EXPLICIT ALTERNATING DIRECTION METHODS FOR PROBLEMS IN FLUID DYNAMICS

BY

AZZAM AHMAD AL-WALI

A Doctral Thesis

Submitted in partial fulfilment of the requirements

for the award of Doctor of Philosophy

of the Loughborough University of Technology

August, 1994

SUPERVISOR: PROFESSOR D. J. EVANS, PhD, D.Sc.,
Department of Computer Studies

In the name of God, Most Gracious, Most Merciful.

Cowper wrote: *"Oars alone can ne'er prevail*
To reach the distant Coast;
The breath of heaven must swell the sail,
Or all the toil is lost."
Indeed!.

ACKNOWLEDGEMENTS

All praise be to God the Almighty, Who has given me the desire, will and perseverance to complete the work of this thesis.

I am greatly indebted to my supervisor Professor D. J. Evans who to me has a "breath of heaven" swelling my sail. His friendly unfailing guidance, continuous help, incessant encouragement, and his informative remarks has helped me steer my way during this period of endurance.

I thank him for allowing me the opportunity of working under his supervision and for his patience throughout. His conscientious character, his concern for those who work with him and under his supervision, his hard work, his versatile knowledge and experience, has made me share with other students and visiting researchers a deep admiration of him. I hope that I shall benefit from his example in learning and in the passing on of knowledge to my students and colleagues in the future.

I also wish to thank Dr. C. li for many fruitful discussions, Dr. A. Benson for his comments, Dr. W. Yousif and Mr. G. Samra for their technical assistance, and Mrs J. Poulton for her friendly and professional management of the affairs of staff and students of the Parallel Algorithms Research Centre.

Abstract

Recently an iterative method was formulated employing a new splitting strategy for the solution of tridiagonal systems of difference equations. The method was successful in solving the systems of equations arising from one dimensional initial boundary value problems, and a theoretical analysis for proving the convergence of the method for systems whose constituent matrices are positive definite was presented by Evans and Sahimi [22]. The method was known as the Alternating Group Explicit (AGE) method and is referred to as AGE-1D. The explicit nature of the method meant that its implementation on parallel machines can be very promising.

The method was also extended to solve systems arising from two and three dimensional initial-boundary value problems, but the AGE-2D and AGE-3D algorithms proved to be too demanding in computational cost which largely reduces the advantages of its parallel nature.

In this thesis, further theoretical analyses and experimental studies are pursued to establish the convergence and suitability of the AGE-1D method to a wider class of systems arising from univariate and multivariate differential equations with symmetric and non symmetric difference operators. Also the possibility of a Chebyshev acceleration of the AGE-1D algorithm is considered.

For two and three dimensional problems it is proposed to couple the use of the AGE-1D algorithm with an ADI scheme or an ADI iterative method in what is called the Explicit Alternating Direction (EAD) method. It is then shown through experimental results that the EAD method retains the parallel features of the AGE method and moreover leads to savings of up to 83 % in the computational cost for solving some of the model problems.

The thesis also includes applications of the AGE-1D algorithm and the EAD method to solve some problems of fluid dynamics such as the linearized Shallow Water equations, and the Navier Stokes' equations for the flow in an idealized one dimensional Planetary Boundary Layer.

The thesis terminates with conclusions and suggestions for further work together with a comprehensive bibliography and an appendix containing some selected programs.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 The relation between the problems of fluid dynamics and partial differential equations

The problems of fluid dynamics revolve around determining as a function of time and/or of space one or more of the dependent variables which characterises a fluid (i.e its velocity, and/or some of its thermodynamic variables, e.g pressure, temperature, density, specific internal energy ... etc.).

In most cases such problems may be mathematically described by a single first or second order *partial differential equation* (henceforth abbreviated as PDE) having the general form:

$$L(\tilde{u}) = \mathbf{r} \tag{1.1}$$

or a system of such equations.

In (1.1) $L(\tilde{u})$ is a differential operator which has the general form:

$$L(\tilde{u}) = \sum_{i=1}^{N}\sum_{j=1}^{N} a_{ij}\frac{\partial^2 \tilde{u}}{\partial x_i \partial x_j} + \sum_{k=1}^{N} b_k \frac{\partial \tilde{u}}{\partial x_k} + c\tilde{u} \tag{1.2}$$

where N is the number of dimensions in (1.1), $\mathbf{r}$ is a known function of the independent variables x_i $(i = 1, \ldots N)$, and $\tilde{u}$ is the dependent variable.

If all a_{ij} in (1.2) are equal to zero, then (1.1) is of first order. If any $a_{i,j}$ is different from zero, then (1.1) is of second order. The number of dimensions N is greater or equal to two. If $N = 1$ then (1.1) reduces to a *total differential equation.*

Different types and categories of (1.1) arise from the nature of the physical problem which it expresses. Problems which exhibit solutions of transient nature, or represent an unsteady state, are known as *propagation problems.* They involve time as one of the independent variables in (1.1), and aim at predicting the subsequent behaviour of a system, given its initial state. The type of (1.1) for such problems is either *parabolic* or *hyperbolic* or it may be of mixed type. Hyperbolic PDEs are usually associated with unattenuated convective or advective motions like the Shallow Water equations (7.14), while parabolic equations are associated with propagation problems involving a dissipation mechanism, usually through heat conduction (e.g equation 4.91) or viscous shear (e.g equation 7.2).

On the other hand, equilibrium problems (e.g the *steady flow* problem of equations 7.3) involve finding the steady state configuration of the dependent variable $\tilde{u}$ in a bounded region $\Re$, which satisfies the differential equation throughout $\Re$ and satisfy certain *boundary conditions* on the boundary $\partial\Re$ of $\Re$. Equations of the form (1.1) for such problems will not involve any time derivatives and fall into the category of *elliptic* equations.

Equation (1.1) can be further classified in terms of linearity or nonlinearity, depending on the coefficients a_{ij}, b_k, and c appearing in equation (1.2).

- Equation (1.1) is said to be *non-linear* if any of the coefficients a_{ij}, b_k, and c is a function of $\tilde{u}$ or one of its derivatives.

- If the coefficients a_{ij} are functions only of $\tilde{u}$, and any of its first order derivatives, but not of its second order derivatives, (1.1) is said to be *quasi linear.*

- If all a_{ij} are functions of the independent variables x_k (k=1 ...N) only, then (1.1) is *semi linear.*

- Further if b_k and c are constants, then (1.1) is *linear.*

- If (1.1) is linear, with r=0, then it is said to be homogeneous.

The *mathematical* classification of PDEs and systems of PDEs into one of the three categories (i.e hyperbolic, parabolic and elliptic) is given in section 3.1.

1.2 The thesis outline

This thesis is concerned with improving and extending the AGE method for the solution partial differential equation using finite difference approximations. The AGE method is an iterative method which solves tridiagonal systems and banded systems which arise from the approximation of differential equations by finite difference schemes using central difference operators.

The AGE method is not expected to compete with direct methods in terms of the computational cost. Direct methods are more efficient. They are also very stable relative to the growth of rounding errors when solving positive definite matrix systems. The AGE-1D method is however known to converge in a small number of iterations, and its explicit nature allows a maximum exploitation of parallel computers. This justifies considering further development and analysis of the method which can make it more efficient to use competitively.

In this thesis, chapter two introduces some basic definitions and theories in numerical analysis which are necessary tools most of which are referred to later in the thesis, while chapter three contains a classification of the partial differential equations and introduces some finite difference schemes and their various properties. It also includes a survey of direct and iterative methods which are used for the solution of various linear systems. Chapter 4 introduces the Alternating Direction Implicit methods in two and three dimensions. Also, the AGE method is introduced in this chapter and the relevant literature on the method is surveyed.

In chapter 5, the method is developed by extending the range of application of the AGE-1D algorithm and analysing its convergence for systems arising from the use of unsymmetric central difference operators and also analyzing its convergence

when applied to block symmetric systems. Also a preliminary consideration of the possibility of a Chebyshev acceleration of the AGE-1D method is carried out by determining the conditions under which the eigenvalues of its iteration matrix are real.

Also chapter 5 contains a listing of all the variants of the AGE-1D method with their computational and storage requirements. Also included is a consistency analysis of the AGE-1D and AGE-2D algorithms which were omitted in the literature on the AGE method, together with a convergence analysis for the AGE-2D algorithm.

In chapter 6, the EAD method which is based on combining the applications of the AGE-1D algorithm with the ADI techniques is introduced. The EAD method is then applied to some model problems in two and three dimensions, and a comparative analysis of the computational costs of the EAD method and the AGE-2D and AGE-3D methods is carried out with results showing great savings achieved by using the EAD method.

In chapter 7, the AGE-1D method and the EAD method are applied to solving multivariate equations which are respectively the equations of a one dimensional idealized planetary boundary layer model, and a linearized model of the shallow water equations in two dimensions.

The thesis ends with a chapter on conclusions and suggestions for further research.

Chapter 2

Basic principles of numerical analysis

This chapter introduces the basic concepts, definitions, and rules which are necessary tools in numerical analysis and to which referrence is made in the course of this thesis.

2.1 Matrices and Vectors

A *matrix* is a rectangular array of scalars. These scalars are called the elements of the matrix. They may be complex or real, and have a general representation of a_{ij}. The subscripts i and j indicate respectively the row and column numbers determining the position of the element in the matrix. The size of a matrix is determined by its number of rows m and its number of columns n. The size of the matrix is said to be of size $m \times n$, and is denoted as $A_{m \times n}$. If m=n, then the matrix is said to be a square matrix of order n.

