = 1\ (1)\ bigJ - 1$. It then determines the value of $Taunew$ at $j = 0$ and $j = bigJ$ from the exact solution applied at the boundaries at time $(n + 1)\Delta t$. Once all the values of $Taunew$ have been computed at the new time level, it copies those values into the array $Tauold$ and the process is repeated. In this way, the solution to the FDf at time level $bigN$ corresponding to the time $finalt$ is determined.

The program then finds the exact solution of the PDE at that time level from the analytical form, and makes a comparison of the two values at every grid point. The error at each point is calculated and summarised over all grid points giving the three error measures: rms error, average error and maximum error. The maximum error is the greatest absolute difference between the exact solution and the approximate solution at the final time level. The average error is the arithmetic mean of the absolute differences between the exact analytic solution and the approximate solution at the final time level. The rms error is the root of the mean of the squares of the difference between the exact analytic solution and the approximate solution at every point at the final time level. These measures permit the comparison of one FDf to another. The process is repeated for the modified FDf (the FDf which has been improved upon by removing the largest error term) so that it too can be compared to the exact solution.

The three sets of calculated values (the exact analytic solution, the approximate FDf solution, and the modified FDf solution) are output to files which are later used by the graphical package S, to produce a visual display of the results. For example, one velocity field that was used with initial condition

$$\hat{\tau}\,(x, 0) = e^{-12.5(x-0.5)^2} \tag{3.5}$$

was

$$u\,(x) = -4\,(x - 6)\,. \tag{3.6}$$

This gave a final distribution of values of $\hat{\tau}$ that was neither too concentrated nor too distributed, and so were considered a reasonable test for the FDfs.

3.1.1 Explicit methods .:.

Explicit methods are those in which the value of the function at the new time level can be expressed explicitly in terms of the values at the previous time levels. These are typified by the FDf having only one term involving the new time level. For example

$$\tau_j^{n+1} = a\tau_{j-1}^n + b\tau_j^n + c\tau_{j-1}^n \tag{3.7}$$

is an explicit FDf.

Explicit formulae used in the past include those of Leith (Noye 1992), Fromm (Noye 1992), Rusanov (Rusanov 1970), Martin, and the Upwind $O\{1\}$ (Courant et al. 1952) and $O\{2\}$ (Noye 1992) methods. The master program was modified to cater for these FDfs and the results are included below.

The programs were each run with three different values for the maximum of the Courant number namely, 0.3, 0.6, and 0.96. Typically, the results for $c = 0.6$ lie between those for $c = 0.3$ and $c = 0.96$, and so will not be included in this assessment. Similarly, the three error measures of root mean square error, average absolute error and maximum error are closely correlated, and so only the average absolute error will be used in this discussion.

The results are displayed in tabular form for comparison of performance between different FDfs. The tables list the average absolute error, peak minimum height, peak maximum height, peak shift and CPU time for each method used in the tests. They are sorted in order of performance from poorest to best, with the measure of performance being the average absolute error. In these tables, both the standard and modified versions will appear, enabling comparison between a modified version of one method, and the standard version of another. A table of this type will appear for each of the values of the Courant number, 0.3 and 0.96.

Tables 3.1 and 3.2 show the results obtained from the explicit methods tested.

The Upwind $O\{1\}$ method shown in Figure 3.1 does not have a modified version. It performs poorly, having the largest average absolute error of all the methods tested. This is due to the large amount of artificial diffusion that is created by the FDf. Minor points in its favour are its speed (it used the least CPU time) and the fact that it never creates negative values.

The Upwind $O\{2\}$ method also gives poor results. Its peak is leading that of the exact analytic solution. This is due to the fact that the component waves move faster than are required. As a result, there is a large negative leading oscillation, and the amplitude response is poor. The modification to this method proves worthless, as it has a tendency to introduce more artificial diffusion.

Leith's $O\{2\}$ method (see Figure 3.2) produces qualitatively similar results, except that the peak and large negative oscillation are trailing. In this case however, the modification is an improvement. It corrects the peak position and so reduces the average absolute error.

Fromm developed his third order method based upon an average of Leith and the Upwind $O\{2\}$ method. Naturally for varying velocities it is of order one. It produces a result that is reasonable overall. The modified version is approximately third order. It produces good

Space-varying velocity in one dimension Explicit methods Courant number = 0.3						
method		ave.error	min.height	max.height	peak shift	cpu
Upwind $O\{1\}$ (1,3)		0.0656	0.0000	0.5254	-0.0104	0.2s
Upwind $O\{2\}$ (1,3)	(mod)	0.0315	-0.0913	0.8812	0.0553	0.6s
Upwind $O\{2\}$ (1,3)		0.0310	-0.0917	0.8829	0.0534	0.5s
Leith $O\{2\}$ (1,3)		0.0275	-0.0851	0.9430	-0.0509	0.2s
Leith $O\{2\}$ (1,3)	(mod)	0.0268	-0.0843	0.9411	-0.0491	0.3s
Fromm $O\{\sim 3\}$ (1,3)	(mod)	0.0080	-0.0105	0.9312	0.0089	0.7s
Fromm $O\{\sim 3\}$ (1,3)		0.0076	-0.0106	0.9337	0.0070	0.6s
Martins $O\{2\}$ (1,5)		0.0061	-0.0032	0.9468	-0.0037	0.8s
Martins $O\{2\}$ (1,5)	(mod)	0.0061	-0.0030	0.9450	-0.0019	0.9s
Martins $O\{\sim 3\}$ (1,5)	(mod)	0.0061	-0.0031	0.9443	-0.0019	1.0s
Rusanov MAE $O\{\sim 3\}$		0.0025	-0.0001	0.9947	-0.0070	0.8s
Rusanov MAE $O\{\sim 3\}$	(mod)	0.0025	-0.0001	0.9888	-0.0066	1.0s

Table 3.1: Results of tests of explicit methods for one-dimensional space-varying velocities with $c = 0.3$.

Space-varying velocity in one dimension Explicit methods Courant number = 0.96						
method		ave.error	min.height	max.height	peak shift	cpu
Upwind $O\{1\}$ (1,3)		0.0355	0.0000	0.7128	-0.0021	0.1s
Leith $O\{2\}$ (1,3)		0.0154	-0.0185	0.9491	-0.0315	0.1s
Leith $O\{2\}$ (1,3)	(mod)	0.0132	-0.0192	0.9551	-0.0256	0.1s
Upwind $O\{2\}$ (1,3)	(mod)	0.0108	-0.0160	0.9771	0.0225	0.2s
Upwind $O\{2\}$ (1,3)		0.0089	-0.0162	0.9696	0.0167	0.2s
Fromm $O\{\sim 3\}$ (1,3)		0.0051	-0.0007	0.9750	-0.0085	0.2s
Martins $O\{2\}$ (1,5)		0.0042	-0.0003	0.9753	-0.0056	0.2s
Rusanov MAE $O\{\sim 3\}$		0.0041	0.0000	0.9954	-0.0082	0.3s
Rusanov MAE $O\{\sim 3\}$	(mod)	0.0038	0.0000	0.9653	-0.0062	0.3s
Martins $O\{\sim 3\}$ (1,5)	(mod)	0.0030	-0.0003	0.9664	0.0008	0.3s
Fromm $O\{\sim 3\}$ (1,3)	(mod)	0.0028	-0.0006	0.9752	-0.0029	0.2s
Martins $O\{2\}$ (1,5)	(mod)	0.0023	-0.0003	0.9747	0.0000	0.3s

Table 3.2: Results of tests of explicit methods for one-dimensional space-varying velocities with $c = 0.96$.

Figure 3.1: Artificial diffusion produced by the Upwind $O\{1\}$ formula.

Figure 3.2: Leith's method with trailing oscillation.

results for large values of the Courant number ($c = 0.96$), and yet interestingly produces slightly poorer results than the standard method for small values of c ($c = 0.3$).

Martins third order method also degrades to order one for varying velocities. It has been modified to order two accuracy, and again to approximately third order accuracy. These three FDfs will be referred to as Martins $O\{1\}$, $O\{2\}$ and $O\{\sim 3\}$ respectively. All of these display excellent peak positioning, and in general are very good everywhere. The $O\{\sim 3\}$ version suffers in amplitude response at $c = 1$.

The Rusanov Minimum Amplitude (MAE) Error method shown in Figure 3.3 is fourth order accurate for constant velocities. An approximately third order modified method has been developed for varying velocities. It shows very reasonable result, performing better than all other explicit methods tested. It produces the best peak heights, and excellent minimum heights (all other methods except Upwind $O\{1\}$ produced comparatively large negative values). The peak height was however diminished for $c = 0.96$. The modified version improves on peak positioning, but reduces peak height.

Note that the modified versions are the better performers for $c = 0.96$, while for $c = 0.3$, the methods appear in the order that they would be expected to for constant velocities. This suggests that the modifications are useful for applications with large c, but not so useful for applications requiring small c.

A general observation is that the methods are generally more accurate for the cases of larger maximum Courant number, than for those with a smaller maximum. This is not true however for Rusanov's minimum amplitude error method. Also of note is that the modification to the FDf to increase the order of its accuracy has very little effect on improving the accuracy of the method.

This is most probably due to the higher-order terms each only contributing small amounts to the error so that removal of any one of them has little effect. This can be readily verified for the case of Martin's method by substituting the values used in the simulation into the lowest order error terms in the modified discretised PDE.

The program was designed to record the maximum values of d and h, the parameters used in the correction of the method to the higher order actually used. Typically they were values less than 0.004 when c was approximately one. Observation of the modified FDfs shows that the modifications made would only produce small improvements in accuracy.

This suggests that the derivatives of the space-varying velocity field are only small relative to the magnitude of the fluid velocity so that the errors introduced by the varying velocity field are small. Consequently, the modifications do little to improve the accuracy of the FDfs.

3.1.2 Marching methods ::.

Marching methods are those in which the FDf has two terms associated with values at the new time level. For example, the general formula

$$a\tau_{j-1}^{n+1} + b\tau_j^{n+1} = c\tau_{j-1}^n + d\tau_j^n + e\tau_{j+1}^n \tag{3.8}$$

Figure 3.3: Rusanov's method showing the artificial diffusion introduced by modification.

Space-varying velocity in one dimension Marching methods Courant number = 0.3					
method	ave.error	min.height	max.height	peak shift	cpu
Roberts and Weiss (2,2)	0.0388	-0.1505	0.9324	-0.0661	0.5s
Box (2,2)	0.0118	-0.0144	0.9848	0.0233	0.4s
Noye and Tan (2,3)	0.0044	-0.0006	0.9794	-0.0053	0.6s

Table 3.3: Results of tests of marching methods for one-dimensional space-varying velocities with $c = 0.3$.

Space-varying velocity in one dimension Marching methods Courant number = 0.96					
method	ave.error	min.height	max.height	peak shift	cpu
Roberts and Weiss (2,2)	0.0610	-0.2292	0.8803	-0.0939	0.2s
Box (2,2)	0.0053	-0.0009	0.9862	0.0085	0.1s
Noye and Tan (2,3)	0.0049	-0.0001	0.9854	-0.0066	0.2s

Table 3.4: Results of tests of marching methods for one-dimensional space-varying velocities with $c = 0.96$.

could be used to march from the left hand boundary when the velocity $u > 0$. This is because it is possible with these methods to express one of the values at the new time level in terms of the known values at the remaining points on the stencil (the value of the other point at the new time level is given initially from the boundary condition). In this way it is possible to march across the grid, at each step determining the value at the new time level which will be the known value at the new time level at the next step.

The essential difference here is that points on a given time level must be calculated sequentially, whereas in explicit methods they can be calculated in an arbitrary order, or indeed simultaneously. This has important ramifications in the use of super computers that utilise vector processing. The vectorisation of the process is not as easy to achieve for marching methods as it is for explicit methods, which partially defeats some of the advantages that can be obtained by the use of super computers.

Slight modification to the general computer program for explicit methods yields a program that is useful for marching methods.

The tests were carried out for the Roberts and Weiss $(2, 2)$ (Noye 1992), Box $(2, 2)$ (Noye 1992), and Noye and Tan $(2, 3)$ (Noye and Tan 1988) methods. None of these methods have a modified version. The results are included below.

Tables 3.3 and 3.4 display the results obtained from the tests using marching methods.

Roberts and Weiss produces poor results in many aspects. It has poor amplitude response

and the peak trails the exact solution. Its greatest flaw is that it has large trailing oscillations (see Figure 3.4). In general, this method does not behave well for this test.

In Figure 3.5 we see that the peak of the Box $(2, 2)$ method is slightly skewed, with a hiccough at the base of its leading edge. It is however a vast improvement on Roberts and Weiss.

The method developed by Noye and Tan has good maximum amplitude, peak positioning and minimum values. In short, it produces good results.

3.1.3 Fully implicit methods :::

These methods have three or more terms associated with the new time level. An example of an implicit FDf is

$$a\tau_{j-1}^{n+1} + b\tau_j^{n+1} + c\tau_{j+1}^{n+1} = d\tau_{j-1}^n + e\tau_j^n + f\tau_{j+1}^n. \tag{3.9}$$

As a result these methods cannot be solved explicitly for a single unknown value. Instead a series of simultaneous equations must be solved to determine the value of the function at all grid points at the new time level.

An efficient method of solving these equations is the Thomas algorithm. Noye (1992) gives a thorough derivation of the algorithm. The FDf (3.9) when applied at each grid point at time level $n + 1$ produces a system of equations that may be represented by a tridiagonal matrix. The first equation which contains only two unknowns can be used to eliminate the first unknown of the second equation, leaving the second equation with only two unknowns. This is repeated for each of the equations until the last equation, which also only contains two unknown values. Elimination of the first unknown determines the value of the other, and so by a process of back substitution all unknowns can easily be determined. This algorithm provides a speedy method of solution of these $(j - 1)$ simultaneous equations.

The original program was modified to encorporate this algorithm, thus making it suitable for implicit FDfs.

The implicit methods tested in this study are the Crank-Nicolson $(3, 3)$ (Noye 1992), Leonard-Noye$(3, 3)$ (Leonard and Noye 1990), Linear Finite Element Crank-Nicolson type $(3, 3)$ and Noye $(3, 3)$ methods. Tabular results of this method are included below.

Note that marching methods, sometimes called semi-implicit methods, can also be solved using only the forward sweep of the Thomas algorithm, as the coefficient of τ_{j+1}^{n+1} in each of the simultaneous equations is zero.

The marching methods discussed in the previous section are also included in Tables 3.5 and 3.6, which summarise the results for the tests run, mainly to enable the comparison of the time required when marching methods are implemented using an implicit method program.

As was the case for marching methods, none of these implicit methods were tested against modified versions.

The Crank-Nicolson method produces a large trailing oscillation, especially for larger values of the Courant number as shown in Figure 3.6. As is the trend for methods with oscillation, the peak is shifted towards the side with the oscillation (in this case the peak is trailing).

Figure 3.4: The large trailing oscillations of the Roberts and Weiss formula.

Figure 3.5: Hiccough at the base of the peak of the Box $(2, 2)$ method.

Space-varying velocity in one dimension Implicit methods Courant number = 0.3					
method	ave.error	min.height	max.height	peak shift	cpu
Roberts and Weiss (2,2)	0.0388	-0.1505	0.9324	-0.0661	0.6s
Crank-Nicolson (3,3)	0.0303	-0.1047	0.9513	-0.0547	0.4s
Leonard-Noye (3,3)	0.0217	-0.0620	0.9627	-0.0430	0.6s
Box (2,2)	0.0118	-0.0144	0.9848	0.0233	0.6s
Noye and Tan (2,3)	0.0044	-0.0006	0.9794	-0.0053	1.0s
LFE Crank-Nicolson type	0.0017	0.0000	1.0091	-0.0032	0.5s
Noye (3,3)	0.0014	0.0000	1.0085	-0.0015	0.6s

Table 3.5: Results of tests of implicit methods for one-dimensional space-varying velocities with $c = 0.3$.

Space-varying velocity in one dimension Implicit methods Courant number = 0.96					
method	ave.error	min.height	max.height	peak shift	cpu
Roberts and Weiss (2,2)	0.0610	-0.2292	0.8803	-0.0939	0.2s
Crank-Nicolson (3,3)	0.0358	-0.1359	0.9401	-0.0622	0.1s
Leonard-Noye (3,3)	0.0145	-0.0206	0.9776	-0.0313	0.2s
LFE Crank-Nicolson type	0.0077	-0.0046	1.0016	-0.0167	0.2s
Box (2,2)	0.0053	-0.0009	0.9862	0.0085	0.2s
Noye and Tan (2,3)	0.0049	-0.0001	0.9854	-0.0066	0.3s
Noye (3,3)	0.0012	0.0000	1.0069	-0.0009	0.2s

Table 3.6: Results of tests of implicit methods for one-dimensional space-varying velocities with $c = 0.96$.

Figure 3.6: Large trailing oscillations of the Crank-Nicolson method.

The Leonard-Noye method produces qualitatively similar results to Crank-Nicolson, but it is more accurate.

The Linear Finite Element (LFE) Crank-Nicolson type gives very reasonable results for small values of c, but the peak starts to trail and develops a trailing hiccough near the base of the peak when the Courant number is approximately one. See Figure 3.7

Noye's $(3, 3)$ method produces a minimum value of zero, so no negative answers are produced, and has only a very small peak shift. Its amplitude response is excellent, actually producing a value in excess of one. Although at first this might seem unreasonable, it is possibly due to a component wave of the peak moving slightly and producing larger values at this position. This method produces excellent results for this test (see Figure 3.8).

Also of interest is a comparison of the CPU time required to run marching methods using the program for implicit methods. The CPU times for Roberts and Weiss, Box, and Noye and Tan methods suggests that the marching methods program requires only approximately 65% of the CPU time required by the implicit program to solve the same problem using the same FDf.

As a brief note it is worthwhile to mention that a different choice of parameters for the velocity field might well produce a different ordering of the accuracy of the methods investigated here. If the Gaussian peak of the analytic solution were more steep, the measure of average absolute error would give more weight to the peak position. That is, if the peak position was shifted by a small amount, the error measure would indicate a much worse result for a more steep peak than for a less steep peak. However, if the test peak was less steep, then more weight would be given to the height of the peak.

The test chosen represents a reasonable comparison between peak height and peak position. Although some re-ordering might occur for different test parameters, the qualitative nature of the methods used (and hence the above analysis) would still be similar.

3.2 Time-Varying Velocity Fields

Again we require a velocity field for which we have an analytic solution to the advection equation for comparison of the FDMs to the exact solution.

One such velocity field that has a time-varying component is

$$u(t) = U \sin(\omega t), \tag{3.10}$$

for which the advection equation

$$\frac{\partial \hat{\tau}}{\partial t} + U \sin(\omega t) \frac{\partial \hat{\tau}}{\partial x} = 0, \tag{3.11}$$

has the analytic solution

$$\hat{\tau}(x, t) = e^{-k\left[x - x_0 - 2U\omega^{-1} \sin^2\left(\frac{\omega t}{2}\right)\right]^2}. \tag{3.12}$$

This equation represent the advection of a gaussian peak, initially at position x_0, in a velocity field that varies sinusoidally in time.

Figure 3.7: The LFE Crank-Nicolson method has a trailing peak.

Figure 3.8: Excellent results are produced by the Noye (3,3) formula.

Time-varying velocity in one dimension Explicit methods Final time = 4.0 Courant number =$\pi/30$						
method		ave.error	min.height	max.height	peak shift	cpu
Upwind $O\{1\}$ (1,3)		0.0664	0.0000	0.1345	0.0058	1.8s
Upwind $O\{2\}$ (1,5)	(mod)	0.0489	-0.0178	0.3911	0.0102	4.3s
Upwind $O\{2\}$ (1,5)		0.0487	-0.0180	0.3942	0.0102	3.7s
Fromm (1,5)	(mod)	0.0420	-0.0374	0.4622	0.0002	5.1s
Fromm (1,5)		0.0419	-0.0392	0.4674	0.0002	4.3s
Martin $O\{\sim 3\}$ (1,5)	(mod)	0.0393	-0.0365	0.4911	0.0000	7.7s
Martin $O\{2\}$ (1,5)	(mod)	0.0393	-0.0365	0.4911	0.0000	6.2s
Martin $O\{1\}$ (1,5)		0.0392	-0.0380	0.4973	0.0000	5.7s
Leith (1,3)	(mod)	0.0181	-0.0260	0.7138	0.0000	2.0s
Leith (1,3)		0.0168	-0.0310	0.7351	0.0000	1.6s
Rusanov $O\{\sim 3\}$(1,5)	(mod)	0.0098	-0.0334	0.8334	0.0000	8.0s
Rusanov (1,5)		0.0092	-0.0359	0.8537	0.0000	6.4s

Table 3.7: Results of tests of explicit methods for one-dimensional time-varying velocities with $t = 4$ and $c = \pi/30$.

As the distribution advects, it oscillates back and forth, but maintains its original shape.

Again an existing program was modified to simulate the advection of the peak in this time-varying velocity field. The programs were run for the same methods as those in the space-varying velocity case, and each time for two different maximum values of the Courant number namely, $\pi/4$ and $\pi/30$. They were also run for two different final times, $t = 4$ and $t = 5$, so that the results could be viewed when the peak was in the centre and at the extremity of its trajectory.

The velocity field

$$u = \frac{\pi}{2} \sin\left(\frac{\pi}{2}t + \frac{\pi}{2}\right), \tag{3.13}$$

with initial condition

$$\hat{\tau}\,(x, 0) = e^{-144(x-1.5)^2} \tag{3.14}$$

had previously been determined (Waluyo 1991) to produce a reasonable test for the accuracy of FDMs in solving the advection equation in one dimension.

3.2.1 Explicit methods .:.

In the interests of clarity, only the results for the Courant number $\pi/30$ are included. This is the more stringent test of the FDf for most methods. The results are given for both times $t = 4$ and $t = 5$ in Tables 3.7 and 3.8. The modified versions are included to allow direct comparison of the standard and modified versions.

Time-varying velocity in one dimension Explicit methods Final time = 5.0 Courant number =$\pi/30$						
method		ave.error	min.height	max.height	peak shift	cpu
Upwind $O\{1\}$ (1,3)		0.0616	0.0000	0.1183	0.0184	2.0s
Upwind $O\{2\}$ (1,5)		0.0528	0.0000	0.3667	0.0559	4.6s
Upwind $O\{2\}$ (1,5)	(mod)	0.0528	0.0000	0.3636	0.0528	5.4s
Leith (1,3)		0.0511	-0.2293	0.5992	-0.0768	1.9s
Leith (1,3)	(mod)	0.0504	-0.2123	0.5864	-0.0773	2.4s
Fromm (1,5)		0.0427	-0.0513	0.4424	0.0146	5.5s
Fromm (1,5)	(mod)	0.0427	-0.0483	0.4374	0.0120	6.3s
Martin $O\{2\}$ (1,5)	(mod)	0.0409	-0.0420	0.4670	-0.0020	9.0s
Martin $O\{\sim 3\}$ (1,5)	(mod)	0.0409	-0.0420	0.4671	-0.0020	8.7s
Martin $O\{1\}$ (1,5)		0.0406	-0.0446	0.4739	0.0004	7.9s
Rusanov (1,5)		0.0243	-0.1400	0.7671	-0.0207	8.0s
Rusanov (1,5)	(mod)	0.0238	-0.1344	0.7459	-0.0209	10.5s

Table 3.8: Results of tests of explicit methods for one-dimensional time-varying velocities with $t = 5$ and $c = \pi/30$.

As for the space-varying velocity case, the Upwind $O\{1\}$ method gave poor amplitude response. In this case however, it was nearly flat, as is shown in Figure 3.9. This is again due to the excessive artificial diffusion produced by the method. Its one redeeming point is that it does not produce negative values. Otherwise, this method is worthless.

The Upwind $O\{2\}$ method produces a single negative oscillation each side of the peak at time $t = 4$ (that is , when the peak is in its initial central position) for $c = \pi/4$, but due to the increased amount of diffusion, there is only a leading oscillation for $c = \pi/30$. The method has too much diffusion at $t = 5$ to be classified as reliable. There is no real difference between the standard and modified methods for $c = \pi/30$.

Leith also produces similar oscillations at $t = 4$. Although the peak is located exactly, amplitude response is not as good as is desired. The modified version introduces a further small amount of diffusion. At $t = 5$ there exists large trailing oscillations, with poor peak positioning and amplitude response. The modification does however produce a slight improvement.

At $t = 4$, Fromm's method possesses good peak positioning, but poor amplitude response and has a negative leading and trailing oscillation as shown in Figure 3.10. At $t = 5$, the leading oscillation has become more negative.

The Martin $O\{2\}$ and $O\{\sim 3\}$ methods produced results with little difference between them. They display better peak positioning than the $O\{1\}$ version. They are qualitatively similar to Fromm's method.

Rusanov's MAE method produces good results for $t = 4$. It has excellent peak positioning, good amplitude response but possesses small negative leading and trailing oscillations. At

Figure 3.9: Large amounts of diffusion produced by the Upwind $O\{1\}$ method.

Figure 3.10: The poor amplitude response and negative values produced by Fromm's method.

Time-varying velocity in one dimension Implicit methods Final time = 4.0 Courant number =$\pi/30$					
method	ave.error	min.height	max.height	peak shift	cpu
Noye and Tan (2,3)	0.0300	-0.0417	0.5975	0.0000	4.8s
Box (2,2)	0.0205	-0.0877	0.8291	0.0051	3.8s
Noye (3,3)	0.0000	0.0000	1.0000	0.0000	3.8s
Leonard-Noye (3,3)	0.0000	0.0000	1.0000	0.0000	3.6s
Crank-Nicolson (3,3)	0.0000	0.0000	1.0000	0.0000	3.1s
LFE Crank-Nicolson type	0.0000	0.0000	1.0000	0.0000	3.1s

Table 3.9: Results of tests of implicit methods for one-dimensional time-varying velocities with $t = 4$ and $c = \pi/30$.

$t = 5$ it develops large trailing oscillations and the peak trails (see Figure 3.11). This method requires the largest amount of CPU time.

In general the modified versions were not an improvement for $t = 4$, and only produced a slight improvement for $t = 5$. The methods performed in a fashion that was to be expected for their order of accuracy for the constant velocity case.

3.2.2 Implicit methods :::

For this test, the same implicit methods as before were tested.

In this case however, the marching methods were included also. They were not treated by a test that used a marching method program, as the only additional information that could be obtained would be the CPU time required to run the methods.

The inclusion of the marching methods in the implicit method analysis for space-varying velocities serves as a reasonable guide to the advantage to be gained by producing an algorithm that tests marching methods.

For this reason, marching methods will not be differentiated from implicit methods for the remainder of this thesis.

The most striking point to notice when analysing the performance of the implicit methods is the outstanding accuracy of some of the methods at $t = 4$ (see Table 3.9). In particular the Crank-Nicolson, Leonard-Noye, LFE Crank-Nicolson type, and the Noye $(3, 3)$ methods all produced perfect results.

Results at time $t = 5$ are given in Table 3.10.

The Noye and Tan method although possessing poor amplitude response, produced a good peak position for both $t = 4$ and $t = 5$. It has small oscillations on both sides of the peak, and maintains this shape throughout its motion.

Figure 3.11: Trailing oscillations developed by Rusanov's formula at $t = 5$.

Time-varying velocity in one dimension Implicit methods Final time = 5.0 Courant number =$\pi/30$					
method	ave.error	min.height	max.height	peak shift	cpu
Crank-Nicolson (3,3)	0.0833	-0.3593	0.6424	-0.0905	3.5s
Leonard-Noye (3,3)	0.0738	-0.3429	0.6713	-0.0798	4.9s
Box (2,2)	0.0437	-0.2678	0.7262	0.0590	4.4s
Noye and Tan (2,3)	0.0321	-0.0417	0.5729	0.0034	6.1s
LFE Crank-Nicolson type	0.0209	-0.1011	0.8952	-0.0115	3.8s
Noye (3,3)	0.0203	-0.0945	0.8984	-0.0108	5.0s

Table 3.10: Results of tests of implicit methods for one-dimensional time-varying velocities with $t = 5$ and $c = \pi/30$.

The Box method has reasonable peak height, but there are small oscillations on either side of the peak at $t = 4$ (see Figure 3.12). These become large leading oscillations at $t = 5$.

The Crank-Nicolson method as noted earlier produced exact results for $t = 4$. At $t = 5$ however, it has many large trailing oscillations that gradually get smaller in size. These contrasting results can be seen in Figures 3.13 and 3.14

Leonard-Noye, LFE Crank-Nicolson and Noye $(3, 3)$ methods all demonstrate similar qualitative behaviour to Crank-Nicolson. They are listed here in order of increasing accuracy as the size of these oscillations decrease. A graph of the Noye $(3, 3)$ method at $t = 5$ is displayed in Figure 3.15.

As a general comparison between explicit and implicit methods, the explicit methods have poorer amplitude response. Four of the six implicit methods tested are perfect at $t = 4$, but develop many large trailing oscillations at $t = 5$. The explicit methods are not as oscillatory at $t = 5$.

These oscillations produced by the implicit methods are a result of a problem with the speed of the component waves of the Gaussian peak. As the results are perfect at $t = 4$, they are expected to be perfect again at $t = 6$, $t = 8$, $t = 10$ etc. That is, there is no loss of peak height, but the Gaussian peak is distorted by wave components moving at different speeds as the peak advects. When it reaches its outer limit, the current direction changes, and the same phenomenon rebuilds the peak.

A similar point can be made about the parameters used in this test as for the space-varying test. If the peak were chosen to be of a different width, the results of the test would be quantitatively different, and some re-ordering of accuracy might result. The qualitative nature of the methods would however remain the same.

Figure 3.12: The Box (2,2) method has oscillations on both sides of the peak.

Figure 3.13: Perfect results produced by Crank-Nicolson at $t = 4$.

Figure 3.14: Large trailing oscillations produced by Crank-Nicolson at $t = 5$.

Figure 3.15: Smaller oscillations are produced by the Noye (3,3) formula.

Chapter 4

Two-Dimensional Advection Tests using Two-Dimensional Stencils

The concepts used in the applications of FDMs to one-dimensional problems involving advection may be expanded to two dimensions with some modifications. There are two major approaches to this problem. The first is to use a truly two-dimensional stencil, and the second is to use the method of time splitting to reduce the two-dimensional problem to two simplier one-dimensional problems.

In this chapter we will discuss the first of these methods, the use of two-dimensional stencils; the alternative method of time splitting will be discussed in Chapter 5.

4.1 The Two-Dimensional Problem

In two-dimensional problems we may use a spatial grid of points (also in two dimensions) at which we wish to know the value of the function at some later time.

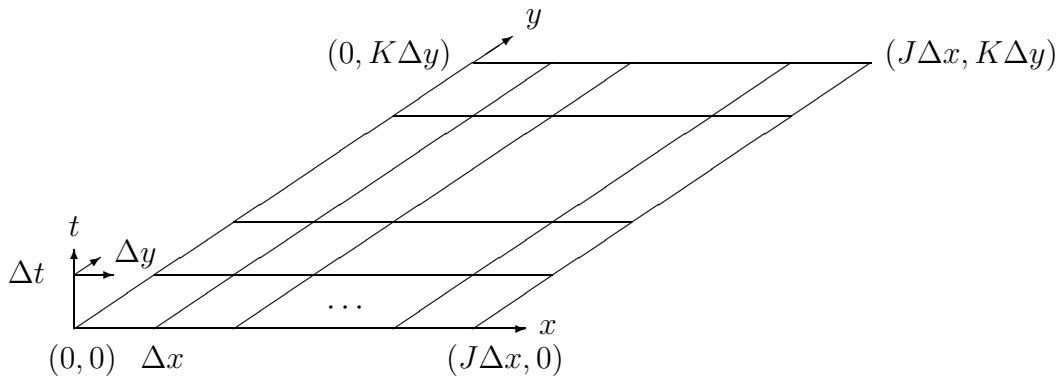

Figure 4.1: The two-dimensional FDM grid

The FDf predicts the values at these points at a new time level based upon the known values at the previous time level of points nearby in the x and y-directions .

The extra subscript, k, is introduced to denote the spatial position in the y-direction of the grid point.

The problem of interest now is the solution of the two-dimensional advection equation

$$\frac{\partial \hat{\tau}}{\partial t} + u\frac{\partial \hat{\tau}}{\partial x} + v\frac{\partial \hat{\tau}}{\partial y} = 0, \tag{4.1}$$

where u and v represent the components of the velocity in the x and y directions respectively, and so a two-dimensional velocity field with a known analytic solution is required as a standard against which the performance of different FDMs can be judged.

If we have a one-dimensional velocity field u for which we have an analytic solution $\hat{\tau}_x$ say, and similarly a one-dimensional velocity field v with analytic solution $\hat{\tau}_y$, then the product $\hat{\tau}_x\hat{\tau}_y$ is a solution to the two-dimensional velocity field with components u and v. Note that $\hat{\tau}_y$ is independent of x, and $\hat{\tau}_x$ is independent of y. For example, if

$$\frac{\partial \hat{\tau}_x}{\partial t} + u\frac{\partial \hat{\tau}_x}{\partial x} = 0 \tag{4.2}$$

$$\frac{\partial \hat{\tau}_y}{\partial t} + v\frac{\partial \hat{\tau}_y}{\partial y} = 0. \tag{4.3}$$

then for the two-dimensional advection equation,

$$\begin{aligned}
&\tfrac{\partial}{\partial t}\left(\hat{\tau}_x\hat{\tau}_y\right) + u\tfrac{\partial}{\partial x}\left(\hat{\tau}_x\hat{\tau}_y\right) + v\tfrac{\partial}{\partial y}\left(\hat{\tau}_x\hat{\tau}_y\right) \\
= \;& \hat{\tau}_x\left(\tfrac{\partial \hat{\tau}_y}{\partial t} + v\tfrac{\partial \hat{\tau}_y}{\partial y} + u\tfrac{\partial \hat{\tau}_y}{\partial x}\right) + \hat{\tau}_y\left(\tfrac{\partial \hat{\tau}_x}{\partial t} + u\tfrac{\partial \hat{\tau}_x}{\partial x} + v\tfrac{\partial \hat{\tau}_x}{\partial y}\right) \\
= \;& \hat{\tau}_x u\tfrac{\partial \hat{\tau}_y}{\partial x} + \hat{\tau}_y v\tfrac{\partial \hat{\tau}_x}{\partial y} \\
= \;& 0.
\end{aligned}$$

Therefore, the product of the exact one-dimensional solutions is an exact solution to the two-dimensional problem.