A *vector* is a matrix with one row (*row vector*) or one column (*column vector*). A vector is defined in the real Euclidean space $\mathcal{R}^n$, if all its elements are real. If any of its elements is complex, then it is defined in a complex Euclidean space $\mathcal{C}^n$. The

dimension of the space is determined by the number of elements of the vector. A set of vectors $\mathbf{v}^k, k = 1 \cdots n$ belonging to a space $\mathcal{C}^n$ (or $\mathcal{R}^n$) are *linearly dependent* if there exists n complex (or real) numbers α_k $(k = 1 \cdots n)$, not all equal to zero, such that:

$$\alpha_1 \mathbf{v}^{(k)} + \alpha_2 \mathbf{v}^{(2)} + \cdots + \alpha_n \mathbf{v}^{(n)} = 0 \tag{2.1}$$

If no such set of numbers exist, the vectors $\mathbf{v}^{(k)}$ are said to be *linearly independent.* They form a *basis* for the space $\mathcal{C}^n$ (or $\mathcal{R}^n$). A fundamental property of a *basis* is that any vector $\mathbf{u} \in \mathcal{C}^n$ (or $\mathcal{R}^n$) can be represented uniquely as a linear combination of the base vectors. i.e we can write:

$$\mathbf{u} = \sum_{k=1}^{n} c_k \mathbf{v}^{(k)} \tag{2.2}$$

where c_k $(k = 1, 2, \cdots n)$ are scalars. Basic algebraic operations on matrices and vectors can be performed if the matrix/vector operands are conformable with each other for the respective operations. Thus, for the matrices A, B, C and D, and the vectors $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$, we may have the following equalities:

$$\begin{aligned} C_{m,q} &= A_{m,n} \times B_{p,q} \quad (conformable\ only\ if\ n = p) \\ where\ c_{ij} &= \sum_{k=1}^{n} a_{ik} b_{kj} \\ C_{m,q} &= A_{m,n} + B_{p,q} \quad (conformable\ only\ if\ m = p\ and\ n = q) \\ where\ c_{ij} &= a_{ij} + b_{ij} \end{aligned}$$

The inner product of two vectors: The *inner* or *scalar* product of two vectors $\mathbf{y}$ and $\mathbf{z}$ belonging to the space $\mathcal{C}^n$, is denoted by $(\mathbf{y}, \mathbf{z})$ and is defined as: $(\mathbf{y}, \mathbf{z}) = \sum_{i=1}^{n} y_i \overline{z}_i$ where $\overline{z}_i$ is the *complex conjugate* of z_i.

Two vectors $\mathbf{x}$ and $\mathbf{y}$ are said to be *orthogonal* if their scalar product $(\mathbf{x}, \mathbf{y}) = 0$. A system of vectors is *orthogonal* if any two vectors in the system are orthogonal.

The *length* of a vector is given as:

$$\sqrt{(\mathbf{x},\mathbf{x})} = \sqrt{\sum_{i=1}^{n} x_i \bar{x}_i} \quad ; \; for \;\; \mathbf{x} \in \mathcal{R}^n \;\; or \;\; \sqrt{(\mathbf{x},\mathbf{x})} = \sqrt{\sum_{i=1}^{n} x_i{}^2}$$

If all the elements of a vector $\mathbf{x}$ are divided by a scalar α, such that the length of the vector becomes equal to unity, then the vector is said to be *normalized.*

Two normalized vectors $\mathbf{x}$ and $\mathbf{y}$ which are orthogonal are said to be *orthonormal.*

Theorem 2.1 *Vectors forming an orthogonal system, are linearly independent.*

Positive Definite matrix: A matrix A is said to be positive definite if for all $\mathbf{x} \in \mathcal{R}^n$, $\mathbf{x} \neq 0$ and $A \in \mathcal{R}^{n,n}$ we have:

$$(\mathbf{x}, A\mathbf{x}) > 0 \tag{2.3}$$

A is said to be *positive* semi definite if:

$$(\mathbf{x}, A\mathbf{x}) \geq 0 \quad for\ all\ \mathbf{x} \neq 0 \tag{2.4}$$

with the equality holding for at least one $\mathbf{x} \in \mathcal{C}^n$.

2.2 Common types of matrices

We now define the following matrices:

Diagonal matrix: Any square matrix D whose off diagonal elements are equal to zero, (i.e $d_{ij} = 0$ for $i \neq j$) is said to be a *diagonal* matrix.

Identity matrix: Any diagonal matrix I whose entries along the diagonal are equal to 1 is said to be an *identity* matrix.

Null or zero matrix: This is a matrix whose elements are all zeros.

Band matrix: A matrix A is said to be a *Banded* matrix, of band 2p+1 if its elements $a_{ij} = 0$ for $\mid i - j \mid > p$. If p=1, then A is a *tridiagonal* matrix.

Other properties: A matrix A is said to be *symmetric* if $a_{ij} = a_{ji}$, and *skew symmetric* if $a_{ij} = -a_{ij}$. It is said to be a *hermitian* matrix if $a_{ij} = \overline{a}_{ij}$, where the bar indicates the complex conjugate of a. If $a_{ij} = -\overline{a}_{ij}$ then the matrix is *skew hermitian*. A is said to be *unitary* if $AA^H = I$ (A^H being the the conjugate transpose of A) and A is *normal* if $AA^H = A^H A$.

Non Singular matrix: A square matrix A is said to be a *nonsingular* matrix if there exists a matrix Q such that $AQ = QA = I$, Q is called the *inverse* of A and is usually denoted as A^{-1}.

2.3 Properties of matrices

Definition 2.1 Determinant of a matrix *The determinant of a square matrix A is a scalar quantity denoted by 'det A' or | A | and is given by:*

$$| A | = \sum_{j_1, j_2, \ldots, j_n} p(j_1, j_2, \ldots, j_n) a_{1j_1}, a_{2j_2}, \ldots a_{nj_n} \tag{2.5}$$

where $p(j_1, j_2, \ldots, j_n)$ *is a permutaion equal to* ∓ 1, *given in general as:*

$$p(j_1, j_2, \ldots, j_n) = sign \prod_{1 \leq s < r \leq n} (j_r - j_s) \tag{2.6}$$

For a singular matrix $| A | = 0$

Theorem 2.2 *If for a vector* x, *and a matrix A, with* $| A | \neq 0$ *then* Ax=0 $\Rightarrow$ x=0

Definition 2.2 Trace of a matrix: *This is denoted by* $tr(A)$, *and is equal to the sum of the diagonal elements in a square matrix.*

Definition 2.3 Eigenvalues of a matrix: *An eigenvalue of a matrix* $A_{n \times n}$ *is a real, or complex scalar* λ, *which for some nonzero vector* y *satisfies the relation:*

$$A\mathbf{y} = \lambda \mathbf{y} \text{ or } (\lambda I - A)\mathbf{y} = \mathbf{0} \tag{2.7}$$

where **0** *is a zero vector.*

The corresponding vectors **y** which satisfy equation (2.7) are the eigenvectors of A. A has n eigenvalues which can be all distinct or repeated up to a multiplicity n, depending on the shape and properties of A. Similarily A has n eigenvectors some of which may be identical. From theorem (2.2), equation (2.7) can have a non-trivial solution only if :

$$| \lambda I - A | = 0 \tag{2.8}$$

The expansion of the determinant $| \lambda I - A |$ yields an n^{th} degree polynomial in λ, $f(\lambda)$, called the *Characteristic Polynomial*, and the roots of the equation (2.8) are the eigenvalues of the matrix A. Equation (2.8) is called the *Characteristic Equation*. The Characteristic Polynomial is given as :

$$f(\lambda) = | \lambda I - A | = \lambda^n + a_{n-1}\lambda^{n-1} + \cdots + a_1\lambda + a_0 \tag{2.9}$$

where $-a_1 =$ (the sum of the eigenvalues) $= tr(A)$

Theorem 2.3 The Cayley - Hamilton theorem. *This theorem states that any square matrix A satisfies its own Characteristic equation. Thus,*

$$F(A) = A^n + a_{n-1}A^{n-1} + \ldots + a_1A + a_0I = \prod_{i=1}^{n}(A - \lambda_i I) = 0_{n,n}$$

Theorem 2.4 Gerschgorin's First Theorem. *This theorem states that the modulus of the largest eigenvalue of a square matrix is less or equal to the* maximum sum of the *moduli of the elements along any row or any column.*

Theorem 2.5 Gerschgorin's Circle Theorem. *This theorem states that the eigenvalues of a matrix A lie in the union of the discs given by :*

$$| z - a_{ii} | \leq \sum_{\substack{j=1 \\ j \neq i}}^{n} | a_{ij} | \quad (i = 1, 2 \ldots n)$$

in the complex plane z.

The *spectral radius* of A is the radius of the smallest circular disc in the complex plane with centre (a_{ii}) at the origin, which contains all the eigenvalues of A. It is denoted by ρ. From theorem 2.5 it is given as the modulus of the largest eigenvalue of A.

The eigenvalues of a tridiagonal matrix A of the form given as:

$$A = \begin{bmatrix} a & b & & & \\ c & a & b & & 0 \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \ddots \\ & 0 & & c & a & b \\ & & & & c & a \end{bmatrix} \qquad \begin{array}{l} \textit{where } a, b, \textit{ and } c \textit{ are real} \\ \textit{with } bc > 0 \end{array} \tag{2.10}$$

are given by the following formula:

$$\lambda_s = a + 2\sqrt{bc}\cos\frac{s\pi}{n+1} \quad (s = 1, 2, \ldots n) \tag{2.11}$$

An extension of the above formula to apply to block tridiagonal matrices with real elements, is derived and tested next. This for a square matrix A of the form:

$$A = \begin{bmatrix} D & H & & & \\ V & D & H & & 0 \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \ddots \\ & 0 & & V & D & H \\ & & & & V & D \end{bmatrix}_{nm \times nm} \qquad \begin{array}{l} \textit{where } D, H, \textit{ and } V \textit{ are} \\ \textit{any commutative matrices} \\ \textit{having real eigenvalues.} \end{array} \tag{2.12}$$

is given as:

$$\lambda_{s,j}(A) = \lambda_j(D) + 2\sqrt{\lambda_j(H)\lambda_j(V)} \quad \cos\frac{s\pi}{n+1} \qquad \begin{array}{l} (s = 1, 2, \ldots n), \ (j = 1, 2 \ldots m) \end{array} \tag{2.13}$$

where n is the number of blocks along the diagonal. m is the size of the matrices D, H, and V.