So, the velocity fields used in the one-dimensional tests can be used in the two-dimensional tests. That is, for space-varying two-dimensional advection, if

$$u = m\left(x + a\right) \tag{4.4}$$

and

$$v = m\left(y + a\right) \tag{4.5}$$

then the two-dimensional advection equation has an exact solution given by

$$\hat{\tau}\left(x, y, t\right) = e^{-k\left[\left((x+a)e^{mt} - x_o - a\right)^2 + \left((y+a)e^{-mt} - y_o - a\right)^2\right]}. \tag{4.6}$$

A similar solution exists for time-varying two-dimensional advection, and will be discussed later.

4.2 Two-Dimensional Stencils

In the explicit approach which uses two-dimensional stencils to solve two-dimensional problems, the value of the function at a point in the new time level is immediately determined form the values of the function at previous time levels. For example, a general two-dimensional stencil might be

$$\begin{array}{rlll} \tau_{j,k}^{n+1} = & a\tau_{j-1,k-1}^{n} & +b\tau_{j,k-1}^{n} & +c\tau_{j+1,k-1}^{n} \\ & +d\tau_{j-1}^{n} & +e\tau_{j,k}^{n} & +f\tau_{j+1,k}^{n} \\ & +g\tau_{j-1,k+1}^{n} & +h\tau_{j,k+1}^{n} & +i\tau_{j+1,k+1}^{n} \end{array} \tag{4.7}$$

4.2.1 Space-varying velocities in two dimensions

The standard test for the case of a space-varying velocity field in two dimensions is to have symmetry about the line $y = x$ and with the same velocity field as before in both the x and y directions, that is,

$$u = -4\,(x - 6) \tag{4.8}$$
$$v = -4\,(y - 6)\,, \tag{4.9}$$

The programs for testing one-dimensional space-varying advection were modified to cater for the additional dimension.

The methods tested were the Upwind $(1, 3)$ and FTCS $(1, 9)$ two-level methods, and the Leapfrog $(1, 4, 1)$ and Compact $(1, 4, 1)$ three-level methods.

The general formula (4.7) represents a two-level method, with its stencil involving grid points at the $(n+1)^{th}$ and n^{th} time levels. A three-level method involves grid points at the $(n-1)^{th}$ time level also. Such methods require initial values at two time levels, which was achieved in the tests by supplying the exact values for these required levels.

As only four of these methods are tested, they will be analysed together, rather than by further subgrouping into two-level and three-level methods. None of their FDfs have a modified version.

The two-dimensional stencils produced a variety of results. They all had restricted stability. Only the Compact $(1, 4, 1)$ method was stable for $c = 0.96$. All methods are stable for $c = 0.3$ and $c = 0.6$ with the exception that the Leapfrog $(1, 4, 1)$ method is stable only at $c = 0.3$.

In Tables 4.1 and 4.2 we see that the Upwind $(1, 3)$ method is the worst performer. Its peak height is less than 0.3, due to large amounts of numerical diffusion. Characteristically, it is the only method that does not produce negative values. Its contour plot (see Figure 4.2) is interesting as it shows the peak is diffused most in the direction perpendicular to the motion of the peak.

The contour plot of the Leapfrog $(1, 4, 1)$ method (Figure 4.3) shows small oscillations that trail the peak in both the x and y directions. We see from the cross sectional plot that the peak trails the exact solution. Amplitude response is however reasonable.

Figure 4.2: Upwind method with increased diffusion perpendicular to trajectory.

Figure 4.3: Leapfrog method with oscillations in x and y-directions.

Space-varying velocity in two dimensions Two-dimensional stencils Courant number = 0.3					
method	ave.error	min.height	max.height	peak shift	cpu
Upwind (1,3)	0.6419	0.0000	0.2817	-0.0069	21.2s
Leapfrog (1,4,1)	0.2944	-0.0860	0.9079	-0.0499	25.6s
FTCS (1,9)	0.2841	-0.0802	0.8875	-0.0489	54.3s
Compact (1,4,1)	0.0989	-0.0426	0.9659	0.0188	27.8s

Table 4.1: Results of tests of two-dimensional formulae in space-varying velocities for $c = 0.3$.

Space-varying velocity in two dimensions Two-dimensional stencils Courant number = 0.96					
method	ave.error	min.height	max.height	peak shift	cpu
Compact (1,4,1)	0.0726	-0.0007	0.9954	-0.0148	8.3s

Table 4.2: Results of tests of two-dimensional formulae in space-varying velocities for $c = 0.96$.

The FTCS $(1, 9)$ method produces results that are qualitatively similar to those of the Leapfrog $(1, 4, 1)$ method.

The Compact $(1, 4, 1)$ method is the best performer. Although it has a hiccough at the base of the leading edge of the peak for small Courant numbers, (see Figure 4.4), its performance for $c = 0.96$ is very reasonable. Figure 4.5 shows that at $c \sim 1$ this method produces a good approximation to the exact solution. In particular, the minimum value of the approximation is only just negative, and the maximum value is nearly one.

4.2.2 Time-varying velocities in two dimensions

Again the standard test for the case of a time-varying velocity field in two dimensions is to have symmetry in the line $y = x$ and with the same velocity field as before, that is,

$$u = v = U \sin \omega t. \tag{4.10}$$

The exact solution for the two-dimensional advection equation using this velocity field is

$$\hat{\tau}\,(x, y, t) = e^{-k\left[\left(x - x_o - 2U\omega^{-1}\sin^2\left(\frac{\omega t}{2}\right)\right)^2 + \left(y - y_o - 2U\omega^{-1}\sin^2\left(\frac{\omega t}{2}\right)\right)^2\right]}. \tag{4.11}$$

In this case we have a Gaussian peak that oscillates back and forth along the line $y = x$ in a similar manner to the peak oscillation along the *x-axis* in the one-dimensional case.

Figure 4.4: The Compact (1,4,1) has a hiccough for small values of c.

Figure 4.5: The Compact (1,4,1) formula performs well for $c \sim 1$.

Time-varying velocity in two dimensions Two-dimensional stencils Final time = 4.0 Courant number = $\pi/30$					
method	ave.error	min.height	max.height	peak shift	cpu
Upwind (1,3)	0.2191	0.0000	0.0177	0.0093	158.2s
Compact (1,4,1)	0.1062	-0.1039	0.7084	0.0034	337.0s
Leith (1,3)	0.0798	-0.0232	0.5418	0.0000	169.9s
Leapfrog (1,4,1)	0.0000	0.0000	1.0000	0.0000	191.9s

Table 4.3: Results of tests of two-dimensional formulae in time-varying velocities for $c = \pi/30$ and $t = 4$.

Time-varying velocity in two dimensions Two-dimensional stencils Final time = 5.0 Courant number = $\pi/30$					
method	ave.error	min.height	max.height	peak shift	cpu
Leapfrog (1,4,1)	0.6303	-0.2331	0.4165	-0.0944	229.8s
Leith (1,3)	0.2807	-0.1374	0.3587	-0.0765	209.8s
Upwind (1,3)	0.1871	0.0000	0.0134	0.0246	201.6s
Compact (1,4,1)	0.1615	-0.2697	0.6124	0.0521	425.1s

Table 4.4: Results of tests of two-dimensional formulae in time-varying velocities for $c = \pi/30$ and $t = 5$.

The two-dimensional stencil methods that were tested using the space-varying velocity problem were again put to the test using this velocity field. Results are displayed in Tables 4.3 and 4.4.

Again, stability proved to be a problem, with only the Compact $(1, 4, 1)$ method being stable for $c = \pi/4$, $\simeq 0.785$.

The Upwind $(1, 3)$ method performed disastrously, achieving a maximum height of less than 0.02. Diffusion has destroyed all information about the presence of the peak. Figure 4.6 shows this effect graphically.

The Compact $(1, 4, 1)$ method developed erratic wiggles at $t = 4$ which became a large oscillation at $t = 5$. Amplitude response was poor as was peak positioning at $t = 5$.

While the FTCS $(1, 9)$ method has excellent peak positioning at $t = 4$, its amplitude response is poor. At $t = 5$ it develops trailing oscillations and the peak trails also.

The Leapfrog $(1, 4, 1)$ method is perfect at $t = 4$ (see Figure 4.7). However, at $t = 5$ it develops a trailing peak and many trailing oscillations. Its contour plot (Figure 4.8) makes these oscillations obvious.

Also of interest is the graph produced for the Courant number of $\pi/4$ at time $t = 4$. Figure 4.9 captures the Leapfrog $(1, 4, 1)$ method as errors are growing, and shortly after this the method

Figure 4.6: Diffusion produced to disastrous effect by the Upwind $O\{1\}$ formula.

Figure 4.7: Excellent performance of the Leapfrog (1,4,1) formula at $t = 4$.

Figure 4.8: Large oscillations in the Leapfrog (1,4,1) formula at $t = 5$.

explodes.

As a summary of FDMs using two-dimensional stencils, they appear to be of restricted use. They are unstable for a large range of the Courant number, and those methods tested produce poor results even when they are stable. In short, one would hope that there is a better alternative.

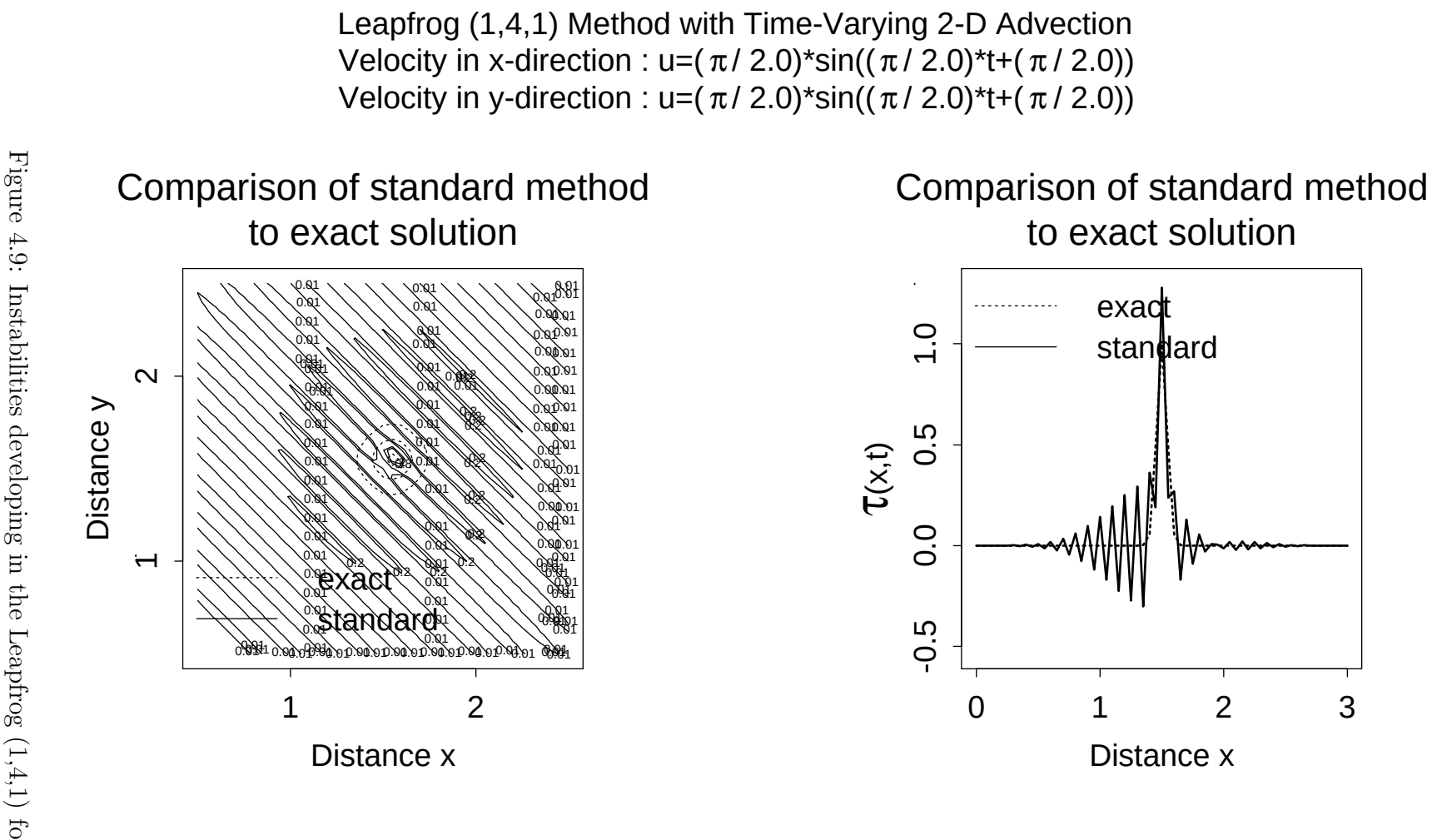

Figure 4.9: Instabilities developing in the Leapfrog (1,4,1) formula.

Chapter 5

Two-Dimensional Advection Tests using Time Splitting

5.1 Time Splitting

An alternative method of solving two-dimensional problems is to use "time splitting", also known as the method of "fractional steps".

In time splitting, the values at the grid points are determined in two steps over each time step.

The first step is to evaluate the points on the grid at the time $t + (\Delta t)/2$ by running the one-dimensional stencil in the x-direction for each value of y.

The second step is to use these values to evaluate the points on the grid by using the values at time $t + (\Delta t)/2$ and running the one-dimensional stencil in the y-direction for each value of x.

Thus all values are now known at the new time level $t + \Delta t$, as required.

Hence time splitting can be used to solve two-dimensional problems using two applications of one-dimensional formulae. An important point is that the new time level values may depend upon which direction is chosen for the first step. If we had chosen to evaluate the points at time $t + (\Delta t)/2$ by running in the y-direction and then at time $t + \Delta t$ by running in the x-direction the value at the time level $t + \Delta t$ might not be the same.

Consider the time-splitting process, with the x-direction being the first direction to be processed. We have

$$\tau_{j,k-1}^{n+1/2} = a_{-1}\tau_{j-1,k-1}^{n} + b_{-1}\tau_{j,k-1}^{n} + c_{-1}\tau_{j+1,k-1}^{n} \tag{5.1}$$

$$\tau_{j,k}^{n+1/2} = a_{0}\tau_{j-1,k}^{n} + b_{0}\tau_{j,k}^{n} + c_{0}\tau_{j+1,k}^{n} \tag{5.2}$$

$$\tau_{j,k+1}^{n+1/2} = a_{1}\tau_{j-1,k+1}^{n} + b_{1}\tau_{j,k+1}^{n} + c_{1}\tau_{j+1,k+1}^{n} \tag{5.3}$$

and in the y-direction

$$\tau_{j,k}^{n+1} = d_{0}\tau_{j,k-1}^{n+1/2} + e_{0}\tau_{j,k}^{n+1/2} + f_{0}\tau_{j,k+1}^{n+1/2}. \tag{5.4}$$

So

$$\begin{array}{rlll} \tau_{j,k}^{n+1} = & a_{-1}d_0\tau_{j-1,k-1}^n & +b_{-1}d_0\tau_{j,k-1}^n & +c_{-1}d_0\tau_{j+1,k-1}^n \\ & +a_0e_0\tau_{j-1,k}^n & +b_0e_0\tau_{j,k}^n & +c_0e_0\tau_{j+1,k}^n \\ & +a_1f_0\tau_{j-1,k+1}^n & +b_1f_0\tau_{j,k+1}^n & +c_1f_0\tau_{j+1,k+1}^n \end{array} \tag{5.5}$$

If however the y-direction was the first direction to be processed then we would find

$$\begin{array}{rlll} \tau_{j,k}^{n+1} = & a_0d_{-1}\tau_{j-1,k-1}^n & +b_0d_0\tau_{j,k-1}^n & +c_0d_1\tau_{j+1,k-1}^n \\ & +a_0e_{-1}\tau_{j-1,k}^n & +b_0e_0\tau_{j,k}^n & +c_0e_1\tau_{j+1,k}^n \\ & +a_0f_{-1}\tau_{j-1,k+1}^n & +b_0f_0\tau_{j,k+1}^n & +c_0f_1\tau_{j+1,k+1}^n \end{array} \tag{5.6}$$

and we see the two methods are not the same.