Example: Consider the matrix A given as:

$$A = \begin{bmatrix} \begin{bmatrix} 2 & 5 & 0 \\ 5 & 2 & 5 \\ 0 & 5 & 2 \end{bmatrix} & \begin{bmatrix} 1 & 4 & 0 \\ 4 & 1 & 4 \\ 0 & 4 & 1 \end{bmatrix} & & 0 \\ \begin{bmatrix} -3 & 1 & 0 \\ 1 & -3 & 1 \\ 0 & 1 & -3 \end{bmatrix} & \begin{bmatrix} 2 & 5 & 0 \\ 5 & 2 & 5 \\ 0 & 5 & 2 \end{bmatrix} & \begin{bmatrix} 1 & 4 & 0 \\ 4 & 1 & 4 \\ 0 & 4 & 1 \end{bmatrix} & \\ 0 & \begin{bmatrix} -3 & 1 & 0 \\ 1 & -3 & 1 \\ 0 & 1 & -3 \end{bmatrix} & \begin{bmatrix} 2 & 5 & 0 \\ 5 & 2 & 5 \\ 0 & 5 & 2 \end{bmatrix} & \end{bmatrix} \tag{2.14}$$

which is of the form given in (2.12) where

$$D = \begin{bmatrix} 2 & 5 & 0 \\ 5 & 2 & 5 \\ 0 & 5 & 2 \end{bmatrix}; H = \begin{bmatrix} 1 & 4 & 0 \\ 4 & 1 & 4 \\ 0 & 4 & 1 \end{bmatrix}; \text{ and } V = \begin{bmatrix} -3 & 1 & 0 \\ 1 & -3 & 1 \\ 0 & 1 & -3 \end{bmatrix}$$

The eigenvalues of D, H, and V are given as:

$$\lambda(D) = [9.07160678, 2, -5.0710678]$$

$$\lambda(H) = [6.6568542, 1, -4.6568542]$$

$$\lambda(V) = [-1.5857864, -3, -4.4142136]$$

Applying (2.13) to (2.14) give the eigenvalues of A as:

$$\begin{aligned}
\lambda_{1,1} &= 9.07160678 + 2\sqrt{6.6568542 \times (-1.5857864)} \cos\frac{\pi}{4} \\
\lambda_{2,1} &= 9.07160678 + 2\sqrt{6.6568542 \times (-1.5857864)} \cos\frac{\pi}{2} \\
\lambda_{3,1} &= 9.07160678 + 2\sqrt{6.6568542 \times (-1.5857864)} \cos\frac{3\pi}{4} \\
\lambda_{1,2} &= 2 + 2\sqrt{(1) \times (-3)} \cos\frac{\pi}{4} \\
\lambda_{2,2} &= 2 + 2\sqrt{(1) \times (-3)} \cos\frac{\pi}{2} \\
\lambda_{3,2} &= 2 + 2\sqrt{(1) \times (-3)} \cos\frac{3\pi}{4} \\
\lambda_{1,3} &= -5.0710678 + 2\sqrt{(-4.6568542) \times (-4.4142136)} \cos\frac{\pi}{4} \\
\lambda_{2,3} &= -5.0710678 + 2\sqrt{(-4.6568542) \times (-4.4142136)} \cos\frac{\pi}{2} \\
\lambda_{3,3} &= -5.0710678 + 2\sqrt{(-4.6568542) \times (-4.4142136)} \cos\frac{3\pi}{4}
\end{aligned}$$

$$i.e \quad \lambda_{1,1} = 9.07160678 + 4.5948556i$$

$$\lambda_{2,1} = 9.07160678$$

$$\lambda_{3,1} = 9.07160678 - 4.5948556i$$

$$\lambda_{1,2} = 2.0 + 2.4494897i$$

$$\lambda_{2,2} = 2.0$$

$$\lambda_{3,2} = 2.0 + 2.4494897i$$

$$\lambda_{1,2} = 1.3408507$$

$$\lambda_{1,2} = -5.0710678$$

$$\lambda_{1,2} = -11.482986$$

To check the authenticity of the formula (2.13), the Characteristic Polynomial for A was determined, using REDUCE (a package which facilitates the symbolic computation of matrix functions) as:

$$DETA = \mid (VI - A) \mid \tag{2.15}$$

$$\begin{aligned} i.e.\ DETA \ = \ & -(V^9 - 18V^8 - 20V^7 + 2504V^6 - 16708V^5 - 152V^4 \\ & +315168V^3 - 1374592V^2 + 2412576V - 1464640) \end{aligned} \tag{2.16}$$

Solving for the roots of DETA=0, using an IMSL library subroutine gives the values of the nine eigenvalues of A as:

Real Part	*Imaginary Part*
9.07106781186548	+4.49485564215113
9.07106781186548	0
9.07106781186548	−4.49485564215113
2.0	+2.44948974278318
2.0	0
2.0	−2.44948974278318
1.34085065082633	0

$-11.4829862745573 \qquad 0$

$-5.07106781186547 \qquad 0$

which agrees with the values obtained using formula (2.13). This exercise was repeated for several such block tridiagonal matrices and always formula (2.13) has given the correct results.

For A *triangular* matrix the eigenvalues are the elements along the main diagonal. For a general matrix the eigenvalue which is often of most interest is the one with the largest modulus, i.e the Spectral Radius. This may be obtained numerically using an iterative procedure called the *power method* (see [47]). If all the eigenvalues are required the QR algorithm described in [44] is the usual method.

2.4 Vector norms and related matrix norms

The norm of a vector is a non-negative real scalar which is usually used to represent the magnitude of any vector $\mathbf{x}$ in its space C^n (or $\mathbf{R}^n$). It is denoted by $\| \mathbf{x} \|$, and it satisfies the following relations:

a) $\| \mathbf{x} \| > 0 \quad for\ \mathbf{x} \neq 0 \quad (positivity)$

b) $\| k\,\mathbf{x} \| = |k| . \| \mathbf{x} \|, \forall\ scalar\ k, \quad (homogeneity)$

c) $\| \mathbf{x} + \mathbf{y} \| \leq \| \mathbf{x} \| + \| \mathbf{y} \| \quad for\ \ all\ \mathbf{x}, \mathbf{y} \in C^n$

The most widely used norm is the *Holder norm*, (or p- norm) which is defined as:

$$\| \mathbf{x} \|_p = \left(\sum_i | x_i |^p \right)^{\frac{1}{p}} \tag{2.17}$$

Special cases of this norm are the 1,2 and ∞ norms given respectively as:

$$\begin{aligned} \| \mathbf{x} \|_1 &= | x_1 | + | x_2 | + \ldots + | x_n | \\ \| \mathbf{x} \|_2 &= \left(| x_1 |^2 + | x_2 |^2 + \ldots + | x_n |^2 \right)^{\frac{1}{2}} \quad (The\ L_2\ norm\ or\ Euclidean\ norm) \\ \& \| \mathbf{x} \|_\infty &= \max_i | x_i | \quad (i = 1, 2, \ldots n), \quad (\ The\ maximum\ norm) \end{aligned}$$

It can be seen that $\| \mathbf{x} \|_2$ is equal to the length of $\mathbf{x}$ in C^n (or $\mathbf{R}^n$).

Similarly a matrix norm is a real scalar which is used as a measure of the *magnitude* of a matrix, which satisfies:

$a)$ $\quad \| A \| > 0 \ for \ A \neq 0$

$b)$ $\quad \| kA \| = | k | \times \| A \| \ \ for \ any \ scalar \ k \in C, \ A \in C^{n,n}$

$c)$ $\quad \| A + B \| \leq \| A \| + \| B \| \quad for \ any \ A, \ B \in C^{n,n}$

$d)$ $\quad \| AB \| \leq \| A \| \times \| B \| \quad for \ any \ A, \ B \in C^{n,n}$

The matrix *Holder* norm which is *subordinate* to the vector Holder norm is defined as:

$$\| A \|_p = \max_{\mathbf{x} \neq \mathbf{0}} \frac{\| A\mathbf{x} \|_p}{\| \mathbf{x} \|_p} \tag{2.18}$$

for which the following relation holds:

$$\| A\mathbf{x} \|_p \leq \| A \|_p \times \| \mathbf{x} \|_p \quad for \ all \ \mathbf{x} \in C^n \tag{2.19}$$

with equality holding for at least one $\mathbf{x} \neq 0$. Relation (2.19) implies that $\| A \|$ serves as the upper bound for the *amplification power* of A.

Again, the special cases $\| A \|_1$, $\| A \|_2$, $\| A \|_\infty$, are the most commonly used. These are given as:

$$\| A \|_1 = \max_j \sum_{i=1}^{n} | a_{i,j} |$$

$$\| A \|_\infty = \max_i \sum_{j=1}^{n} | a_{i,j} |$$

$$\| A \|_2 = (The \ maximum \ eigenvalue \ of \ A^H A)^{\frac{1}{2}} = [\rho(A^H A)]^{\frac{1}{2}}$$

It can be seen that $\| A \|_1$ ($\| A \|_\infty$) is the largest of the sums of the magnitudes of the elements along any column (row) in A. Also $\| A \|_2 = \rho(A)$ if A is symmetric or Hermitian, and thus this norm is referred to as the *spectral norm.*

2.5 Other definitions and theorems

Definition 2.4 *A matrix A is said to be convergent to zero if the sequence of matrices* A^1, A^2, A^3,... A^n *converges to the null matrix i.e. if for any norm:*

$$lim_{n\to\infty} \| A^n \| = 0 \tag{2.20}$$

It follows from the property (d) of matrix norms (i.e. $\| AB \| \leq \| A \| \times \| B \|$) that a sufficient condition for A to be *convergent to zero* is:

$$\| A \| < 1 \tag{2.21}$$

The following important theorem has a simple proof given in [50].

Theorem 2.6 *A necessary and sufficient condition for A to be* convergent to zero *is:*

$$\rho(A) < 1 \tag{2.22}$$

Other important theorems and definitions which are referred to in this thesis are given below.

Theorem 2.7 *The eigenvalues of the inverse to a matrix A are the reciprocals of the eigenvalues of A. i.e*

$$\lambda_i(A^{-1}) = \frac{1}{\lambda_i(A)}$$

Theorem 2.8 *If A and B are square matrices which commute and have distinct eigenvalues, then they share a complete set of simultaneous eigenvectors.*

Definition 2.5 Similar matrices and Similarity Transfomations If a matrix A is postmultipled by any non-singular matrix L and premultiplied by L^{-1}, to produce a matrix $T = L^{-1}AL$, then is said to have undergone a *similarity transformation.* The matrices A and T are said to be *similar* .

Similar matrices share the same eigenvalues λ_i, and for any eigenvector x of A the corresponding eigenvector of T is L^{-1}x.

Theorem 2.9 *For any square matrix A, there is a unitary matrix L=U such that* $U^{-1}AU = T$ *is upper triangular.*

The eigenvalues of A must be shared by the similarity matrix T, and appear along its main diagonal. When T is diagonal, it is usually denoted by Λ, and A is said to be *diagonalizable.* The proof for theorem 2.9 is given in [45].

Theorem 2.10 *Any square matrix which has linearly independent eigenvectors can be diagonalized by the similarity transformation* $U^{-1}AU$ *where the columns of U are the eigenvectors of A.*

Theorem 2.11 *If A is a* positive definite matrix, *then there exists a lower triangular matrix L such that* $LL^T = A$.

Chapter 3

Current methods for the finite difference solution of partial differential equations

In this chapter the procedures and steps that are involved in the numerical finite difference solution of partial differential equations are introduced. We first describe how the partial differential equations and systems of equations are mathematically classified into their various categories.

3.1 Classification of partial differential equations

3.1.1 Classification of first and second order partial differential equations

The classification of PDEs is pursued for first and second order two dimensional equations and then generalized to equations of higher dimensions.