Hence the final values may be dependent upon the choice of initial step direction. However, this undesirable discrepancy diminishes as Δx, Δy and Δt tend to zero.

In the problems that we consider in this chapter, this difference does not occur, as u is dependent only upon x and t, and v is dependent only upon y and t.

The programs for testing one-dimensional advection for space-varying velocity fields and for time-varying velocity fields were adapted to facilitate time splitting and hence enabled analysis of two-dimensional advection FDMs. The graphical routines were also adapted to give contour plots for better interpretation of results.

5.2 Space-Varying Velocities

The test previously used for space-varying velocity fields in two dimensions for two-dimensional stencils will be used here for time split methods. This enables ready comparison of the use of time splitting to fully two-dimensional formulae.

5.2.1 Explicit methods .:.

As the test is the same as for two-dimensional stencils, and the methods to be tested are the same as in the one-dimensional case, no further discussion of the test is necessary. Results are displayed in Tables 5.1 and 5.2.

As is to be expected, the Upwind $O\{1\}$ method performs poorly, due to the large amount of numerical diffusion it produces.

Upwind $O\{2\}$ shows similar results to its one-dimensional analogue, with a leading peak and amplitude response that becomes very reasonable as c approaches 1. It still possesses a large leading oscillation which produces some negative values, but this occurs in the x and y directions, and not along the diagonal line $y = x$. Note the shape of the contours the method produces in Figure 5.1.

Again, Leith's method performs similarly to its one-dimensional counter part. Of interest is the shape of its contours. They are very similar to Upwind $O\{2\}$, except that they are rotated by an angle of 180 degrees. See Figure 5.2.

Space-varying velocity in two dimensions Explicit methods Courant number = 0.3						
method		ave.error	min.height	max.height	peak shift	cpu
Upwind $O\{1\}$		0.6489	0.0000	0.2760	-0.0105	33.1s
Upwind $O\{2\}$	(mod)	0.3280	-0.0804	0.7764	0.0551	72.9s
Upwind $O\{2\}$		0.3232	-0.0809	0.7796	0.0532	67.0s
Leith (1,3)		0.2843	-0.0803	0.8893	-0.0512	30.4s
Leith (1,3)	(mod)	0.2777	-0.0794	0.8857	-0.0494	37.4s
Fromms	(mod)	0.0800	-0.0097	0.8672	0.0088	85.3s
Fromms		0.0770	-0.0099	0.8719	0.0068	73.2s
Martins $O\{1\}$		0.0608	-0.0030	0.8965	-0.0040	97.0s
Martins $O\{\sim 3\}$	(mod)	0.0604	-0.0029	0.8918	-0.0022	115.6s
Martins $O\{2\}$	(mod)	0.0598	-0.0030	0.8931	-0.0022	106.3s
Rusanovs MAE		0.0231	-0.0001	0.9895	-0.0072	96.4s
Rusanovs MAE	(mod)	0.0225	-0.0001	0.9777	-0.0068	119.4s

Table 5.1: Results of explicit methods in the two-dimensional time-split test with space-varying fluid velocities using $c = 0.3$.

Space-varying velocity in two dimensions Explicit methods Courant number = 0.96						
method		ave.error	min.height	max.height	peak shift	cpu
Upwind $O\{1\}$		0.3576	0.0000	0.5081	-0.0022	10.5s
Leith (1,3)		0.1478	-0.0176	0.9008	-0.0312	9.9s
Leith (1,3)	(mod)	0.1278	-0.0183	0.9123	-0.0253	11.9s
Upwind $O\{2\}$	(mod)	0.1059	-0.0157	0.9547	0.0228	22.6s
Upwind $O\{2\}$		0.0922	-0.0157	0.9401	0.0169	20.4s
Fromms		0.0511	-0.0007	0.9506	-0.0086	23.5s
Rusanovs MAE	(mod)	0.0447	0.0000	0.9317	-0.0064	37.1s
Martins $O\{1\}$		0.0438	-0.0003	0.9512	-0.0058	31.1s
Rusanovs MAE		0.0380	0.0000	0.9908	-0.0084	31.2s
Martins $O\{\sim 3\}$	(mod)	0.0288	-0.0003	0.9339	0.0005	35.9s
Fromms	(mod)	0.0276	-0.0006	0.9510	-0.0032	27.5s
Martins $O\{2\}$	(mod)	0.0230	-0.0003	0.9500	-0.0003	34.5s

Table 5.2: Results of explicit methods in the two-dimensional time-split test with space-varying fluid velocities using $c = 0.96$.

Figure 5.1: The deformed contours of the Upwind $O\{2\}$ formula.

Figure 5.2: The different direction of contours of the Leith formula compared to Upwind $O\{2\}$.

Space-varying velocity in two dimensions Implicit methods Courant number = 0.3					
method	ave.error	min.height	max.height	peak shift	cpu
Crank-Nicolson (3,3)	0.3205	-0.0997	0.9050	-0.0549	55.9s
Leonard-Noye (3,3)	0.2223	-0.0598	0.9268	-0.0432	84.1s
Box (2,2)	0.1179	-0.0138	0.9678	0.0236	74.3s
Noye and Tan (2,3)	0.0464	-0.0011	0.9591	-0.0055	124.3s
LFE Crank-Nicolson type	0.0198	-0.0009	1.0182	-0.0035	62.1s
Noye (3,3)	0.0181	-0.0011	1.0172	-0.0018	84.1s

Table 5.3: Results of implicit methods in the two-dimensional time-split test with space-varying fluid velocities using $c = 0.3$.

Fromm's method becomes quite a reasonable approximation for $c = 0.96$, although its amplitude response at $c = 0.3$ is not as good as is desired. Recall that it is developed from an average of the Upwind $O\{2\}$ and Leith's method. It is surprising then that this method produces such accurate results (see Figure 5.3). Also, notice that the modified version of this method performs considerably better than the standard method at $c = 0.96$ due to the much more exact peak positioning.

Although little difference can be seen between Martins $O\{1\}$ and $O\{2\}$ methods, the latter is the better performer, again due to better location of the peak. Martins $O\{\sim 3\}$ is not an improvement upon the $O\{2\}$ method, as it does not improve further upon the peak position, but does introduce more artificial diffusion. However, it is a very reasonable performer overall.

Rusanov's MAE method again does the best job for $c = 0.3$. Its amplitude response is by far the best, and its peak positioning is quite good. The modified version detracts from the attributes of the standard version for $c = 0.96$, as it too introduces artificial diffusion (see Figure 5.4).

With the exception of Fromm's method, the modifications to improve order of accuracy with variable velocity do little to improve the accuracy of the methods.

5.2.2 Implicit methods :::

From Tables 5.3 and 5.4 we see the poorest result is produced by the Crank-Nicolson method, which has a large negative trailing oscillation, and trailing peak. See Figure 5.5.

The Leonard-Noye method is qualitatively similar, except that the negative oscillation is smaller in magnitude, and the peak height is closer to one. Both these methods produce contour plots similar to Leith.

The Box method on the other hand performs similarly to the Upwind $O\{2\}$ method, except that its peak height is quite reasonable.

Figure 5.3: Fromm's method produces reasonable results.

Figure 5.4: Artificial diffusion introduced by the modification to Rusanov's formula.

Figure 5.5: Large trailing oscillations in the Crank-Nicolson (3,3) method.

Space-varying velocity in two dimensions Implicit methods Courant number = 0.96					
method	ave.error	min.height	max.height	peak shift	cpu
Crank-Nicolson (3,3)	0.3912	-0.1281	0.8838	-0.0623	18.5s
Leonard-Noye (3,3)	0.1425	-0.0203	0.9557	-0.0310	25.3s
LFE Crank-Nicolson type	0.0765	-0.0048	1.0031	-0.0166	19.6s
Box (2,2)	0.0596	-0.0009	0.9726	0.0083	23.5s
Noye and Tan (2,3)	0.0489	-0.0001	0.9710	-0.0068	38.6s
Noye (3,3)	0.0125	-0.0001	1.0138	-0.0012	25.0s

Table 5.4: Results of implicit methods in the two-dimensional time-split test with space-varying fluid velocities using $c = 0.96$.

The LFE Crank-Nicolson type produces good results for $c = 0.3$, but begins to deteriorate as c increases (see Figure 5.6).

The Noye and Tan method produces good results throughout, although it requires the most CPU time.

Excellent results are produced by the Noye $(3, 3)$ method, (Figure 5.7), with very good peak positioning and only small negative values (for c=0.96 the minimum value is -0.0001). This method once again produces a peak height in excess of one.

Yet again, the implicit methods outperform the explicit methods. Moreover, time splitting produces far better results than the two-dimensional stencils.

5.3 Time-Varying Velocities

The test used in this section is the same as that used for the time-varying velocity fields in two dimensions for two-dimensional stencils.

5.3.1 Explicit methods .:.

Results for the explicit methods are shown in Tables 5.5 and 5.6. As is by now to be expected, the Upwind $O\{2\}$ method is almost flat at all values of the Courant number, and for $t = 4$ awards.

The Upwind $O\{1\}$ method performs almost as poorly, with numerical diffusion attenuating its amplitude to 0.15 by time $t = 4$. The modified version contains even more of this diffusion.

Leith has better amplitude response at $t = 4$, achieving heights of approximately 0.5, but at $t = 5$ develops its characteristic trailing oscillations and trailing attenuated peak. The modified version is a slight improvement at $t = 5$, largely because the diffusion it creates aids in damping these oscillations.

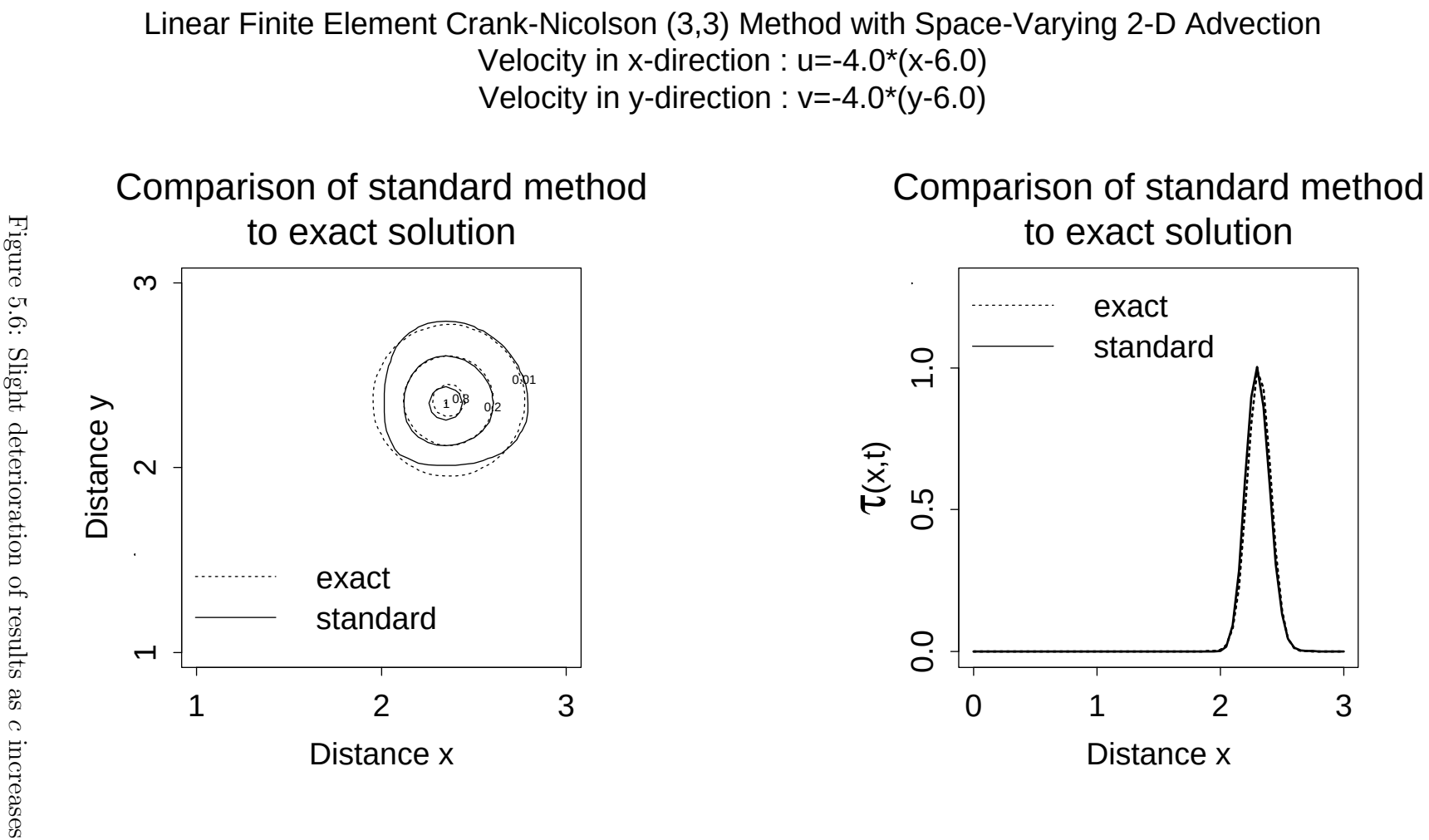

Figure 5.6: Slight deterioration of results as c increases.

Figure 5.7: Excellent results produced by the Noye (3,3) method.

Time-varying velocity in two dimensions Explicit methods Final time = 4.0 Courant number = $\pi/30$						
method		ave.error	min.height	max.height	peak shift	cpu
Upwind $O\{1\}$		0.2282	0.0000	0.0181	0.0057	215.8s
Upwind $O\{2\}$	(mod)	0.2197	-0.0070	0.1530	0.0100	199.1s
Upwind $O\{2\}$		0.2188	-0.0071	0.1554	0.0101	206.9s
Fromms	(mod)	0.1856	-0.0173	0.2136	0.0002	193.1s
Fromms		0.1855	-0.0183	0.2184	0.0002	196.2s
Martins $O\{1\}$		0.1754	-0.0189	0.2473	0.0000	190.6s
Martins $O\{\sim 3\}$	(mod)	0.1740	-0.0179	0.2411	0.0000	
Martins $O\{2\}$	(mod)	0.1740	-0.0179	0.2411	0.0000	188.7s
Leith (1,3)	(mod)	0.0845	-0.0186	0.5095	0.0000	167.4s
Leith (1,3)		0.0800	-0.0228	0.5404	0.0000	168.1s
Rusanovs MAE	(mod)	0.0490	-0.0278	0.6946	0.0000	183.3s
Rusanovs MAE		0.0482	-0.0307	0.7288	0.0000	183.5s

Table 5.5: Results of explicit methods in the two-dimensional time-split test with time-varying fluid velocities using $t = 4$ and $c = \pi/30$.

Time-varying velocity in two dimensions Explicit methods Final time = 5.0 Courant number = $\pi/30$						
method		ave.error	min.height	max.height	peak shift	cpu
Leith (1,3)		0.2799	-0.1374	0.3590	-0.0769	210.1s
Leith (1,3)	(mod)	0.2697	-0.1245	0.3439	-0.0775	207.6s
Upwind $O\{2\}$		0.2373	0.0000	0.1345	0.0559	278.5s
Upwind $O\{2\}$	(mod)	0.2373	0.0000	0.1322	0.0527	243.7s
Upwind $O\{1\}$		0.1981	0.0000	0.0140	0.0184	259.7s
Fromms		0.1867	-0.0227	0.1957	0.0143	240.9s
Fromms	(mod)	0.1862	-0.0211	0.1913	0.0118	243.0s
Martins $O\{2\}$	(mod)	0.1785	-0.0196	0.2181	-0.0019	247.6s
Martins $O\{\sim 3\}$	(mod)	0.1785	-0.0196	0.2182	-0.0019	
Martins $O\{1\}$		0.1784	-0.0211	0.2246	0.0004	248.5s
Rusanovs MAE		0.1316	-0.1074	0.5885	-0.0202	228.6s
Rusanovs MAE	(mod)	0.1263	-0.1003	0.5563	-0.0204	230.9s

Table 5.6: Results of explicit methods in the two-dimensional time-split test with time-varying fluid velocities using $t = 5$ and $c = \pi/30$.

Time-varying velocity in two dimensions Implicit methods Final time = 4.0 Courant number = $\pi/30$					
method	ave.error	min.height	max.height	peak shift	cpu
Noye and Tan (2,3)	0.1377	-0.0249	0.3570	0.0000	699.9s
Box (2,2)	0.1213	-0.0727	0.6874	0.0046	517.0s
Noye (3,3)	0.0000	0.0000	1.0000	0.0000	527.6s
Leonard-Noye (3,3)	0.0000	0.0000	1.0000	0.0000	517.1s
LFE Crank-Nicolson type	0.0000	0.0000	1.0000	0.0000	453.8s
Crank-Nicolson (3,3)	0.0000	0.0000	1.0000	0.0000	408.9s

Table 5.7: Results of implicit methods in the two-dimensional time-split test with time-varying fluid velocities using $t = 4$ and $c = \pi/30$.