Consider the second order equation:

$$A\frac{\partial^2\tilde{u}}{\partial x^2}+B\frac{\partial^2\tilde{u}}{\partial x\partial y}+C\frac{\partial^2\tilde{u}}{\partial y^2}+D\frac{\partial\tilde{u}}{\partial x}+E\frac{\partial\tilde{u}}{\partial y}+F\tilde{u}+G=0 \tag{3.1}$$

The solution of such equation has the form:

$$\tilde{u}=\tilde{u}(x,y)$$

which represent a surface in the (x,y,$\tilde{u}$) space called the *integral surface.* If on the integral surface there exist curves across which the partial derivatives $\frac{\partial^2\tilde{u}}{\partial x^2}$, $\frac{\partial^2\tilde{u}}{\partial x\partial y}$, and $\frac{\partial^2\tilde{u}}{\partial y^2}$ are discontinuous or indeterminate, these curves are called the *characteristics* curves ([32]). In the directions of these curves in the (x,y) plane equation (3.1) involves only *total* differentials.

To obtain these *characteristic directions*, it is customary to abbreviate $\frac{\partial\tilde{u}}{\partial x}$, $\frac{\partial\tilde{u}}{\partial y}$, $\frac{\partial^2\tilde{u}}{\partial x^2}$, $\frac{\partial^2\tilde{u}}{\partial x\partial y}$, and $\frac{\partial^2\tilde{u}}{\partial y^2}$ as p, q, r, s, and t respectively. Thus (3.1) becomes:

$$Ar+Bs+Ct+H=0 \tag{3.2}$$

where H represents the remainder of the terms in (3.1).

Along the tangents to any curve C in the (x,y) plane inside the solution domain, u,p, and q satisfy the total differential formulae:

$$d\tilde{u} = \frac{\partial\tilde{u}}{\partial x}dx+\frac{\partial\tilde{u}}{\partial y}dy \tag{3.3}$$

$$dp = rdx+sdy \tag{3.4}$$

$$dq = sdx+tdy \tag{3.5}$$

If (3.4) and (3.5) were used to eliminate r and t in (3.2) we get:

$$-\left[A\left(\frac{dp}{dx}\right)+H\right]\frac{dy}{dx}-C\frac{dq}{dx}+s\left[A\left(\frac{dy}{dx}\right)^2-B(\frac{dy}{dx})+C\right]=0 \tag{3.6}$$

If C is chosen so that the tangents to it at any point has a slope $\frac{dy}{dx}$ which satisfies:

$$\left[A(\frac{dy}{dx})^2-B(\frac{dy}{dx})+C\right]=0 \tag{3.7}$$

then the term involving s vanishes, and (3.6) reduces to a relation between the total differentials of p and q given as:

$$\left[A(\frac{dp}{dx}) + H\right] \frac{dy}{dx} + C\frac{dq}{dx} = 0 \tag{3.8}$$

Thus, the roots of equation (3.7) determines the characteristics directions for which (3.8) holds. These characteristics relate directly to the classification of the PDE's, and to their numerical and analytic methods of solution.

Equation (3.7) has two real distinct, one real double, or conjugate complex, roots depending whether its discriminant $B^2 - 4AC$ is positive, equal to zero, or negative respectively. Where two real roots exist, it implies the existence of two real characteristics and (3.2) is said to be *hyperbolic.* If a double root exists, then there is only one real characteristic and (3.2) is a *parabolic* PDE. If no real roots exist, then the characteristics are complex and (3.2) is an *elliptic* equation.

For first order two dimentional PDEs, the characteristic directions at any point point in the solution domain, are defined as the directions along which the first order derivatives are undetermined, or not defined uniquely, and the PDE reduces to a simpler *ordinary* differential equation in $\tilde{u}$. Any first order equation in two dimensions has one real characteristic, the direction of which is obtained as illustrated by the following example. Consider the equation:

$$C\frac{\partial \tilde{u}}{\partial x} + D\frac{\partial \tilde{u}}{\partial y} = E \tag{3.9}$$

Along the tangential direction to some curve C in the (x,y) plane in the solution domain, the following total differential formula is satisfied:

$$d\tilde{u} = \frac{\partial \tilde{u}}{\partial x}dx + \frac{\partial \tilde{u}}{\partial y}dy \tag{3.10}$$

If we substitute for $\frac{\partial \tilde{u}}{\partial x}$ in (3.9) from (3.10) and rearrange we get:

$$C\frac{d\tilde{u}}{dx} + (D - C\frac{dy}{dx})\frac{\partial \tilde{u}}{\partial y} = E \tag{3.11}$$

If C is chosen so that:

$$D - C\frac{dy}{dx} = 0 \tag{3.12}$$

then $\frac{\partial \tilde{u}}{\partial y}$ is eliminated from (3.11) which becomes the ODE:

$$\frac{d\tilde{u}}{dx} = \frac{E}{C} \tag{3.13}$$

If on the other hand, we substituted for $\frac{\partial \tilde{u}}{\partial y}$ in (3.9) from (3.10) instead of $\frac{\partial \tilde{u}}{\partial x}$, we get a similar total differential equation given as:

$$\frac{d\tilde{u}}{dy} = \frac{E}{D} \tag{3.14}$$

which again only holds along the same characteristic direction

$$\frac{dy}{dx} = \frac{D}{C}$$

obtained from (3.13). Thus, equation (3.9) is always *hyperbolic.* Finally we note that the classification of PDEs is independent of the co-ordinate system used.

3.1.2 Classification of multidimensional PDEs and systems of PDEs

Consider the multidimensional second order PDE in the general form given as:

$$\sum_{j=1}^{N}\sum_{i=1}^{N} a_{ij}\frac{\partial^2 \tilde{u}}{\partial x_i \partial x_j} + H = 0 \tag{3.15}$$

where $N \geq 2$ and $H = f(x, y, \tilde{u}, \frac{\partial \tilde{u}}{\partial x_i})$, $\quad i, j = 1 \ldots N.$

A classification for (3.15) is given by [5], depending on the eigenvalues of the matrix A whose elements are a_{ij} as follows:

a) If any of the eigenvalues is zero, (3.15) is *parabolic.*

b) If all the eigenvalues are non-zero and are of the same sign, (3.15) is *elliptic.*

c) If all the eigenvalues are non zero, and all but one, are of the same sign (3.15) is *hyperbolic.*

In what follows, the classification of systems of first order PDEs is considered, for two and higher dimensions. It is noted that systems of second order equations could be transformed, with the aid of some auxiliary variables to larger systems of first order equations, and classified accordingly.

Consider the following system of n equations in two dimensions:

$$\left.\begin{array}{rcrcl} c_{1i}\frac{\partial \tilde{u}_i}{\partial x} & + & d_{1i}\frac{\partial \tilde{u}_i}{\partial y} & = & e_1 \\ \vdots & + & \vdots & = & \vdots \\ c_{ji}\frac{\partial \tilde{u}_i}{\partial x} & + & d_{ji}\frac{\partial \tilde{u}_i}{\partial y} & = & e_j \\ \vdots & + & \vdots & = & \vdots \\ c_{ni}\frac{\partial \tilde{u}_i}{\partial x} & + & d_{ni}\frac{\partial \tilde{u}_i}{\partial y} & = & e_n \end{array}\right\} \begin{array}{l} i = 1 \ldots n \\ \\ j = 1 \ldots n. \end{array} \tag{3.16}$$

n is also the number of independent variables.

This system can be written in matrix form as:

$$C\frac{\partial \bar{u}}{\partial x} + D\frac{\partial \bar{u}}{\partial y} = E \tag{3.17}$$

where C and D are square matrices with elements c_{ji} and d_{ji} respectively, with $\bar{u} = [\tilde{u}_1, \tilde{u}_2 \ldots \tilde{u}_n]^T$ and $E = [e_1, e_2, \ldots e_n]^T$.

We may seek n characteristic directions to the system (3.17), which may be rewritten after premultiplication by C^{-1} (assuming C is non singular) as:

$$\frac{\partial \bar{u}}{\partial x} + A\frac{\partial \bar{u}}{\partial y} = B \tag{3.18}$$

where $A = C^{-1}D$ and $B = C^{-1}E$. The characteristic directions for (3.18) can be obtained in a similar manner to the derivation of the characteristic direction of the scalar equation (3.18). Thus, the following analogue to (3.13) is obtained.

$$\frac{\partial \bar{u}}{\partial x} + (A - I\frac{dy}{dx})\frac{\partial \bar{u}}{\partial y} = B \tag{3.19}$$

Thus, (3.18) reduces to a system of total differential equations if:

$$(A - I\frac{dy}{dx})\frac{\partial \bar{u}}{\partial y} = \vec{0} \tag{3.20}$$

A necessary condition for (3.20) to hold without having a trivial solution for $\frac{\partial \bar{u}}{\partial y}$ is that:

$$det\left[A - I\frac{dy}{dx}\right] = 0$$

The slopes of the characteristic directions for (3.18) are thus given as the eigenvalues of the matrix A.

If A has n real eigenvalues with n linearly independent eigenvectors $\frac{\partial \bar{u}}{\partial y}$ then (3.18) is *hyperbolic.*

If A has q real eigenvalues, where $1 \leq q \leq n-1$, and no complex eigenvalues, (i.e some eigenvalues are equal) then (3.18) is *parabolic.*

If A has no real eigenvalues (i.e they are all complex) then (3.18) is *elliptic.*

If A has real and complex eigenvalues then (3.18) is a mixed system. From the above, it can be seen that it is sufficient for A to be *symmetric* for (3.18) to be hyperbolic, and for A to be *skew symmetric* for (3.18) to be elliptic.

Classification of systems of PDEs in more than two dimensions is given in [30] and [5] among others. Here it will only be noted that systems in three dimensions such as:

$$\frac{\partial \bar{u}}{\partial x} + A\frac{\partial \bar{u}}{\partial y} + B\frac{\partial \bar{u}}{\partial z} = \vec{0} \tag{3.21}$$

are considered hyperbolic if " for all α, β with $\alpha^2 + \beta^2 = 1$, there exist a nonsingular transformation matrix P such that:

$$P(\alpha A + \beta B)P^{-1} = D \tag{3.22}$$

where D is a diagonal matrix with real elements. The symmetry of A and B is sufficient to guarantee that the above system is *hyperbolic*"([34] p:180).