Fromm's method also shows large amounts of diffusion. See Figure 5.8.

Yet again the modified versions of the Martins FDf perform almost identically, and also similarly to the standard version. These all have maximum values of less than 0.25.

The Rusanov MAE method gives the best results, managing to maintain its peak height in excess of 0.5 at time $t = 5$. It does however develop trailing oscillations in the x and y directions, and also a trailing peak as shown in Figure 5.9.

The performance of these explicit methods may be summarised by saying that artificial diffusion is abundant in this two-dimensional test. As a result, the modified versions (introducing more diffusion) are of little value. However, for those methods that develop oscillations at $t = 5$, the modification can give some small improvement. However, none of these methods could be said to perform well in this test.

5.3.2 Implicit methods :::

Tables 5.7 and 5.8 display results obtained by the implicit methods. The Box method produces large leading and trailing oscillations with associated negative values at $t = 4$, but only leading at $t = 5$. Figure 5.10 shows the peak leading at $t = 5$. Amplitude response is reasonable at $t = 4$, but is poor at the extremities of the trajectory.

The Noye and Tan method is adversely affected by large amounts of artificial diffusion (see Figure 5.11). Its peak is well centred at $t = 5$ for $c = \pi/30$, and its oscillations are only small in magnitude.

As before the Crank-Nicolson, Leonard-Noye, LFE Crank-Nicolson type and the Noye $(3, 3)$ methods all produce similar results. They all give perfect results for $t = 4$, yet have large trailing oscillations at $t = 5$. All are affected by the increased amount of diffusion inherent in the modified versions, but Noye $(3, 3)$ is the most resilient, maintaining a peak height in excess of 0.8. See Figures 5.12 and 5.13.

Figure 5.8: Large amounts of numerical diffusion inherent in Fromm's formula.

Figure 5.9: This test produces diffusion in Rusanov's method.

Figure 5.10: The Box (2,2) method with erratic leading oscillations.

Figure 5.11: Diffusion inherent in the Noye and Tan (3,3) method.

Time-varying velocity in two dimensions Implicit methods Final time = 5.0 Courant number = $\pi/30$					
method	ave.error	min.height	max.height	peak shift	cpu
Crank-Nicolson (3,3)	0.6052	-0.2308	0.4127	-0.0910	134.8s
Leonard-Noye (3,3)	0.5286	-0.2302	0.4507	-0.0803	152.6s
Box (2,2)	0.2538	-0.1945	0.5274	0.0585	659.7s
Noye and Tan (2,3)	0.1457	-0.0239	0.3282	0.0032	763.5s
LFE Crank-Nicolson type	0.1205	-0.0905	0.8013	-0.0105	142.9s
Noye (3,3)	0.1171	-0.0849	0.8072	-0.0098	624.0s

Table 5.8: Results of implicit methods in the two-dimensional time-split test with time-varying fluid velocities using $t = 5$ and $c = \pi/30$.

These four methods outperform the explicit methods at $t = 4$, but are similar at $t = 5$. The large trailing oscillations at the ends of the spatial range of the oscillation of the peak are their major problem.

5.4 The Solution to the Two-Dimensional Finite-Difference Formula using Time-Splitting

The contour plots for both the space-varying and the time-varying two-dimensional tests, and in particular for the implicit methods are qualitatively similar to the results for the one-dimensional tests. Of note is the fact that the trailing and leading oscillations occur mainly along the line

$$x = x_p \tag{5.7}$$

and

$$y = y_p, \tag{5.8}$$

where (x_p, y_p) is the location of the peak. That is, these trailing and leading oscillations are not occurring in the direction of travel of the peak to as large an extent as they are in the x and y directions.

Note that, the numerical solution to the two-dimensional problem is the product of two one-dimensional numerical solutions. That is, if $\tau_1(x,t)$ is the numerical solution of the FDf for the one-dimensional problem in the x-direction, and $\tau_2(y,t)$ is the numerical solution of the FDf for the one-dimensional problem in the y direction, then the numerical solution to the FDf in the two-dimensional problem $\tau(x,y,t)$ is given by

$$\tau\left(x,y,t\right) = \tau_1\left(x,t\right)\tau_2\left(y,t\right). \tag{5.9}$$

This statement is not to be confused with the similar point made in Section 4.1, which involved the exact solution of the advection equation, as opposed to the numerical solution of the FDf.

Figure 5.12: Large trailing oscillations produced by the Crank-Nicolson (3,3) method.

Figure 5.13: Smaller trailing oscillations produced by the Noye (3,3) method.

Equation (5.9) can easily be illustrated for the maximum and minimum peak heights obtained in the two-dimensional case.

We would expect

$$\max \tau (x, y, t) = \max \tau_1 (x, t) \times \max \tau_2 (y, t) \tag{5.10}$$

and

$$\min \tau (x, y, t) = \min \left[\max \tau_1 (x, t) \times \min \tau_2 (y, t) , \min \tau_1 (x, t) \times \max \tau_2 (y, t) \right] \tag{5.11}$$

As the two-dimensional tests used here have symmetry between the x and y directions we expect

$$\max \tau (x, y, t) = \left[\max \tau_1 (x, t) \right]^2 \tag{5.12}$$

and

$$\min \tau (x, y, t) = \max \tau_1 (x, t) \times \min \tau_1 (x, t) . \tag{5.13}$$

Indeed this is the case, as the reader can verify using any of the results given herein.

The important ramification follows. Consider a multi-dimensional problem (specifically in three-dimensional space) in which the components of the fluid velocities are dependent only upon the position in that direction (the direction in which the component acts) and also may be dependent upon time. Using a one-dimensional FDf and time splitting to solve the multi-dimensional problem gives the same solution as the alternative approach of solving each component as a one-dimensional problem, and multiplying the resultant FDf solution.

So, once the behaviour of the particular FDf is understood in one-dimensional space, then it behaviour in many higher-dimensional problems can be predicted.

This has the potential of increasing the speed of programs that produce solutions to problems of this sort.

5.5 Comparison of Two-Dimensional Stencils and Time Splitting

Another point of interest is the similarity between some of the results produced by time splitting, and some produced by using a fully two-dimensional formula. For example, Leith's method with time splitting and the FTCS $(1, 9)$ two-dimensional stencil give similar results.

We saw in Section 5.1 that the process of time splitting effectively produces a two-dimensional stencil.

Recall that the FDf for Leith's method is

$$\tau_j^{n+1} = \frac{1}{2} \left(c + c^2 \right) \tau_{j-1}^n + \left(1 - c^2 \right) \tau_j^n - \frac{1}{2} \left(c - c^2 \right) \tau_{j+1}^n \tag{5.14}$$

If we produce the two-dimensional stencil equivalent of Leith's method when time split we

find

$$\begin{aligned}\tau_j^{n+1} = \quad & \tfrac{1}{4}\left(c_x^2c_y^2 + c_xc_y^2 - c_x^2c_y - c_xc_y\right)\tau_{j-1,k+1}^n \\ & +\tfrac{1}{2}\left(-c_x^2c_y^2 - c_y^2 + c_x^2c_y - c_y\right)\tau_{j,k+1}^n \\ & +\tfrac{1}{4}\left(c_x^2c_y^2 - c_xc_y^2 - c_x^2c_y + c_xc_y\right)\tau_{j+1,k+1}^n \\ & +\tfrac{1}{2}\left(-c_x^2c_y^2 - c_xc_y^2 + c_x^2 + c_x\right)\tau_{j-1,k}^n \\ & +\left(c_x^2c_y^2 - c_y^2 - c_x^2 + 1\right)\tau_{j,k}^n \\ & +\tfrac{1}{2}\left(-c_x^2c_y^2 + c_xc_y^2 + c_x^2 - c_x\right)\tau_{j+1,k}^n \\ & +\tfrac{1}{4}\left(c_x^2c_y^2 + c_xc_y^2 + c_x^2c_y + c_xc_y\right)\tau_{j-1,k-1}^n \\ & +\tfrac{1}{2}\left(-c_x^2c_y^2 + c_y^2 - c_x^2c_y + c_y\right)\tau_{j,k-1}^n \\ & +\tfrac{1}{4}\left(c_x^2c_y^2 - c_xc_y^2 + c_x^2c_y - c_xc_y\right)\tau_{j+1,k-1}^n\end{aligned} \tag{5.15}$$

which to a second-order approximation is

$$\begin{aligned}\tau_j^{n+1} = \quad & \tfrac{1}{4}\left(-c_xc_y\right)\tau_{j-1,k+1}^n & +\tfrac{1}{2}\left(-c_y^2 - c_y\right)\tau_{j,k+1}^n & & +\tfrac{1}{4}\left(c_xc_y\right)\tau_{j+1,k+1}^n \\ & +\tfrac{1}{2}\left(c_x^2 + c_x\right)\tau_{j-1,k}^n & +\left(-c_y^2 - c_x^2 + 1\right)\tau_{j,k}^n & & +\tfrac{1}{2}\left(c_x^2 - c_x\right)\tau_{j+1,k}^n \\ & +\tfrac{1}{4}\left(c_xc_y\right)\tau_{j-1,k-1}^n & +\tfrac{1}{2}\left(c_y^2 + c_y\right)\tau_{j,k-1}^n & & +\tfrac{1}{4}\left(-c_xc_y\right)\tau_{j+1,k-1}^n\end{aligned} \tag{5.16}$$

which is the two-dimensional FTCS $(1, 9)$ formula.

So, when the Courant number c is small, the two methods have similar FDfs, and produce similar results.

Chapter 6

Conclusion

We have seen in Chapter 1 some of the many interesting applications of the advection-diffusion-mortality equation, and have discussed the reasons for concentrating on advection in varying-velocity fields.

Chapter 2 discussed the method by which the accuracy of FDfs can be increased from first order for varying velocities to a higher order of accuracy. It then discussed the aspects of stability for these modified methods.

Chapters 3, 4 and 5 discussed the testing of several FDfs including some that had been modified. These tests showed how these FDfs might behave in velocity fields that vary in space and time. The errors produced by these FDfs can be explained in terms of artificial diffusion and wave speed errors. Artificial diffusion is produced by those FDfs for which the von Neumann amplification factor G is such that

$$|G| < 1. \tag{6.1}$$

This was the case for many methods, in particular the explicit methods and the two-dimensional formulas. Some of the implicit methods did not produce any numerical diffusion what-so-ever.

Wave speed error is associated with component waves of the Gaussian peak traveling at speeds other than $u(x,t)$. This leads to negative values, which in many situations are not physically realistic (for example, a negative concentration of pollutant is impossible). All of the methods discussed here display this phenomenon to some extent, with the exception of the Upwind $O\{1\}$ method (which is of little value due to the large amounts of numerical diffusion that it produces). As it is possible to produce FDfs without numerical diffusion, it is the relative wave speed error that must be improved upon to produce more accurate results.

In general, implicit methods perform considerably better than explicit methods, as many of them have a von Neumann amplification factor of one and so do not produce numerical diffusion.

Two-dimensional formulas discussed in Chapter 4 have greater restrictions on stability, and so are useful only over smaller time steps. As they are understandably more difficult to

develop than one-dimensional methods, they are often of a low order of accuracy. A high-order accuracy two-dimensional formula would require a large and complex computational stencil.

However, two-dimensional formulae can be avoided using the method of time splitting discussed in Chapter 5. With this technique it is possible to use one-dimensional formulae which are more stable, of higher order of accuracy, and may also require less computational effort. Also discussed is the fact that solutions to some two-dimensional problems in advection can be solved as a product of one-dimensional FDf solutions. In retrospect, all the test results produced in that chapter could have been produced using the method discussed in Section 5.4 with considerably less computational effort. However, problems involving velocity fields that are not of the type discussed in in Section 5.4 would still be solved using the method of time splitting as in Chapter 5.

For these tests, the modifications made to the FDfs to improve their accuracy have had little effect. This is because for these velocity fileds, the values of the coefficients of the correcting terms (d and h) are small. This may not be so for all velocity fields, but is the case for many. In particular, a criterion for these corrections to be large is that $\partial u/\partial x$, $\partial u/\partial t$ and higher order derivatives must have large magnitudes (see Equations 2.13 and 2.17). That is, the fluid velocities must change rapidly with respect to the magnitude of the velocities. So a conclusion that can be drawn is that in many cases it is more advantageous to use a method that is of higher order accuracy for constant fluid velocities than it is to modify a lower order formula to be of order of accuracy greater then one for varying fluid velocities.

This suggests that further work is required to develop FDfs that are of higher order accuracy for constant fluid velocities, with the von Neumann amplification factor such that

$$|G| = 1 \tag{6.2}$$

to ensure that no numerical diffusion is introduced. The intention would be to considerably decrease the wave speed errors, and to give consideration to the difficulties introduced by such formulae (for example, calculating values near boundaries, solving systems of equations, and reducing computational effort). These FDfs should be tested using varying-velocity fields, and should include realistic two-dimensional situations such as vortices that cannot be solved as the product of one-dimensional solutions in the manner described in Section 5.4.

Appendix A

Stability analyses

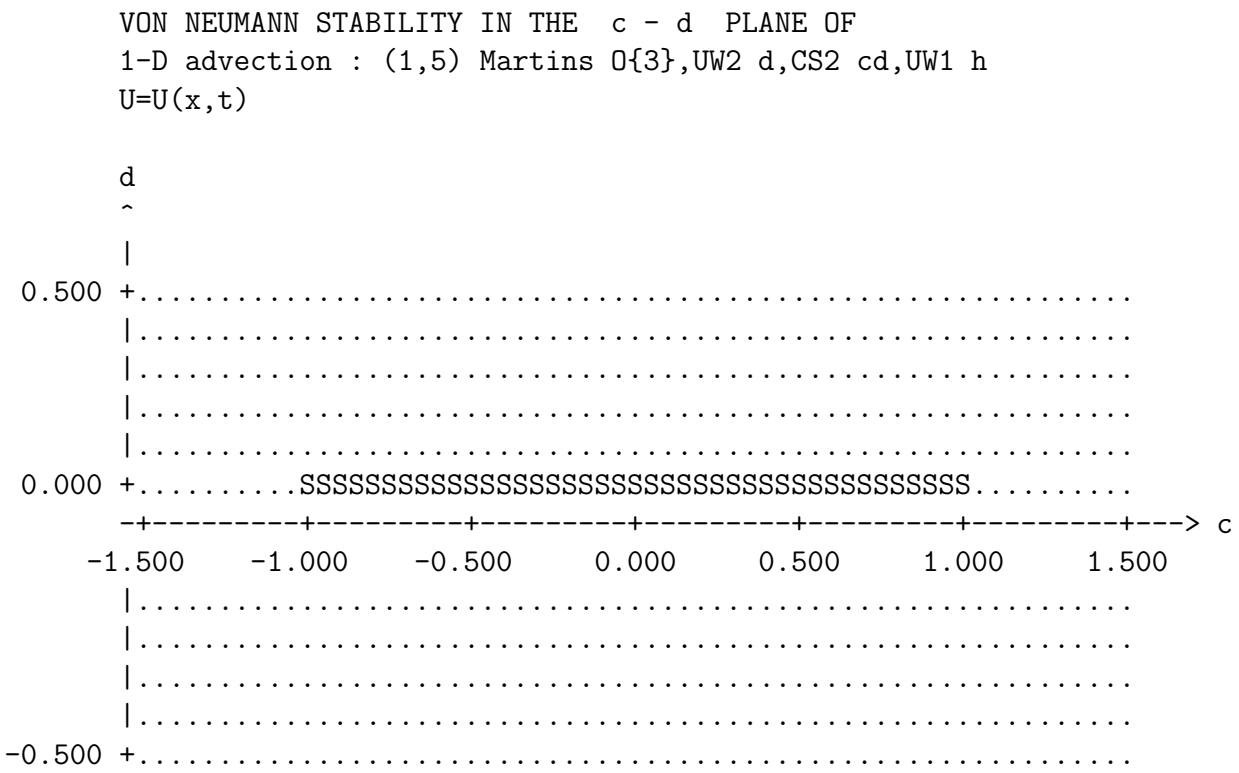

Figure A.1: Von Neumann stability plot of a Martin's $O\{3\}$ method showing limited stability near the origin.

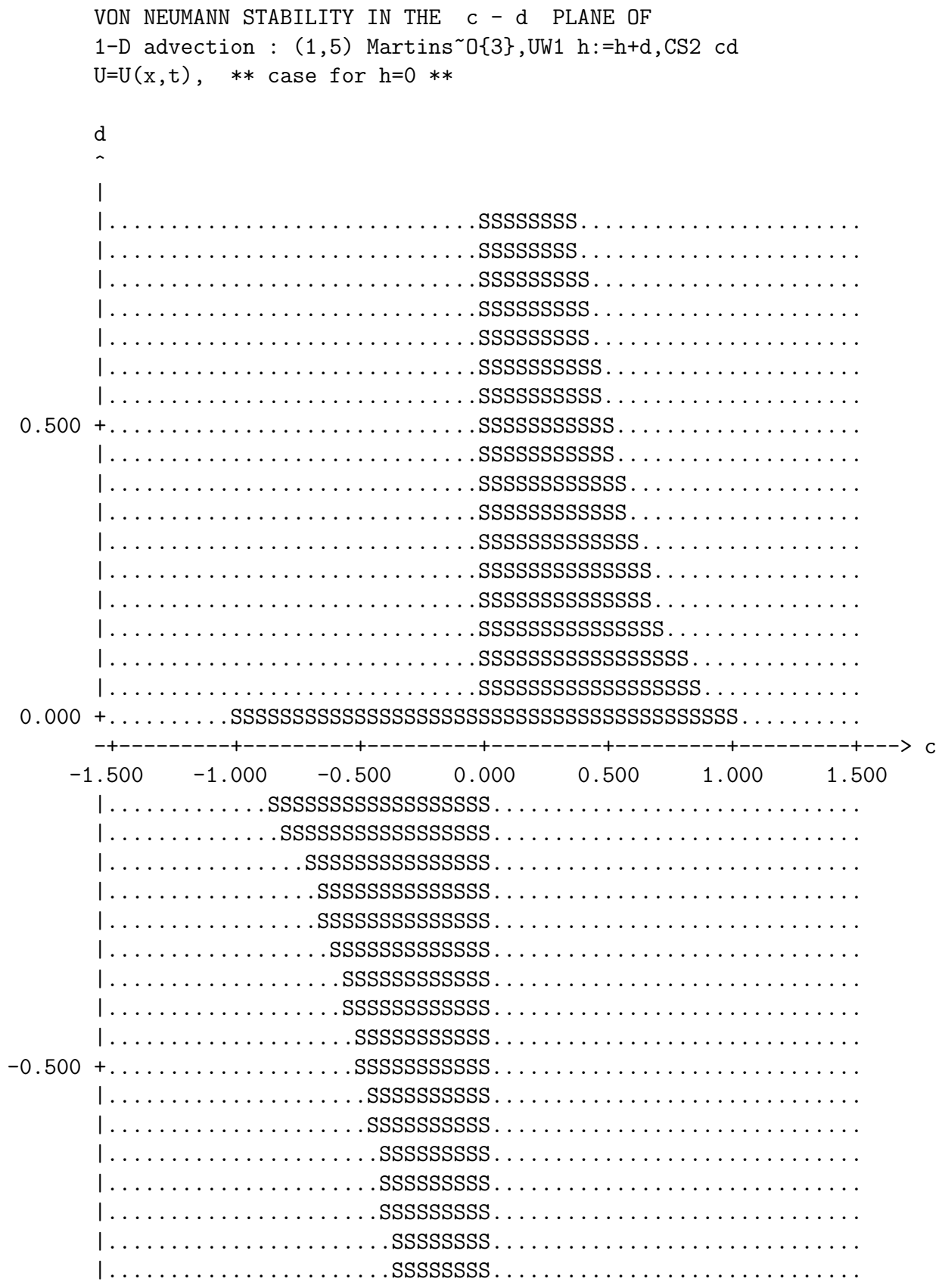

Figure A.2: Von Neumann stability plot of a Martin's $O\{\sim 3\}$ method showing grater stability near the origin.

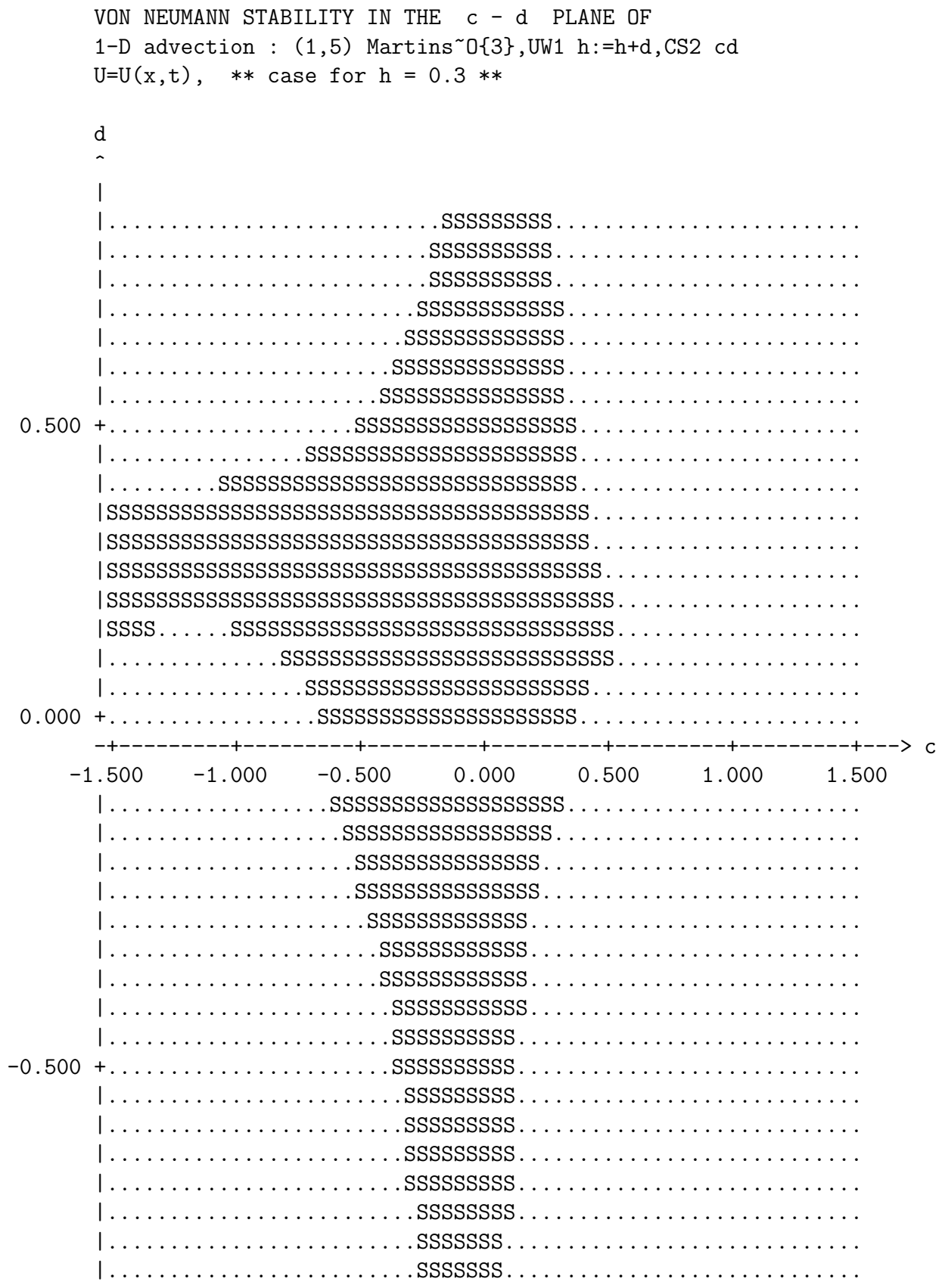

Figure A.3: Von Neumann stability plot of a Martin's $O\{\sim 3\}$ method showing stability when $h = 0.3$.