3.1.3 Boundary and Initial value Problems

For the solution of a PDE to be unique, appropriate conditions on certain line/s or surface/s (non characteristic) should be prescribed. Take for example, a two

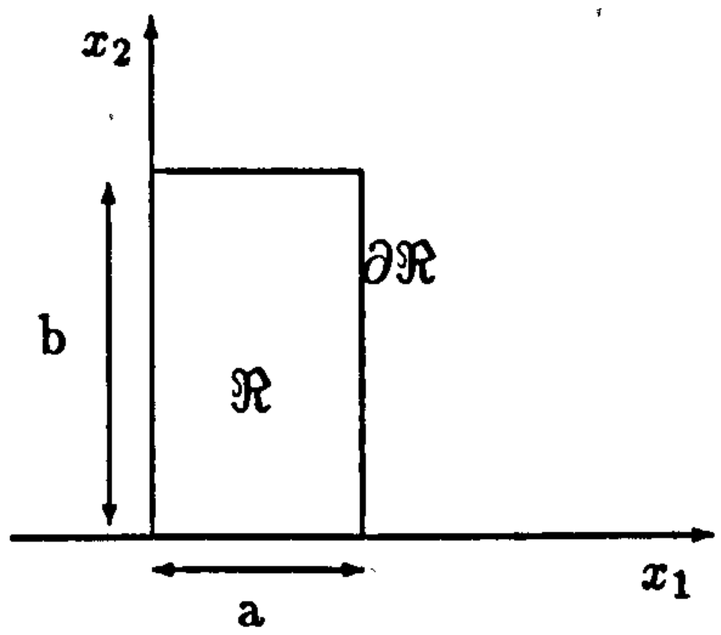


Figure 3.1: A rectangular domain of solution for a 2D pde

dimensional PDE, whose solution is defined and sought in the region $\Re$ shown in figure 3.1.

- If only initial conditions are prescribed along the x_1 axis (or any line $x_2 =$ *constant*) then the problem defined by the PDE is an *initial value problem*, and the conditions prescribed to it are often referred to as *Cauchy conditions.*

- If some conditions are prescribed on the x_2 axis as well (or any line $x_1 =$ *const*) then the problem defined by the PDE is an *initial-boundary* value problem. If further conditions are prescribed on $x_1 = a$, then the problem is overprescribed.

- If certain conditions are given at the boundary $\partial\Re$ of $\Re$, then the problem defined by the PDE is a *boundary value problem.*

The type of conditions which may be prescribed fall in one of the following categories:

Dirichlet Conditions: For such conditions the dependent variable ($\tilde{u}$) satisfies

$$\tilde{u} = f$$

where f is known function.

Neumann or derivative conditions: Here the normal derivative of the function satisfies a given function along a certain line or boundary e.g $\frac{\partial \tilde{u}}{\partial x_2} = f$ (on the x_1-axis).

Robin or mixed conditions: Here the dependent variable satisfies a combination of the above Dirichlet and Neumann conditions.

$$e.g \quad \frac{\partial \tilde{u}}{\partial x_2} + K\tilde{u} = f, \quad K \neq 0 \quad (along\ the\ x_1\ axis)$$

3.2 Discretization and the derivation of finite difference formulae

Finite difference methods for solving differential equations are derived by replacing the continuous derivatives of a function, in the differential equation, by finite dif-

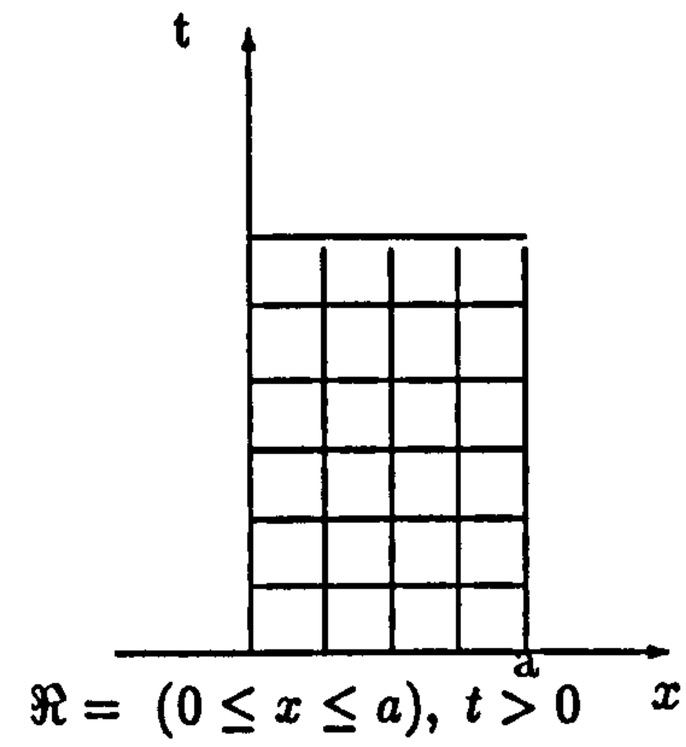


Figure 3.2: A rectilinear grid of meshpoints

ference expressions, involving the values of the function at discrete points defined by a grid covering the region $\Re$ of the solution. The first step therefore, is the construction of a rectilinear grid covering $\Re$ with lines parallel to the coordinates of the independent variables x_i. If one of the independent variables is time (fig: 3.2), then the resulting finite difference schemes may be implicit or explicit, depending on whether the spatial derivatives are evaluated at the current or subsequent time steps. This is illustrated, following the approach of [34], by considering the following

general equation given as:

$$\frac{\partial \tilde{u}}{\partial t} = L\tilde{u} \tag{3.23}$$

where $L(x_i, D_i^j)$ is linear. $i = 1 \ldots p$; $j = 1 \ldots q$

and where p is the number of space dimensions, q is the order of the differential equation, and $D_i = \frac{\partial}{\partial x_i}$.

From the Taylor series expansion of $\tilde{u}^{t+k}$, at any chosen point in the grid we have:

$$\begin{aligned} \tilde{u}^{t+k} &= (1 + k\frac{\partial}{\partial t} + \frac{1}{2}k^2\frac{\partial^2}{\partial t^2} + \ldots)\tilde{u}^t \quad (k \text{ is the time step}) \\ &= \exp(k\frac{\partial}{\partial t})\tilde{u}^t = \exp(kL)\tilde{u}^t \quad (\text{from equation } 3.23) \end{aligned} \tag{3.24}$$

which leads to a general expression for exact difference formulae, having the form:

$$\tilde{u}^{t+k} = \exp[(1 + \theta - \theta)kL]\tilde{u}^t \tag{3.25}$$

$$\Leftrightarrow \exp(-\theta kL)\tilde{u}^{t+k} = \exp[(1 - \theta)kL]\tilde{u}^t \tag{3.26}$$

The difference formula is *explicit* if $\theta = 0$ and implicit otherwise. i.e $\theta \neq 0$.

Exact formulae expressing derivatives at a point in terms of finite difference operators (defined below), and vice versa, are dealt with in great detail in [31].

If in the mesh of gridpoints shown in figure 3.2 every gridpoint T is denoted by m and n. Then $m = n = 0$ defines the gridpoint $O(0, 0)$ at the origin, and $T_{m,n}$ defines a point whose ordinates are (mh, nk) where h and k are the gridspacings in the t and x directions respectively.

A set of important difference operators are defined as follows:

$$\begin{aligned} \delta_x U_m^n &= U_{m+\frac{1}{2}}^n - U_{m-\frac{1}{2}}^n && (\textit{Central difference operator}) \\ \nabla_x U_m^n &= U_m^n - U_{m-1}^n && (\textit{Backward difference operator}) \\ \triangle_x U_m^n &= U_{m+1}^n - U_m^n && (\textit{forward difference operator}) \end{aligned}$$

Higher order (central, backward, and forward) difference operators can be defined recursively, however, only the second order difference operators will be given below

together with other useful operators.

$$\begin{aligned}
\delta_x^2 U_m^n &= \delta_x(U_{m+\frac{1}{2}}^n - U_{m-\frac{1}{2}}^n) = U_{m+1}^n - 2U_m^n + U_{m-1} \\
\nabla_x^2 U_m^n &= \nabla x(U_m^n - U_{m-1}^n) = U_m^n - 2U_{m-1}^n + U_{m-2}^n \\
\Delta_x^2 U_m^n &= \Delta_x(U_{m+1}^n - U_m^n) = U_{m+2}^n - 2U_{m+1}^n + U_m^n \\
\mu_x U_m^n &= \frac{1}{2}(U_{m+\frac{1}{2}}^n + U_{m-\frac{1}{2}}^n) \quad (\mu_x \textit{ is an averaging operator}) \\
H_x U_m^n &= U_{m+1}^n - U_{m-1}^n \quad (\textit{mean central difference}) \\
E_x U_m^n &= U_{m+1}^n \; ; E_x^\alpha U_m^n = U_{m+\alpha}^n
\end{aligned}$$

where E is called the *shift operator* and α is any constant.

It can be seen that the following relations between these operators are valid:

$$\begin{aligned}
\delta_x U_m^n &= \nabla_x U_{m+\frac{1}{2}}^n = \Delta_x U_{m-\frac{1}{2}}^n \\
\Delta &= E - 1 \quad , \nabla = 1 - E^{-1} \\
\mu &= \frac{1}{2}(E^{\frac{1}{2}} + E^{-\frac{1}{2}}) \quad ; \mu = (1 + \frac{\delta^2}{4})^{\frac{1}{2}} \; ; \\
\delta &= E^{\frac{1}{2}} - E^{-\frac{1}{2}} \quad H_x = 2\mu\delta_x = \Delta_x + \nabla_x
\end{aligned}$$

These difference operators are also related to the derivative D_i^1 (here $\frac{\partial}{\partial x}$ or D) at a point $T_{m,n}$ through the Taylor Series, where:

$$\begin{aligned}
E_x U_m^n = U_{m+1}^n &= (1 + \frac{hD}{1!} + \frac{h^2D^2}{2!} + \ldots + \frac{h^nD^n}{n!})U_m^n \\
\Rightarrow E &= e^{hD}
\end{aligned} \tag{3.27}$$

$$\Delta = E - 1 \quad \Delta = e^{hD} - 1$$

$$\Rightarrow \Delta = (\frac{hD}{1!} + \frac{h^2D^2}{2!} + \ldots \frac{h^nD^n}{n!} \ldots) \tag{3.28}$$

$$\nabla = 1 - E^{-1} = 1 - e^{-hD}$$

$$\nabla = (\frac{hD}{1!} - \frac{(hD)^2}{2!} + \frac{(hD)^3}{3!} - \ldots \frac{(-1)^{n+1}(hD)^n}{n!} + \ldots) \tag{3.29}$$

$$\delta = E^{\frac{1}{2}} - E^{-\frac{1}{2}} = e^{\frac{hD}{2}} - e^{\frac{-hD}{2}} = 2sinh(\frac{hD}{2})$$

$$\Rightarrow \delta = \left[(hD) + \frac{(hD)^3}{2^2.3!} + \ldots \frac{(hD)^5}{2^4.5!} + \ldots \frac{(hD)^n}{2^{2n-1}.n!} + \ldots\right] \tag{3.30}$$

$$\Rightarrow \mu = \frac{1}{2}(E^{\frac{1}{2}} + E^{-\frac{1}{2}}) = \frac{1}{2}(e^{\frac{hD}{2}} + e^{-\frac{hD}{2}}) = cosh(\frac{hD}{2}) \tag{3.31}$$

and $H_x = 2\mu_x\delta_x = 2cosh(\frac{hD}{2}) \times 2sinh(\frac{hD}{2}) = 2sinh(hD)$.