Appendix B

Program Listings

The following is coding for two of the programs used in the tests. The first program simulates the space-varying velocity field, and the second program simulates the time-varying velocity field. Slight modifications were required to these programs to test the varying methods.

```
*     NUMERICAL TEST FOR 1-D ADVECTION
*
*     ======  Space Varying Velocity  ==========
*
*     List of symbols:
*
*       taunew = new value of tau at time level n+1
*       tauold = old value of tau at time level n
*       xmax = maximum value on x-axis : 0 < x < xmax
*       bigj = number of space steps in x-axis
*       finalt = final time
*       u  = velocity of advection in x-direction = slope*(x + ay), ay is a
*            constant.
*       slope = slope of space-varying velocity
*       umax = maximum absolute velocity
*       peakfk = parameter of gaussian peak, smaller values give steeper
*                peaks
*       peakx0 = location of initial peak
*       deltax = space step in x-direction
*       deltat = time step delta-t
*       c = space-varying courant number = u*deltat/deltax
*       d = parameter used in correction for space-varying velocity
*         = -u*(du/dx)*(deltat**2)/(2*deltax)
*         = -(slope**2)*(x+ay)*(deltat**2)/(2*deltax)
*       h = parameter used in correction for space-varying velocity
*         = d + deltat^3/(6*deltax)*{u*(du/dx)^2 + u^2*(d2u/dx2)}
*         = d + deltat^3/(6*deltax)*{u*slope^2 + u^2*0}
*         = d + (deltat^3*u*slope^2)/(6*deltax)
```

```
*         = d + (deltat^3*(x+ay)*slope^3)/(6*deltax)
*         = -(slope^2*(x+ay)*(deltat^2)/(6*deltax))*(3-slope*deltat)
*       cmax = maximum absolute value of space-varying courant number
*            = umax*deltat/deltax
*       dmax = maximum absolute value of dj
*            = abs(slope*umax*(deltat**2)/(2*deltax))
*       hmax = maximum absolute value of hj
*            = abs(slope*umax*deltat^2*(3-slope*deltat)/(6*deltax))
*       bign = number of time steps
*       peakxf = final peak position, based upon initial position +
*                u=dx/dt=slope*(x+ay) * integrated from t=0 to t=finalt
*              = (peakx0+ay)*exp(slope*finalt)-ay
*       abserr = absolute error
*       exactv = exact solution using initial value
*              = exp(-peakfk*((x+ay)*exp(-slope*t) - ay - peakx0)**2)
*   ! !                                                              !

    1 format('Gauss Test for 1-D Advection Equation',
     &          ' using Space Varying Velocity')
    2 format('Leith (1,3) Method with Space-Varying 1-D Advection')
    3 format('-----------------------------------',
     &          '------------------------------')
    4 format('Velocity in x-direction : u=' ,f4.1, '*(x', spf4.1, ')')

   11 format('  umax : maximum velocity in x-direction         : ',f8.4)
   12 format('  bigj : number of space-steps in x-direction    : ',i3)
   13 format('  deltat : step length of time                   : ',f8.4)
   14 format('  deltax : step length on x-axis                 : ',f8.4)
   15 format('  finalt : final time                            : ',f8.4)
   16 format('  bign : number of time steps                    : ',i3)
   17 format('  peakx0 : initial peak position                 : ',f8.4)
   18 format('  peakxf : final peak position exact solution    : ',f8.4)

   21 format('  cmax : theoretical maximum of courant number   : ',f8.4)
   22 format('  dmax : theoretical maximum absolute value of d : ',f9.5)
   23 format('  hmax : theoretical maximum absolute value of h : ',f9.5)
   24 format('Digital information')
   25 format('  actual maximum value of c = ',spf10.5)
   26 format('  actual minimum value of c = ',spf10.5)
   27 format('  actual maximum value of d = ',spf10.5)
   28 format('  actual minimum value of d = ',spf10.5)
   29 format('  actual maximum value of h = ',spf10.5)
   30 format('  actual minimum value of h = ',spf10.5)

   41 format('Error for the Original method')
   42 format('Error for the Improved method')
```

```
   43 format('----------------------------')

   51 format('Error measures')
   52 format('  rms error     = ',f10.4)
   53 format('  average error = ',f10.4)
   54 format('  maximum error = ',f10.4)

   61 format('Digital information')
   62 format('  numerical minimum height = ',f10.4)
   63 format('  numerical maximum height = ',f10.4)
   64 format('  exact maximum height     = ',f10.4)
   65 format('  relative peak height     = ',f10.4)

   71 format('Interpolated information')
   72 format('  numerical maximum height          = ',f10.4)
   73 format('  numerical peak position           = ',f10.4)
   74 format('  peak shift (interpolated - exact) = ',f10.4)

   81 format('cpu time used = ',f10.3,'s')

   91 format(f5.3,1x,f30.27)
   92 format(f5.3)
*                                                                       !
      implicit undefined(a-z)
      integer          j, bigj, peakj, n, bign
      double precision x(0:200), t(0:5000),u(0:200), c(0:200), d(0:200)
      double precision h(0:200), taunew(0:200), tauold(0:200)
      double precision xmax, slope, ay, finalt, peakx0, peakfk, cmax
      double precision umax, deltax, deltat, dmax, hmax
      double precision xt, cj, dj, hj, cj2, a1, a2, a3, a4, a5
      double precision djmax, djmin, cjmax, cjmin, hjmax, hjmin
      double precision peakxf
      double precision exactv, error, sqsum, abserr, taumax, taumin
      double precision sumerr, rmserr, aveerr, errmax, least
      double precision f0, f1, f2, p, q, xp
      double precision maxval
      real             dtime, stats(2), tottm, cputm, systm

* Input Parameters:

      xmax = 3.0d0
      slope = -4.0d0
      ay = -6.0d0
      peakfk = 12.5d0
      peakx0 = 0.5d0
      finalt = 0.1d0
```

```
      write(6,*)'Please enter the number of space steps in x-axis :'
      read*,bigj
      write(6,*)'Please enter the maximum value of the courant number :'
      read*,cmax
*      write(6,*)'Please enter the maximum value of x (xmax):'
*      read*,xmax
*      write(6,*)'Please enter the slope of the velocity parameter ',
*     &          '(slope):'
*      read*,slope
*      write(6,*)'Please enter the x-intercept of the velocity ',
*     &          'parameter (ay):'
*      read*,ay
*      write(6,*)'Please enter the peak parameter (peakfk):'
*      read*,peakfk
*      write(6,*)'Please enter the initial position of the peak ',
*     &          '(peakx0):'
*      read*,peakx0
*      write(6,*)'and the final time (finalt):'
*      read*,finalt
      write(6,*)'Thankyou.'

* Calculation of Parameters:

      if (slope .gt. 0) then
        umax = slope*(xmax+ay)
      else
        umax = slope*ay
      endif

      deltax = xmax / bigj
      deltat = cmax * deltax / umax
      dmax = abs(umax * slope * deltat**2 / (deltax*2.0))
      hmax = abs(umax * slope * deltat**2 * (3-slope*deltat)/(6*deltax))
      bign = nint(finalt/deltat)

      peakxf = (peakx0+ay)*exp(slope*finalt)-ay

* Print the Title :

      write(1,1)
      write(1,2)
      write(1,3)
      write(1,4) slope, ay
      write(1,*)''
      write(1,11) umax
```

```
      write(1,12) bigj
      write(1,13) deltat
      write(1,14) deltax
      write(1,15) finalt
      write(1,16) bign
      write(1,17) peakx0
      write(1,18) peakxf

      write(20,2)
      write(20,4) slope, ay
      write(20,12) bigj
      write(20,21) cmax
      write(20,15) finalt
      write(20,17) peakx0

* Test to see if the number of time steps (bign) reaches the final time
* (finalt) exactly.

      if (abs((bign*deltat)-finalt) .ge. 1d-12) then
        print *,  'Test Failed : try using cmax =',
     &                           finalt*umax/(bign*deltax)
        stop
      endif

* Setting up the Arrays for t, x, u, c & d :

      do 100 n = 0,bign
        t(n) = n * deltat
  100 continue

      cjmax = -100.0
      cjmin = 100.0
      djmax = -100.0
      djmin = 100.0
      hjmax = -100.0
      hjmin = 100.0

      do 200 j = 0,bigj
        x(j) = j * deltax
        u(j) = slope*(x(j)+ay)
        c(j) = u(j)*deltat/deltax
        d(j) = -u(j)*slope*deltat**2/(2*deltax)
        h(j) = d(j) + (deltat**3*u(j)*slope**2)/(6*deltax)

        cjmax = max(c(j),cjmax)
        cjmin = min(c(j),cjmin)
```

```
        djmax = max(d(j),djmax)
        djmin = min(d(j),djmin)
        hjmax = max(h(j),hjmax)
        hjmin = min(h(j),hjmin)

  200 continue

      write(1,21)cmax
      write(1,22)dmax
c     write(1,23)hmax
      write(1,*)''
      write(1,24)
      write(1,25)cjmax
      write(1,26)cjmin
      write(1,27)djmax
      write(1,28)djmin
c     write(1,29)hjmax
c     write(1,30)hjmin

* ---------------------------------------------------------------------
* This calculates the results for the original method
* First set up the initial values tauold(j)

      do 300 j = 0,bigj
        tauold(j) = exp(-peakfk*(x(j)-peakx0)**2 )
  300 continue

* Start the Clock

      tottm = dtime(stats)
      cputm = stats(1)
      systm = stats(2)

* Calculation of the new values, taunew, from old values, tauold,
* at each time step :

      do 400 n = 1, bign

* Calculations of boundary values for x = 0, deltax, xmax-deltax, and xmax :

        xt = exp(-slope*t(n))
        taunew(0) = exp(-peakfk*((x(0)+ay)*xt-ay-peakx0)**2)
        taunew(1) = exp(-peakfk*((x(1)+ay)*xt-ay-peakx0)**2)
        taunew(bigj-1) = exp(-peakfk*((x(bigj-1)+ay)*xt-ay-peakx0)**2)
        taunew(bigj) = exp(-peakfk*((x(bigj)+ay)*xt-ay-peakx0)**2)
```

```
* Calculation of taunew(j,n+1) :

c Only one of these loops is to be used - depending on the stencil

        do 410 j =1,bigj-1
c       do 410 j =2,bigj-2
          cj = c(j)
          cj2 = cj*cj
          a1 = 0
          a2 = 0.5*(cj+cj2)
          a3 = (1-cj2)
          a4 = -0.5*(cj-cj2)
          a5 = 0
          taunew(j) = a1*tauold(j-2) + a2*tauold(j-1) + a3*tauold(j)
     &              + a4*tauold(j+1) + a5*tauold(j+2)
  410   continue

*  Replacing old values by new values :

        do 420 j = 0,bigj
          tauold(j)  = taunew(j)
  420   continue
  400 continue

* Stop the Clock

      tottm = dtime(stats)
      cputm = stats(1)
      systm = stats(2)

*-----------------------------------------------------------------------
* Calculation of error measures : rmserr, average abs value and max-error

      sqsum = 0.0
      sumerr = 0.0
      errmax = 0.0
      taumin = 10.0
      taumax = 0.0
      maxval = 0.0

      do 500 j = 0,bigj

*  exactv = exact values at t = finalt

        xt = (x(j)+ay)*exp(-slope*finalt)
        exactv = exp(-peakfk*(xt-peakx0-ay)**2)
```

```
        error = exactv - taunew(j)
        sqsum = sqsum + error * error
        abserr = abs(error)
        sumerr = sumerr + abserr
        if(taunew(j).ge.taumax) then
          taumax = taunew(j)
          peakj = j
        endif
        taumin = min(taumin,taunew(j))
        maxval = max(maxval,exactv)
        errmax = max(errmax,abserr)

* Writing out the numerical and exact data.

        write(3,91) x(j), taunew(j)
        write(4,91) x(j), exactv
  500 continue
        least = taumin

* Finding the maximum turning point of the numerical solution (xp,p)

      f0 = taunew(peakj-1)
      f1 = taunew(peakj)
      f2 = taunew(peakj+1)
      q = 8*f0*f1+8*f1*f2 +2*f0*f2-f0**2 -16*f1**2 -f2**2
      p = q/(8*(f0 - 2*f1 + f2))

      xp = x(peakj-1)+deltax*(3*f0 -4*f1 +f2)/(2*(f0 - 2*f1 + f2))

      rmserr = sqrt( sqsum/(bigj+1))
      aveerr = sumerr/(bigj+1)

      write(1,*)' '
      write(1,*)' '
      write(1,41)
      write(1,43)
      write(1,51)
      write(1,52)rmserr
      write(1,53)aveerr
      write(1,54)errmax
      write(1,*)''
      write(1,61)
      write(1,62)taumin
      write(1,63)taumax
      write(1,64)maxval
      write(1,65)taumax/maxval
```

```
      write(1,*)''
      write(1,71)
      write(1,72)p
      write(1,73)xp
      write(1,74)xp - peakxf
      write(1,*)''
      write(1,81)cputm

* ----------------------------------------------------------------------

* This calculates the results for the improved method
* first set up the initial values tauold(j)

      do 600 j = 0,bigj
        tauold(j) = exp(-peakfk*(x(j)-peakx0)**2 )
  600 continue

* Start the Clock

      tottm = dtime(stats)
      cputm = stats(1)
      systm = stats(2)

* Calculation of the new values, taunew, from old values, tauold,
* at each time step.

      do 700 n = 1, bign

* Calculations of boundary values for x = 0, deltax, xmax-deltax, and  xmax

        xt = exp(-slope*t(n))
        taunew(0) = exp(-peakfk*((x(0)+ay)*xt-ay-peakx0)**2)
        taunew(1) = exp(-peakfk*((x(1)+ay)*xt-ay-peakx0)**2)
        taunew(bigj-1) = exp(-peakfk*((x(bigj-1)+ay)*xt-ay-peakx0)**2)
        taunew(bigj) = exp(-peakfk*((x(bigj)+ay)*xt-ay-peakx0)**2)

* Calculation of taunew(j,n+1)

c Only one of these loops is to be used - depending on the stencil

        do 710 j =1,bigj-1
c       do 710 j =2,bigj-2
          cj = c(j)
          cj2 = cj*cj
          dj = d(j)
          hj = h(j)
```

```
          a1 = 0
          a2 = 0.5*(cj+cj2+dj+abs(dj))
          a3 = (1-cj2-abs(dj))
          a4 = -0.5*(cj-cj2+dj-abs(dj))
          a5 = 0
          taunew(j) = a1*tauold(j-2) + a2*tauold(j-1) + a3*tauold(j)
     &              + a4*tauold(j+1) + a5*tauold(j+2)
  710   continue

* Replacing old values by new values :

        do 720 j = 0,bigj
          tauold(j)  = taunew(j)
  720   continue
  700 continue

*-----------------------------------------------------------------------

* Stop the Clock

      tottm = dtime(stats)
      cputm = stats(1)
      systm = stats(2)

* Calculation of error measures ; rmserr, ave abs error and max-error :

      sqsum = 0.0
      sumerr = 0.0
      errmax = 0.0
      taumin = 10.0
      taumax = 0.0
      maxval = 0.0

      do 800 j = 0,bigj

*  exactv = exact values at t = finalt

        xt = (x(j)+ay)*exp(-slope*finalt)
        exactv = exp(-peakfk*(xt-peakx0-ay)**2)
        error = exactv - taunew(j)
        sqsum = sqsum + error * error
        abserr = abs(error)
        sumerr   = sumerr + abserr
        if(taunew(j).ge.taumax) then
          taumax = taunew(j)
          peakj = j
```

```
        endif
        taumin = min(taumin,taunew(j))
        maxval = max(maxval,exactv)
        errmax = max(errmax,abserr)

* Writing the improved values to output unit 2.

        write(2,91)x(j),taunew(j)

  800 continue

      least = min(least, taumin)
      write(21,92) least

* Finding the maximum turning point of the numerical solution (xp,p)

      f0 = taunew(peakj-1)
      f1 = taunew(peakj)
      f2 = taunew(peakj+1)
      q = 8*f0*f1+8*f1*f2 +2*f0*f2-f0**2 -16*f1**2 -f2**2
      p = q/(8*(f0 - 2*f1 + f2))

      xp = x(peakj-1)+deltax*(3*f0 -4*f1 +f2)/(2*(f0 - 2*f1 + f2))

      rmserr = sqrt( sqsum/(bigj+1))
      aveerr = sumerr/(bigj+1)

      write(1,*)' '
      write(1,42)
      write(1,43)
      write(1,51)
      write(1,52)rmserr
      write(1,53)aveerr
      write(1,54)errmax
      write(1,*)''
      write(1,61)
      write(1,62)taumin
      write(1,63)taumax
      write(1,64)maxval
      write(1,65)taumax/maxval
      write(1,*)''
      write(1,71)
      write(1,72)p
      write(1,73)xp
      write(1,74)xp - peakxf
      write(1,*)''
```

```
write(1,81)cputm

end
```

```
*     NUMERICAL TEST FOR 1-D ADVECTION
*
*     ======  Time Varying Velocity  ==========
*
*     List of symbols:
*
*       taunew = new value of tau at time level n+1
*       tauold = old value of tau at time level n
*       xmax = maximum value on x-axis : 0 < x < xmax
*       bigj = number of space steps in x-axis
*       finalt = final time
*       u  = velocity of advection in x-direction
*          = umax * sin(omega*t + theta)
*       omega = angular frequency of the time varying velocity
*       theta = phase of velocity at time t=0
*       umax = maximum absolute velocity
*       peakfk = parameter of gaussian peak, smaller values give steeper
*                peaks
*       peakx0 = location of initial peak
*       deltax = space step in x-direction
*       deltat = time step delta-t
*       c = time-varying courant number = u*deltat/deltax
*       d = parameter used in correction for space-varying velocity
*         = deltat**2/(2*deltax)*(du/dt+u*du/dx)
*         = deltat**2/(2*deltax)*(umax*omega*cos(omega*t+theta))
*       h = parameter used in correction for space-varying velocity
*       cmax = maximum absolute value of space-varying courant number
*            = umax*deltat/deltax
*       dmax = maximum absolute value of dn
*       hmax = maximum absolute value of hn or an upperbound
*       bign = number of time steps
*       peakxt = peak position at time t, based upon initial peak position +
*                u = dx/dt = umax * sin(omega*t + theta)
*                integrated from t=0 to t
*              = -umax/omega * (cos(omega*t + theta) - cos(theta)) +peakx0
*       peakxf = final peak position
*              = -umax/omega * (cos(omega*finalt + theta) - cos(theta))
*                +peakx0
*       abserr = absolute error
*       exactv = exact solution using initial value
*              =
*   ! !                                                                !

    1 format('Gauss Test for 1-D Advection Equation',
     &          ' using Time Varying Velocity')
    2 format('Martin O{2} and O{~3} (1,5) Methods with Time-Varying',
```

```
     &       ' 1-D Advection')
    3 format('-------------------------------------',
     &          '----------------------------')
    4 format('Velocity in x-direction : u=',f4.1,'*sin(',f4.1,'*t',
     &        spf4.1,')')
    5 format('Velocity in x-direction : u=(pi/',f4.1,')*sin((pi/',f4.1,
     & ')*t+(pi/',f4.1,'))')

   11 format('  umax : maximum velocity in x-direction         : ',f9.4)
   12 format('  bigj : number of space-steps in x-direction    : ',i4)
   13 format('  deltat : step length of time                   : ',f9.4)
   14 format('  deltax : step length on x-axis                 : ',f9.4)
   15 format('  finalt : final time                            : ',f9.4)
   16 format('  bign : number of time steps                    : ',i4)
   17 format('  peakx0 : initial peak position                 : ',f9.4)
   18 format('  peakxf : final peak position exact solution    : ',f9.4)

   20 format('  cmax : theoretical maximum of courant number   : pi/',
     & f9.4)
   21 format('  cmax : theoretical maximum of courant number   : ',f9.4)
   22 format('  dmax : theoretical maximum absolute value of d : ',
     & f10.5)
   23 format('  hmax : upperbound for the absolute value of h  : ',
     & f10.5)
   24 format('Digital information')
   25 format('  actual maximum value of c = ',spf10.5)
   26 format('  actual minimum value of c = ',spf10.5)
   27 format('  actual maximum value of d = ',spf10.5)
   28 format('  actual minimum value of d = ',spf10.5)
   29 format('  actual maximum value of h = ',spf10.5)
   30 format('  actual minimum value of h = ',spf10.5)

   41 format('Error for the Original method')
   42 format('Error for the Improved method')
   43 format('----------------------------')

   51 format('Error measures')
   52 format('  rms error     = ',f10.4)
   53 format('  average error = ',f10.4)
   54 format('  maximum error = ',f10.4)

   61 format('Digital information')
   62 format('  numerical minimum height = ',f10.4)
   63 format('  numerical maximum height = ',f10.4)
   64 format('  exact maximum height     = ',f10.4)
   65 format('  relative peak height     = ',f10.4)
```

```
   71 format('Interpolated information')
   72 format('  numerical maximum height          = ',f10.4)
   73 format('  numerical peak position           = ',f10.4)
   74 format('  peak shift (interpolated - exact) = ',f10.4)

   81 format('cpu time used = ',f10.3,'s')

   91 format(f5.3,1x,f30.27)
   92 format(f5.3)

  101 format('theoretical maximum of courant number : ',f4.1)
  102 format('theoretical maximum of courant number : pi/', f4.1)
  103 format('final time : ',f4.1)
  104 format('number of space-steps in x-direction : ',i3)
  105 format('initial peak position : ',f3.1)
*                                                                       !
      implicit undefined(a-z)
      integer          j, bigj, peakj, n, bign
      double precision x(0:200), t(0:5000), u(0:5000), c(0:5000)
      double precision d(0:5000), h(0:5000), taunew(0:200),tauold(0:200)
      double precision pi, omega, theta, div
      double precision xmax, finalt, peakx0, peakfk, peakxt, cmax
      double precision umax, deltax, deltat, dmax, hmax
      double precision cn, dn, hn, cn2, a1, a2, a3, a4, a5
      double precision dnmax, dnmin, cnmax, cnmin, hnmax, hnmin
      double precision peakxf
      double precision exactv, error, sqsum, abserr, taumax, taumin
      double precision sumerr, rmserr, aveerr, errmax, least
      double precision f0, f1, f2, p, q, xp
      double precision maxval
      real             dtime, stats(2), tottm, cputm, systm

* Input Parameters:

      pi = 3.141592653589793238 46d0
      omega = pi/2.0
      umax = pi/2.0
      peakfk = 144.0
      theta = pi/2.0
      xmax = 3.0d0
      peakx0 = 1.5d0

      write(6,*)'The final time should be of the form 4n or (4n+1),',
     &          'where n = 0,1,2,...  (eg. 4.0D0)'
      write(6,*)'Please enter the final time  :'
```

```
      read*,finalt
      write(6,*)'and the number of space steps in x-axis :'
      read*,bigj
      write(6,*)'The value of cmax is given by cmax = pi/(2*div).'
      write(6,*) 'The parameter div should be of the form k*',
     & xmax/(bigj*finalt)
      write(6,*) 'where k is an integer.'
      write(6,*)'Enter the value for div :'
      read*,div

*      write(6,*)'Please enter the peak parameter (peakfk):'
*      read*,peakfk
*      write(6,*)'Please enter the initial position of the peak ',
*     &          '(peakx0):'
*      read*,peakx0
      write(6,*)'Thankyou.'

* Calculation of Parameters:

      cmax = umax/div
      deltax = xmax / bigj
      deltat = cmax * deltax / umax
      bign = nint(finalt/deltat)
      dmax = umax * omega * deltat**2 /(2*deltax)
      hmax = deltat**3/(6*deltax)*umax*omega**2 + dmax

      peakxf = -umax/omega * (cos(omega*finalt + theta) - cos(theta))
     &         + peakx0

* Print the Title :

      write(1,1)
      write(1,2)
      write(1,3)
      if (int(pi/omega*100)-(pi/omega*100) .ge. 1d-12) then
        write(1,4) umax, omega, theta
      else
        write(1,5) pi/umax, pi/omega, pi/theta
      endif
      write(1,*)''
      write(1,11) umax
      write(1,12) bigj
      write(1,13) deltat
      write(1,14) deltax
      write(1,15) finalt
      write(1,16) bign
```

```
      write(1,17) peakx0
      write(1,18) peakxf

      write(20,2)
      if (int(pi/omega*100)-(pi/omega*100) .ge. 1d-12) then
        write(20,4) umax, omega, theta
      else
        write(20,5) pi/umax, pi/omega, pi/theta
      endif
      if (int(pi/cmax*100)-(pi/cmax*100) .ge. 1d-12) then
        write(20,101) cmax
      else
        write(20,102) pi/cmax
      endif
      write(20,103) finalt
      write(20,104) bigj
      write(20,105) peakx0

* Test to see if the number of time steps (bign) reaches the final time
* (finalt) exactly.

      if (abs((bign*deltat)-finalt) .ge. 1d-12) then
        print *,  'Test Failed'
        stop
      endif

* Setting up the Arrays for t, x, u, c & d :

      cnmax = -100.0
      cnmin = 100.0
      dnmax = -100.0
      dnmin = 100.0
      hnmax = -100.0
      hnmin = 100.0

      do 100 n = 0,bign

        t(n) = n * deltat
        u(n) = umax*sin(omega*t(n)+theta)
        c(n) = u(n)*deltat/deltax
        d(n) = deltat**2/(2*deltax)*umax*omega*cos(omega*t(n)+theta)
        h(n) = -deltat**3/(6*deltax)*umax*omega**2*sin(omega*t(n)+theta)
     &         +d(n)

        cnmax = max(c(n),cnmax)
        cnmin = min(c(n),cnmin)
```

```
        dnmax = max(d(n),dnmax)
        dnmin = min(d(n),dnmin)
        hnmax = max(h(n),hnmax)
        hnmin = min(h(n),hnmin)

  100 continue

      do 200 j = 0,bigj
        x(j) = j * deltax
  200 continue

      write(1,21)cmax
      write(1,22)dmax
      write(1,23)hmax
      write(1,*)''
      write(1,24)
      write(1,25)cnmax
      write(1,26)cnmin
      write(1,27)dnmax
      write(1,28)dnmin
      write(1,29)hnmax
      write(1,30)hnmin

* ---------------------------------------------------------------------
* This calculates the results for the original method
* First set up the initial values tauold(j)

      do 300 j = 0,bigj
        tauold(j) = exp(-peakfk*(x(j)-peakx0)**2 )
  300 continue

* Start the Clock

      tottm = dtime(stats)
      cputm = stats(1)
      systm = stats(2)

* Calculation of the new values, taunew, at the n+1 time level from
* old values, tauold, at each time step :

      do 400 n = 0, bign-1

* Calculations of boundary values for x = 0, deltax, xmax-deltax, and xmax :

        peakxt = -umax/omega * (cos(omega*t(n+1) + theta) - cos(theta))
     &           + peakx0
```

```
        taunew(0) = exp(-peakfk*(peakxt-x(0))**2)
        taunew(1) = exp(-peakfk*(peakxt-x(1))**2)
        taunew(bigj-1) = exp(-peakfk*(peakxt-x(bigj-1))**2)
        taunew(bigj) = exp(-peakfk*(peakxt-x(bigj))**2)

* Calculation of taunew(j,n+1) :

c Only one of these loops is to be used - depending on the stencil

c       do 410 j =1,bigj-1
        do 410 j =2,bigj-2
          cn = c(n)
          cn2 = cn*cn
          a1 = -((abs(cn)+cn)*(1-cn)*(1+cn))/12
          a2 = ((1+cn)*(2-abs(cn))*(abs(cn)+2*cn)+3*(abs(dn)+dn))/6
          a3 = ((1+cn)*(1-cn)*(2-abs(cn))-2*abs(dn))/2
          a4 = ((1-cn)*(2-abs(cn))*(abs(cn)-2*cn)+3*(abs(dn)-dn))/6
          a5 = -((abs(cn)-cn)*(1-cn)*(1+cn))/12
          taunew(j) = a1*tauold(j-2) + a2*tauold(j-1) + a3*tauold(j)
     &              + a4*tauold(j+1) + a5*tauold(j+2)
  410   continue

*  Replacing old values by new values :

        do 420 j = 0,bigj
          tauold(j)  = taunew(j)
  420   continue
  400 continue

* Stop the Clock

      tottm = dtime(stats)
      cputm = stats(1)
      systm = stats(2)

*-----------------------------------------------------------------------
* Calculation of error measures : rmserr, average abs value and max-error

      sqsum = 0.0
      sumerr = 0.0
      errmax = 0.0
      taumin = 10.0
      taumax = 0.0
      maxval = 0.0

      do 500 j = 0,bigj
```

```
*  exactv = exact values at t = finalt

        exactv = exp(-peakfk*(x(j)-peakxf)**2)
        error = exactv - taunew(j)
        sqsum = sqsum + error * error
        abserr = abs(error)
        sumerr = sumerr + abserr
        if(taunew(j).ge.taumax) then
          taumax = taunew(j)
          peakj = j
        endif
        taumin = min(taumin,taunew(j))
        maxval = max(maxval,exactv)
        errmax = max(errmax,abserr)

* Writing out the numerical and exact data.

        write(3,91) x(j), taunew(j)
        write(4,91) x(j), exactv
  500 continue
        least = taumin

* Finding the maximum turning point of the numerical solution (xp,p)

      f0 = taunew(peakj-1)
      f1 = taunew(peakj)
      f2 = taunew(peakj+1)
      q = 8*f0*f1+8*f1*f2 +2*f0*f2-f0**2 -16*f1**2 -f2**2
      p = q/(8*(f0 - 2*f1 + f2))

      xp = x(peakj-1)+deltax*(3*f0 -4*f1 +f2)/(2*(f0 - 2*f1 + f2))

      rmserr = sqrt( sqsum/(bigj+1))
      aveerr = sumerr/(bigj+1)

      write(1,*)' '
      write(1,*)' '
      write(1,41)
      write(1,43)
      write(1,51)
      write(1,52)rmserr
      write(1,53)aveerr
      write(1,54)errmax
      write(1,*)''
      write(1,61)
```

```
      write(1,62)taumin
      write(1,63)taumax
      write(1,64)maxval
      write(1,65)taumax/maxval
      write(1,*)''
      write(1,71)
      write(1,72)p
      write(1,73)xp
      write(1,74)xp - peakxf
      write(1,*)''
      write(1,81)cputm

* ----------------------------------------------------------------------

* This calculates the results for the improved method
* first set up the initial values tauold(j)

      do 600 j = 0,bigj
        tauold(j) = exp(-peakfk*(x(j)-peakx0)**2 )
  600 continue

* Start the Clock

      tottm = dtime(stats)
      cputm = stats(1)
      systm = stats(2)

* Calculation of the new values, taunew, at the n+1 time level from
* old values, tauold, at each time step :

      do 700 n = 0, bign-1

* Calculations of boundary values for x = 0, deltax, xmax-deltax, and xmax :

        peakxt = -umax/omega * (cos(omega*t(n+1) + theta) - cos(theta))
     &           + peakx0
        taunew(0) = exp(-peakfk*(peakxt-x(0))**2)
        taunew(1) = exp(-peakfk*(peakxt-x(1))**2)
        taunew(bigj-1) = exp(-peakfk*(peakxt-x(bigj-1))**2)
        taunew(bigj) = exp(-peakfk*(peakxt-x(bigj))**2)

* Calculation of taunew(j,n+1)

c Only one of these loops is to be used - depending on the stencil

c       do 710 j =1,bigj-1
```

```
        do 710 j =2,bigj-2
          cn = c(n)
          cn2 = cn*cn
          dn = d(n)
          hn = h(n)
          a1 = -((abs(cn)+cn)*(1-cn)*(1+cn))/12
          a2 = ((1+cn)*(2-abs(cn))*(abs(cn)+2*cn)
     &         +6*(cn*dn)+3*(abs(hn)+hn))/6
          a3 = ((1+cn)*(1-cn)*(2-abs(cn))-4*cn*dn-2*abs(hn))/2
          a4 = ((1-cn)*(2-abs(cn))*(abs(cn)-2*cn)
     &         +6*(cn*dn)+3*(abs(hn)-hn))/6
          a5 = -((abs(cn)-cn)*(1-cn)*(1+cn))/12
          taunew(j) = a1*tauold(j-2) + a2*tauold(j-1) + a3*tauold(j)
     &              + a4*tauold(j+1) + a5*tauold(j+2)
  710   continue

* Replacing old values by new values :

        do 720 j = 0,bigj
          tauold(j)  = taunew(j)
  720   continue
  700 continue

* Stop the Clock

      tottm = dtime(stats)
      cputm = stats(1)
      systm = stats(2)

*-----------------------------------------------------------------------
* Calculation of error measures ; rmserr, ave abs error and max-error :

      sqsum = 0.0
      sumerr = 0.0
      errmax = 0.0
      taumin = 10.0
      taumax = 0.0
      maxval = 0.0

      do 800 j = 0,bigj

*  exactv = exact values at t = finalt

        exactv = exp(-peakfk*(x(j)-peakxf)**2)
        error = exactv - taunew(j)
        sqsum = sqsum + error * error
```

```
        abserr = abs(error)
        sumerr   = sumerr + abserr
        if(taunew(j).ge.taumax) then
          taumax = taunew(j)
          peakj = j
        endif
        taumin = min(taumin,taunew(j))
        maxval = max(maxval,exactv)
        errmax = max(errmax,abserr)

* Writing the improved values to output unit 2.

        write(2,91)x(j),taunew(j)

  800 continue

      least = min(least, taumin)
      write(21,92) least

* Finding the maximum turning point of the numerical solution (xp,p)

      f0 = taunew(peakj-1)
      f1 = taunew(peakj)
      f2 = taunew(peakj+1)
      q = 8*f0*f1+8*f1*f2 +2*f0*f2-f0**2 -16*f1**2 -f2**2
      p = q/(8*(f0 - 2*f1 + f2))

      xp = x(peakj-1)+deltax*(3*f0 -4*f1 +f2)/(2*(f0 - 2*f1 + f2))

      rmserr = sqrt( sqsum/(bigj+1))
      aveerr = sumerr/(bigj+1)

      write(1,*)' '
      write(1,42)
      write(1,43)
      write(1,51)
      write(1,52)rmserr
      write(1,53)aveerr
      write(1,54)errmax
      write(1,*)''
      write(1,61)
      write(1,62)taumin
      write(1,63)taumax
      write(1,64)maxval
      write(1,65)taumax/maxval
      write(1,*)''
```

```
write(1,71)
write(1,72)p
write(1,73)xp
write(1,74)xp - peakxf
write(1,*)''
write(1,81)cputm

end
```

Appendix C

Bibliography

Abelink, R. and Wheater, H. S., (1990) *Parameter identification of solute transport models for unsaturated soils.* J. Hydrol., 117: 199-224.

Batu, V., (1989) *A generalized two-dimensional analytical solution for hydrodynamic dispersion in bounded media with first-type boundary condition at the source.* Water Resourses Research, Vol. 25, No. 6. 1125-1132.

Bencala, K. E., McKnight, D. M. and Zellweger, G. W., (1990) *Characterization of transport in an acidic and metal-rich mountain stream based on a lithium traces injection and simulations of transient storage.* Water Resourses Research, Vol. 26, No. 5. 989-1000.

Bensabat, J. and Zeitoun, D. G., (1990) *A least-squares procedure for the solution of transport problems.* International Journal for Numerical Methods in Fluids, Vol. 10. 623-636.

Calder, I. R., (1991) *Implications and assumptions in using the 'total counts' and convection-dispersion equations for tracer flow measurements — with particular reference to transpiration measurements in trees.* J. Hydrol., 125: 144-158.

Celia, M. A., Kindred, J. S. and Herrera, I., (1989) *Contaminant transport and biodegradation — A numerical model for reactive transport in porous media.* Water Resources Research, Vol. 25, No. 6. 1141-1148.

Cheong, H. F., Abdul Khader, M. H., Yang, C. J. and Radhakrishnan, R., (1992) *The dispersion of radioactive tracers along the east coast of Singapore.* Coastal Eng., 17: 71-92.

Chiang, C. Y., Wheeler, M. F. and Bedient, P. B., (1989) *A modified method of characteristics technique and mixed finite elements method for simulation of groundwater solute transport.* Water Resources Research, Vol. 25, No. 7. 1541-1549.

Contractor, D. N. and El-Didy, S. M. A., (1989) *Field application of a finite-element water-quality model to a coal seam with UCG burns.* J. Hydrol., 109: 57-64.

Courant, R., Issacson, E. and Rees, M. (1952) *On the solution of hyperbolic differential equations by finite differences* Communications on Pure and Applied Mathematics, Vol. 5. 243-255.

Cox, R. A. and Nishikawa, T., (1991) *A new total variation diminishing scheme for the solution of advective-dominant solute transport.* Water Resources Research, Vol. 27, No. 10. 2645-2654.

Culver, T. B., Shoemaker, C. A. and Lion, L. W., (1991) *Impact of vapor sorption on the subsurface transport of volatile organic compounds: A numerical model and analysis.* Water Resources Research, Vol. 27, No. 9. 2259-2270.

Dillon, P. J., (1989) *An analytical model of contaminant transport from diffuse sources in saturated porous media.* Water Resources Research, Vol. 25, No. 6. 1208-1218.
D'Yankonov, Y.G. (1963) *Difference schemes with split operators for multidimensional unsteady problems.* U.S.S.R. Computational Mathematics, Vol. 4, No. 2. 92-110
Fio, J. L. and Deverel. S. J., (1991) *Groundwater flow and solute movement to drain laterals, Western San Joaquin Valley, California — quantitative hydrologic assessment.* Water Resources Research, Vol. 27, No. 9. 2247-2257.
Ford, M., Wang, J. and Cheng, R. T., (1990) *Predicting the vertical structure of tidal current and salinity in San Francisco Bay, California.* Water Resources Research, Vol. 26, No. 5. 1027-1045.
Frind, E. O., Duynisveld, W. H. M., Strebel, O. and Boettcher, J., (1990) *Modeling of multicomponent transport with microbial transformation in groundwater: the Fuhrberg case.* Water Resources Research, Vol. 26, No. 8. 1707-1719.
Gane, C. R. and Stephenson, P. L., (1979) *An explicit numerical method for solving transient combined heat conduction and convection problems.* Int. j. numer. methods eng., 14: 1141-1163.
Hartnett, M. and Cawley, A. M., (1991) *Mathematical modelling of the effects of marine aquaculture developments on certain water quality parameters.*
Hassanizadeh, S. M. and Leijnse, T., (1988) *On the modeling of brine transport in porous media.* Water Resources Research, Vol. 24, No. 3. 321-330.
Herbert, A. W., Jackson, C. P. and Lever, D. A., (1988) *Coupled groundwater flow and solute transport with fluid density strongly dependent upon concentration.* Water Resources Research, Vol. 24, No. 10. 1781-1795.
Huyakorn, P.S. and Pinder, G.F., (1983) *Computational Methods in Subsurface flow.* Academic.
Illangasekare, T. H. and Döll, P., (1989) *A discrete kernel method of characteristics model of solute transport in water table aquifers.* Water Resources Research, Vol. 25, No. 5. 857-867.
Jinzhong, Y., (1988) *Experimental and numerical studies of solute transport in two dimensional saturated-unsaturated soil.* Journal of Hydrology, Vol. 97. 303-322.
Kadelec, R. H., Li, X. and Cotten, G. B., (1988) *Modeling solute segregation during freezing of peatland waters.* Water Resources Research, Vol. 24, No. 2. 219-224.
Kaluarachchi, J. J. and Parker, J. C., (1988) *Finite element model of nitrogen species transformation and transport in the unsaturated zone.* Journal of Hydrology, Vol. 103. 249-274.
Kindred, J. S. and Celia, M. A., (1989) *Contaminant transport and biodegradation — conceptual model and test simulations.* Water Resources Research, Vol. 25, No. 6. 1149-1159.
Kinouchi, T., Kanda, M. and Hino, Mikio., (1991) *Numerical simulation of infiltration and solute transport in an s-shaped model basin by a boundary-fitted grid system.* Journal of Hydrology, Vol. 122. 373-406.
Knopman, D. S. and Voss, C. I., (1988) *Further comments on sensitivities, parameter estimation, and sampling design in one-dimensional analysis of solute transport in porous media.* Water Resources Research, Vol. 24, No. 2. 225-238.
Leij, F. J. and Dane, J. H., (1991) *The effect of transverse dispersion on solute transport in soils.* Journal of Hydrology, Vol. 122. 407-422.
Leonard, B.P. and Noye, B.J. (1990) *Second and third order two level implicit FDMs for unsteady one-dimensional convection-diffusion.* Computational Techniques and Applications: CTAC-89, 341-348.

Lindstrom, F. T. and Boersma, L., (1989) *Two-dimensional transport and fate of chemicals emitted by arbitrarily places sources in confined aquifers.* Water Resources Research, Vol. 25, No. 7. 1748-1756.
Lowell, R. P., (1989) *Contaminant transport in a single fracture: periodic boundary and flow conditions.* Water Resources Research, Vol. 25, No. 5. 774-780.
Lunel, T., (1991) *Eurospill: chemical spill model based on modelling turbulent mixing at sea.*
Macarthur, J. W. and Patankar, S. V., (1989) *Robust semidirect finite difference methods for solving the Navier-Stokes and energy equations.* International Journal for Numerical Methods in Fluids, Vol. 9. 325-340.
Noye, B.J., (1992) *Lecture notes for Finite Difference Methods for Partial Differential Equations*
Noye, B.J. and Tan, H.H. (1988) *A third-order semi-implicit finite difference method for solving the one-dimensional convection-diffusion equation* International Journal for Numerical Methods in Engineering, Vol.26 1615-1629.
Patyn, J., Ledoux, E. and Bonne, A., (1989) *Geohydrological research in relation to radioactive waste disposal in an argillaceous formation.* J. Hydrol., 109: 267-285.
Possingham, H. P. and Roughgarden, J., (1990) *Spatial population dynamics of a marine organism with a complex life cycle.* Ecology., 71(3): 973-985.
Putte, M., Yeh, W. W. G. and Mulder, W. A., (1990) *A triangular finite volume approach with high-resolution upwind terms for the solution of groundwater transport equations.* Water Resources Research, Vol. 26, No. 12. 2865-2880.
Roldão, J. S. F., Soares, J. H. P., Wrobel, L. C., Büge, T. R and Dias, N. L. C., (1991) *Pollutant transport studies in the Paraíba Do Sul river, Brazil.*
Rusanov, V. V., *On difference schemes of third order accuracy for non-linear hyperbolic systems.* Journal of Computational Physics, Vol. 5. 507-516.
Savenjie, H. H. G., (1989) *Salt intrusion model for highwater slack, low-water slack, and mean tide on spreadsheet.* J. Hydrol., 107: 9-18.
Solomon, D. K. and Sudicky, E. A., (1991) *Tritium and helium 3 isotope ratios for direct estimation of spatial variations in groundwater recharge.* Water Resources Research, Vol. 27, No. 9. 2309-2319.
Waluyo, D. (1991), *Numerical Test of FDM's for One and Two-Dimensional Advection and Advection Diffusion Equations*
Yates, S. R., (1988) *Three-dimensional radial dispersion in a variable velocity flow field.* Water Resources Research, Vol. 24, No. 7. 1083-1090.
Yates, S. R. and Enfield, C. G., (1989) *Transport of dissolved substances with second-order reaction.* Water Resources Research, Vol. 25, No. 7. 1757-1762.