$$\Rightarrow \ H_x = 2[(hD) + \frac{(hD)^3}{3!} + \frac{(hD)^5}{5!} + \ldots] \tag{3.32}$$

Expressions of higher order differences in terms of derivatives can be obtained by simply raising both sides of the respective equation of (3.28...3.32) to the corresponding exponent. Thus, for example, squaring both sides of (3.30) gives δ^2 as:

$$\delta^2 = [(hD)^2 + \frac{1}{12}(hD)^4 + \frac{(hD)^6}{360} + \ldots] \tag{3.33}$$

Thus suitable finite difference approximations of the derivatives in the differential equations, involve truncating the corresponding exact formula of (3.28 ... 3.30). This truncation leads to the introduction of an error known as the *local truncation error* (L.T.E) which is defined as the difference between the differential equation and the difference equation which is used to replace it. The terms in the (L.T.E) which are of least order in the grid spacings h and k represent the *principle part* of the (L.T.E). It can be seen from (3.28) ... (3.30) that the truncation error resulting from the central difference replacement for $\frac{\partial}{\partial x}$ has a principal part proportional to h^2 while forward and backward difference replacements have truncation errors with principal parts proportional to h.

Equations (3.28) ... (3.30) express the relations between the Δ, ∇, *and* δ difference operators and the derivative defined at the nodal gridpoint $T_{m,n}$ which is, at the backward end of the interval $[T_{m,n} \rightarrow T_{m+1,n}]$, at the forward end of the interval $[T_{m,n} \rightarrow T_{m-1,n}]$, and at the centre of the interval $[T_{m+\frac{1}{2},n} \rightarrow T_{m-\frac{1}{2},n}]$.

More generally the derivative at some arbitrary point $T_{m+\theta,n}$, where $(0 \leq \theta \leq 1)$, in the interval $[T_{m,n} \rightarrow T_{m+1,n}]$ may be defined in terms of finite differences involving

the values of the function at $T_{m+1,n}$, and $T_{m,n}$. This is done using the following relations:

$$U^n_{m+1} = E^{(1-\theta)}U^n_{m+\theta} = e^{(1-\theta)hD}U^n_{m+\theta}$$

$$U^n_m = E^{(-\theta)}U^n_{m+\theta} = e^{(-\theta hD)}U^n_{m+\theta}$$

Thus,

$$(e^{(1-\theta)hD} - e^{-\theta hD})U^n_{m+\theta} = U^n_{m+1} - U^n_m \tag{3.34}$$

$$\Rightarrow U^n_{m+1} - U^n_m =$$
$$(hD + \frac{[(1-\theta)^2 - \theta^2]}{2!}h^2D^2 + \frac{[(1-\theta)^3 + \theta^3]}{3!}h^3D^3 + \dots)U^n_{m+\theta} \tag{3.35}$$

which is a general equation whose L.H.S represents a forward, backward or central difference depending on whether $\theta = 0,\ \frac{1}{2},\ or\ 1$ respectively. First and higher order derivatives may also be represented in terms of differences in the values of the corresponding function, at more than two gridpoints. Using more than two gridpoints generally produces difference schemes of higher order accuracies. Consider, for example, expressing the derivative D (*here* $\frac{\partial}{\partial x}$) at $T_{m,n}$ in terms of the values of the function at the points $T_{m-2,n},\ T_{m-1,n},\ T_{m,n},\ T_{m+1,n},\ and\ T_{m+2,n}$. Also we use the relations:

$$U^n_{m+2} = E^2U^n_m = e^{2hD}U^n_m\ ;\ \ U^n_{m+1} = EU^n_m = e^{hD}U^n_m$$

$$U^n_{m-2} = E^{-2}U^n_m = e^{-2hD}U^n_m\ ;\ \ U^n_{m-1} = E^{-1}U^n_m = e^{-hD}U^n_m$$

Then, if a,b,c,d, and f are arbitrary coefficients then:

$$aU^n_{m+2} + bU^n_{m+1} + cU^n_m + dU^n_{m-1} + eU^n_{m-2}$$
$$= (ae^{2hD} + be^{hD} + c + de^{-ehD} + fe^{-2hD})U^n_m$$
$$= (a+b+c+d+f)U^n_m + (2a+b-d-2f)hD$$
$$+ \frac{(2^2a+b+d+2^2f)}{2!}h^2D^2 + \frac{(2^3a+b-c-2^3f)}{3!}h^3D^3$$
$$+ \frac{(2^4a+b+d+2^4f)}{4!}h^4D^4 + \frac{(2^5a+b-d-2^5f)}{5!}h^5D^5$$
$$+ \frac{(2^6a+b+c+2^6f)}{6!}h^6D^6 + \dots + \dots) \tag{3.36}$$

One can henceforth solve for a, b, c, d, and f to give the various finite difference replacements of desired accuracies with truncation error up to $\mathcal{O}\ (h^5)$. The values a,b,c,d and f must satisfy simultaneously, equating the second term on the R.H.S of (3.36) to D, and equating to zero the first term and the other terms up to the fifth, depending on the accuracy required. Thus, for a difference replacement whose truncation error is of order h^4, a,b,c,d, and f should satisfy:

$$\left.\begin{aligned} &a+b+c+d+f=0\\ &(2a+b-d-2f)h=1\\ &4a+b+d+4f=0\\ &8a+b-d-8f=0\\ &16a+b+d+16f=0 \end{aligned}\right\} \tag{3.37}$$

The system (3.37) is satisfied for:

$$a=\frac{1}{12h}\ ;\ \ b=\frac{-8}{12h}\ ;\ c=0\ ;\ d=\frac{8}{12h}\ \ ;\ and\ \ f=\frac{-1}{12h}.$$

These values nullify the sixth term in (3.36) making the truncation error of the resulting difference replacements of order (h^5).

By substituting the values of a,b,c,d and f, we obtain the finite difference equation for $\frac{\partial \tilde{u}}{\partial x}$ as:

$$(\frac{\partial \tilde{u}}{\partial x})_m^n=\frac{U_{m+2}^n-8U_{m+1}^n+8U_{m-1}^n-U_{m-2}^n}{12h}+\mathcal{O}(h^5) \tag{3.38}$$

The use of various finite difference replacements of the derivatives in the PDEs yield symmetric, or asymmetric, implicit or explicit schemes depending on the way the finite difference approximations are applied.

The properties of various finite difference schemes will be discussed in the next section.

3.3 Derivation of some basic finite difference schemes

The most important properties of a finite differences scheme will be considered by looking at some finite difference replacements of the following heat conduction equation in one space dimension:

$$L(\tilde{u}) = \frac{\partial \tilde{u}}{\partial t} - \frac{\partial^2 \tilde{u}}{\partial x^2} = 0 \tag{3.39}$$

defined over the region $\Re = [0 \leq x \leq 1]$, $t \geq 0$, and subject to the initial condition:

$$\tilde{u}(x,0) = g(x)$$

and the boundary conditions

$$\tilde{u}(0,t) = b_1 \quad and \quad \tilde{u}(1,t) = b_2$$

It can be readily seen that replacing $\frac{\partial}{\partial t}$ and $\frac{\partial^2}{\partial x^2}$ using (3.28) and (3.33), where terms other than the first on the RHS of these equations are neglected, yield a finite difference approximation to (3.39), having a local truncation error of the order $\mathcal{O}(h^2)+ \mathcal{O}(k)$. The resulting formula:

$$\begin{aligned} \frac{\Delta_t U_m^n}{k} &= \frac{\delta_x^2 U_m^n}{h^2} + \left(\frac{k}{2}\frac{\partial^2 \tilde{u}}{\partial t^2} - \frac{h^2}{12}\frac{\partial^4 \tilde{u}}{\partial x^4}\right)_m^n + \mathcal{O}(k^2, h^4) \\ \Longrightarrow \frac{U_m^{n+1} - U_m^n}{k} &= \frac{U_{m+1}^n - 2U_m^n + U_{m-1}^n}{h^2} + \mathcal{O}(h^2) + \mathcal{O}(k) \end{aligned} \tag{3.40}$$

is called the *Classical Explicit* scheme. The principal part of its Local Truncation Error (LTE) is $(\frac{k}{2}\frac{\partial^2 \tilde{u}}{\partial t^2} - \frac{h^2}{12}\frac{\partial^4 \tilde{u}}{\partial x^4}{}_m^n)$.

Similarly if (3.39) is approximated implicitly at the point $(mh, (n+1)k)$ where backward and central differences are used respectively for $\frac{\partial}{\partial t}$ and $\frac{\partial^2}{\partial x^2}$ we get the *fully implicit* scheme given by:

$$\begin{aligned} \frac{\nabla_t U_m^{n+1}}{k} &= \frac{\delta_x^2 U_m^{n+1}}{h^2} + \underbrace{\left(-\frac{k}{2}\frac{\partial^2 \tilde{u}}{\partial t^2} - \frac{h^2}{12}\frac{\partial^4 \tilde{u}}{\partial x^4}\right)_m^{n+1}}_{Principal\ part\ of\ the\ L.T.E} \mathcal{O}(k^2, h^4) \\ \Longrightarrow \frac{U_m^{n+1} - U_m^n}{k} &= \frac{U_{m+1}^{n+1} - 2U_m^{n+1} + U_{m-1}^{n+1}}{h^2} + \mathcal{O}(h^2) + \mathcal{O}(k) \end{aligned} \tag{3.41}$$

whose truncation error is of order $\mathcal{O}(h^2)+\mathcal{O}(k)$ If (3.39) is evaluated implicitly at the point $(mh,(n+\frac{1}{2})k)$ and central differences are used to replace $\frac{\partial}{\partial t}$ and $\frac{\partial^2}{\partial x^2}$ we get the Crank-Nicholson scheme given by:

$$\frac{\delta_t U_m^{n+\frac{1}{2}}}{k} = \frac{\mu_t \delta_x^2 U_m^{n+\frac{1}{2}}}{h^2} = \frac{\frac{1}{2}[\delta_x^2 U_m^{n+1} + \delta_x^2 U_m^n]}{h^2} + \mathcal{O}(k^2) + \mathcal{O}(h^2)$$

$$\begin{aligned} \Longrightarrow \frac{U_m^{n+1} - U_m^n}{k} &= \frac{1}{2}\left(\frac{U_{m+1}^{n+1} - 2U_m^{n+1} + U_{m-1}^{n+1} + U_{m+1}^n - 2U_m^n + U_{m-1}^n}{h^2}\right) \\ &+ \left(\frac{k^2}{24}\frac{\partial^3 \tilde{u}}{\partial t^3} - \frac{h^2}{12}\frac{\partial^4 \tilde{u}}{\partial x^4} + \frac{k^2}{8}\frac{\partial^4 \tilde{u}}{\partial t^2 \partial x^2}\right)_m^{n+\frac{1}{2}} + \mathcal{O}(k^4, h^4) \end{aligned}$$

Another scheme which approximates (3.39) explicitly at (mh, nk), but employs a mean central difference H_t for the time derivative, is the Dufort-Frankel method. Here again the central difference δ_x^2 is used to replace the space derivative. In this scheme the U_m^n appearing in the expression for $\delta_x^2 U_m^n$ is replaced by the arithmetic average of U_m^{n+1} and U_m^{n-1}. The resulting formula is given as:

$$\begin{aligned} \frac{U_m^{n+1} - U_m^{n-1}}{2k} &= \frac{U_{m+1}^n - (U_m^{n+1} + U_m^{n-1}) + U_{m-1}^n}{h^2} \\ &+ \left[\frac{k^2}{6}\frac{\partial^3 \tilde{u}}{\partial t^3} - \frac{h^2}{12}\frac{\partial^4 \tilde{u}}{\partial x^4} + \frac{k^2}{h^2}\frac{\partial^2 \tilde{u}}{\partial t^2}\right]_m^n + \mathcal{O}(k^4) + \mathcal{O}(h^6) \end{aligned} \tag{3.42}$$

which has a truncation error of order $\mathcal{O}(h^2, k^2, \frac{k^2}{h^2})$.

An asymmetric scheme, due to Saul'yev [43] is based on taking at the level n+1, the forward/backward "part" of the second order spatial derivative in (3.39) while keeping, at the level n, the remaining backward/forward "part". Thus, $\frac{\partial^2 \tilde{u}}{\partial x^2}$ is approximated as:

$$\frac{\partial^2 \tilde{u}}{\partial x^2} = \frac{1}{h}\left[\left(\frac{\partial \tilde{u}}{\partial x}\right)_{m+\frac{1}{2}}^n - \left(\frac{\partial \tilde{u}}{\partial x}\right)_{m-\frac{1}{2}}^n\right] + \mathcal{O}(h^2) \tag{3.43}$$

$$and \quad \left(\frac{\partial \tilde{u}}{\partial x}\right)_{m\pm\frac{1}{2}}^n = \left(\frac{\partial \tilde{u}}{\partial x}\right)_{m\pm\frac{1}{2}}^{n+1} - k\left(\frac{\partial^2 \tilde{u}}{\partial x \partial t}\right)_{m\pm\frac{1}{2}}^{n+\theta} \qquad 0 \le \theta \le 1 \tag{3.44}$$

If we replace either of $(\frac{\partial \tilde{u}}{\partial x})_{m+\frac{1}{2}}^n$ or $(\frac{\partial \tilde{u}}{\partial x})_{m-\frac{1}{2}}^n$ in (3.43) using (3.44) and then substitute for $\frac{\partial^2}{\partial x^2}$ in (3.39), we get either of these two equations:

$$\frac{\partial \tilde{u}}{\partial t} = \frac{1}{h}\left[\left(\frac{\partial \tilde{u}}{\partial x}\right)_{m+\frac{1}{2}}^{n+1} - \left(\frac{\partial \tilde{u}}{\partial x}\right)_{m-\frac{1}{2}}^n\right] + \mathcal{O}(h^2) + \mathcal{O}(\frac{k}{h}) \tag{3.45}$$

$$\frac{\partial \tilde{u}}{\partial t} = \frac{1}{h}\left[\left(\frac{\partial \tilde{u}}{\partial x}\right)^{n}_{m+\frac{1}{2}} - \left(\frac{\partial \tilde{u}}{\partial x}\right)^{n+1}_{m-\frac{1}{2}}\right] + \mathcal{O}(h^2) + \mathcal{O}(\frac{k}{h}) \tag{3.46}$$

Two formulae can be obtained if forward and central difference replacements were used for the time and space derivatives respectively in (3.45) and (3.46). These formulae are given by:

$$\frac{U^{n+1}_m - U^n_m}{k} = \frac{[U^{n+1}_{m+1} - U^{n+1}_m - U^n_m + U^n_{m-1}]}{h^2} + \mathcal{O}(\frac{k}{h}, h^2, k) \tag{3.47}$$

$$and\ \frac{U^{n+1}_m - U^n_m}{k} = \frac{[U^n_{m+1} - U^n_m - U^{n+1}_m + U^{n+1}_{m-1}]}{h^2} + \mathcal{O}(\frac{k}{h}, h^2, k) \tag{3.48}$$

and shall be referred to as Saul'yev I and Saul'yev II respectively. The principal part of their truncation error is respectively: $(-T_1 + T_2)$ and $(T_1 + T_2)$ where:

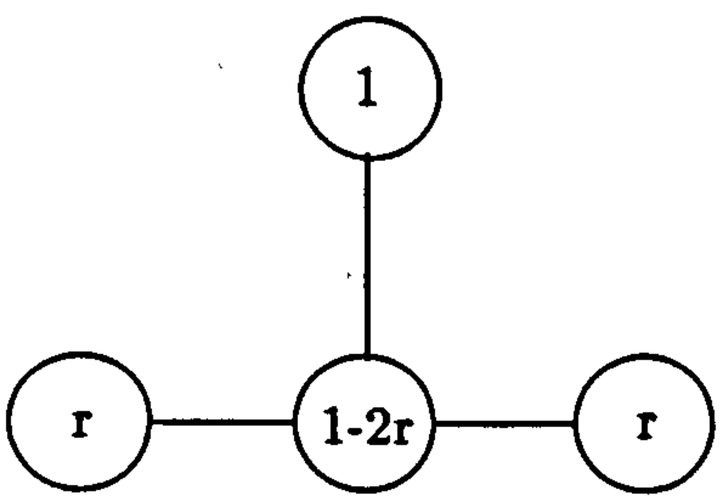
$$T_1 = \left(\frac{k}{h}\frac{\partial^2 \tilde{u}}{\partial t \partial x} + \frac{kh}{6}\frac{\partial^4 \tilde{u}}{\partial x^3 \partial t} + \frac{k^3}{24h}\frac{\partial^4 \tilde{u}}{\partial x \partial t^3}\right)^{n+\frac{1}{2}}_m$$

$$and\ T_2 = \left(-\frac{h^2}{12}\frac{\partial^4 \tilde{u}}{\partial x^4} - \frac{k^2}{8}\frac{\partial^4 \tilde{u}}{\partial x^2 \partial t^2} + \frac{k^2}{24h}\frac{\partial^3 \tilde{u}}{\partial t^3}\right)^{n+\frac{1}{2}}_m$$

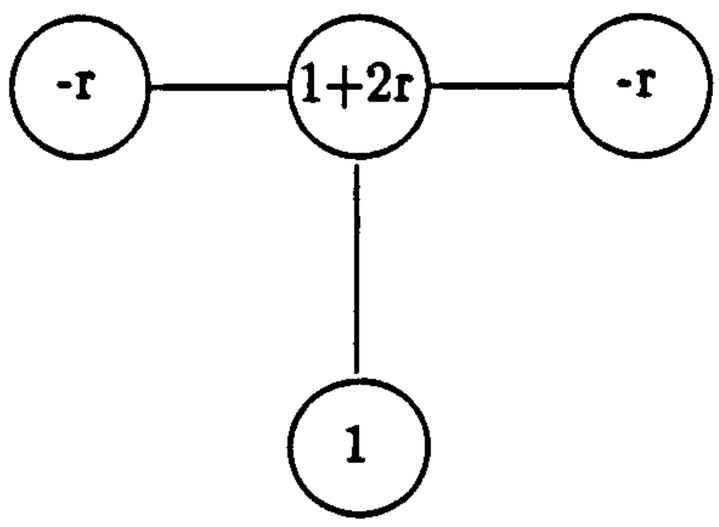
The above difference replacements are only some of the many replacements for (3.39) and shall be referred to while introducing next the concepts of *consistency* and *stability*. A schematic representation of the above schemes is shown in figure 3.3.

3.4 Properties of various finite difference schemes

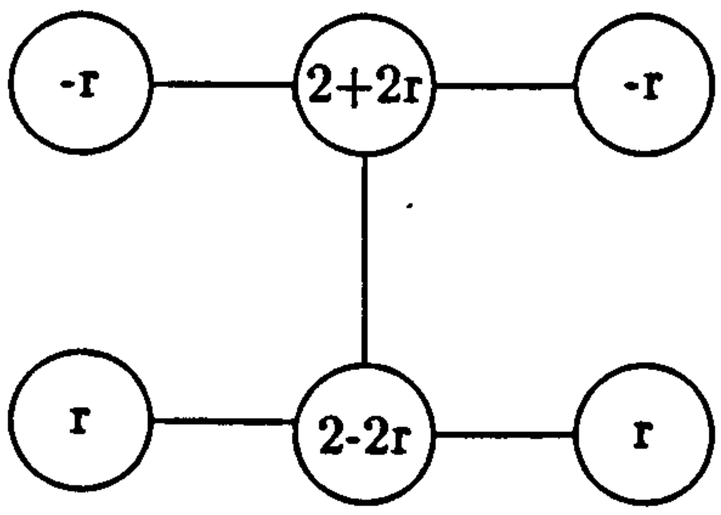
When we resort to finite difference methods to solve differential equations, we need to guarantee that the computational solution (u) is close to the finite difference solution U and converges to the exact solution $\tilde{u}$ of the well posed problem, given by the differential equation and its the auxiliary conditions, as the grid spacings tend to zero.



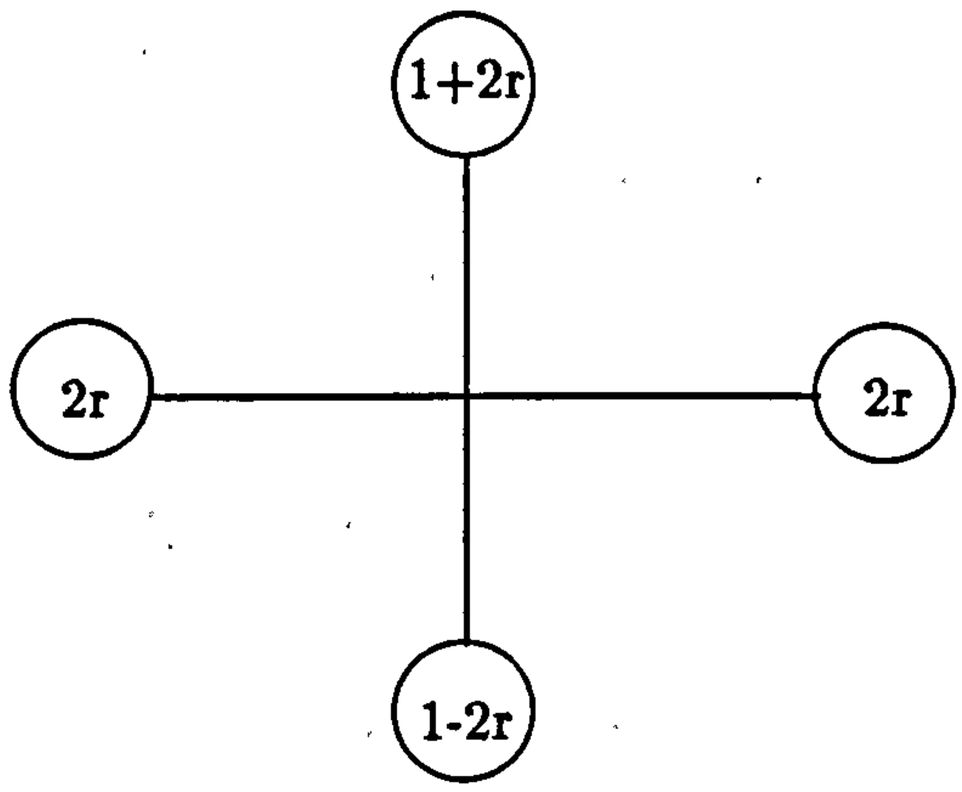
The Explicit scheme



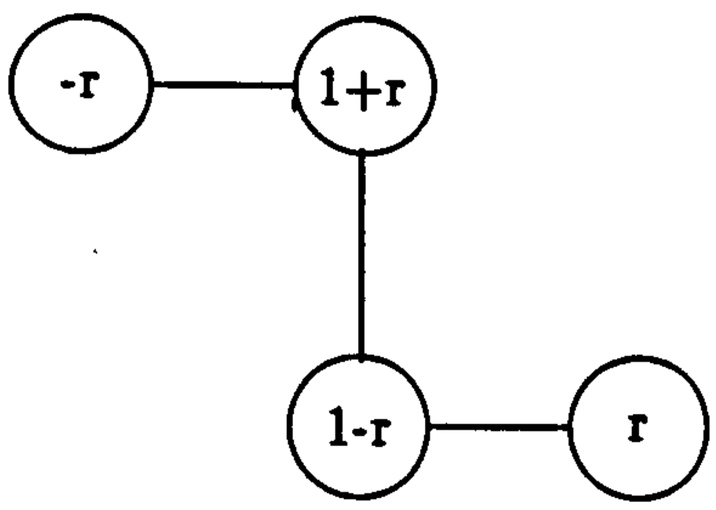
The fully implicit scheme



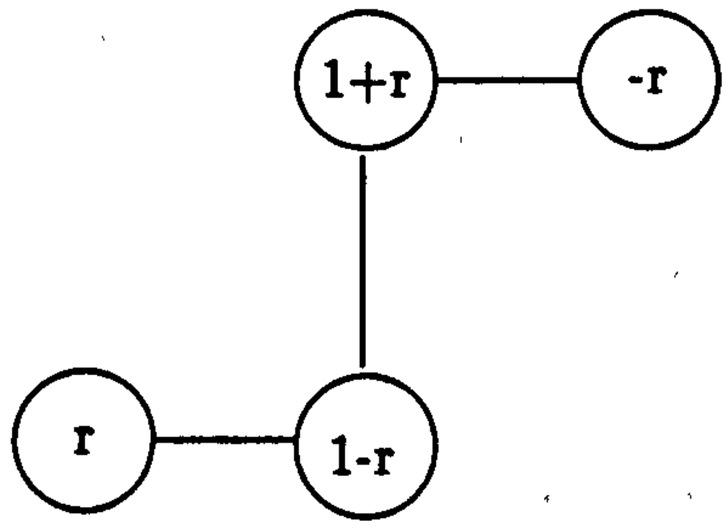
The Crank-Nicholson Scheme



The Dufort- Frankel scheme



The Saul'yev II scheme



The Saul'yev I scheme

Figure 3.3: Diagrams of the computational molecules for various finite difference replacements of equation 3.39

To guarantee this in the case of *linear* differential equations, it is required, firstly to ensure that the difference scheme used is *consistent* with the given differential equation, and secondly that small perturbations in the initial conditions, and errors due to round-off from arithmetic operations performed when solving the difference equations decay, rather than grow, while the computational solution is advanced, i.e, that the solution is *stable.*

3.4.1 Consistency

A difference scheme is said to be consistent with the differential equation, which it approximates, if as the grid is refined (i.e $h \to 0$, $k \to 0$) the local truncation error tends to zero. That is, the finite difference equations become an exact representation of the differential equations.

If we consider the difference formulae given in section 3.3 we see that for each of the Classical Explicit, fully implicit and the Crank-Nicholson schemes, the local truncation error tend to zero unconditionally as $h \to 0$, and $k \to 0$. These formulae are therefore *Unconditionally consistent* with equation (3.39).

As for the Dufort- Frankel scheme and the Saul'yev formulae I and II, we notice that their truncation error terms contain products of the terms $(\frac{k^2}{h^2})$ and $(\frac{k}{h})$ respectively.

This means that if *k and h* tend to zero at the same rate, or h tends to zero faster than k, the truncation error will not tend to vanish, in which case the difference formulae will not be consistent with (3.39). If on the other hand $k \to 0$ at a faster rate than h, the truncation error tend to zero. Therefore, under this condition the Dufort-Frankel and the Saul'yev schemes are consistent. In this case they are said to be *conditionally consistent* with equation (3.39).

If the auxiliary conditions to (3.39) involve derivatives which are also replaced by difference approximations, then for the difference scheme to be consistent with the PDE *problem* it is also required that the truncation error arising from the finite difference approximation to the derivatives in the auxiliary conditions tend to zero

as $h, k \rightarrow 0$.

3.4.2 Stability

The problem of stability is concerned with finding the conditions which satisfy the second requirement, i.e, the requirement that for fixed grid spacings k and h, the difference between the computational solution u and the exact solution of the finite difference equations i.e, $\mid u_m^n - U_m^n \mid$ decays or remain bounded as $n \rightarrow \infty$, where u is the solution calculated up to the accuracy of the machine rather than the exact solution of the difference equations.

The stability of a finite difference scheme can be investigated in a more analytical way through several methods. The most common methods are the Von-Neumann or Fourier series method, the matrix method and the energy method. The first method is suitable for pure initial value problems with periodic initial data thus neglecting the effects of boundary conditions, while the latter two incorporate the effects of the boundary conditions in initial-boundary value problems. The energy method can deal effectively with variable coefficients and indicate the correct choice of computational scheme. But its short comming is that " it provides only sufficient conditions for stability which may be far removed from what is necessary in certain initial- boundary value problems" ([47] p:141). In what follows only illustrations on the theory and applications of the Fourier method and the matrix method are given.

The Fourier method

Consider the equation (3.39) which is discretized over its region of definition $\Re \in (0 \leq x \leq 1)$, $t \geq 0$ using any of the finite difference approximations given in section 3.3. It can easily be shown that the computational error $Z_m^n = u_m^n - U_m^n$ also satisfies the same finite difference equations as U_m^n.

Thus, for the computational error associated with the Classical Explicit scheme, we can write:

$$Z_m^{n+1} = rZ_{m+1}^n + (1-2r)Z_m^n + rZ_{m-1}^n \tag{3.49}$$

In the Fourier method, the errors Z_m^n at the mesh points at a certain time level (say t=0) are represented by a finite Fourier series leading to the error function:

$$Z_m^0 = \sum_{j=0}^{N+1} A_j \exp(i\beta_j mh) \tag{3.50}$$

where $m = (0, 1, 2 \ldots N+1)$, is the index for the (N+2) mesh points dividing the interval $0 \leq x \leq 1$ on the line $t = 0$, β_j is the frequency of the error, and $i = \sqrt{-1}$.

The method can be applied to all linear finite difference approximations. The growth, or decay, of each mode in (3.50) depends on the finite difference equation. If any mode can grow without bound, the difference equations have unstable solutions.

Because of the linearity assumption, separate solutions are additive, and it is sufficient to study the propagation of error due to just a single term $\exp(i\beta mh)$ of the Fourier series in (3.50), where β is any real number.

We therefore seek a solution for (3.49) in the separation-of-variables form, which reduces to $\exp(i\beta mh)$ for $t = 0$. This solution is given by:

$$\exp(\alpha t)\exp(i\beta mh) = \exp(\alpha nk)\exp(i\beta mh)$$

where $\alpha(\beta)$ is complex in general.

Therefore, we can see that the original error component will not grow as $n \rightarrow \infty$ if:

$$|\exp(\alpha k)| \leq 1 \tag{3.51}$$

here we introduce the parameter $\xi = \exp(\alpha k)$, which is known as the *amplification factor* for the m^{th} Fourier mode of the error distribution as it propagates one step forward in time.

To determine this factor for the Classical Explicit scheme, we substitute for Z_m^n by $\exp(\alpha nk)\exp(i\beta mh) \neq 0$ in the "error" equation (3.49).

This gives:

$$\exp(\alpha(n+1)k)\exp(i\beta mh) \;=\; (1-2r)\exp(\alpha nk)\exp(i\beta mh) +$$

$$r\exp(\alpha nk) \quad \times \quad [\exp(i\beta(m+1)h) + \exp(i\beta(m-1)h)]$$

The cancellation of common terms and substituting ξ for $\exp(\alpha k)$ leads to:

$$\xi = (1-2r) + r[\exp(i\beta h) + \exp(-i\beta h)] \tag{3.52}$$

By using the relation: $\exp(i\beta h) + \exp(-i\beta h) = 2\cos(\beta h)$ and the trigonometric relation $[1-\cos(\beta h)] = 2\sin^2(\frac{\beta h}{2})$ equation (3.52) becomes:

$$\xi = 1 - 4r\sin^2(\frac{\beta h}{2})$$

To obtain the stability criteria, the magnitude of the *amplification factor* for all the error modes should be less or equal to one. This is satisfied if:

$$-1 \leq 1 - 4r\sin^2(\frac{\beta h}{2}) \leq +1 \tag{3.53}$$

for all (βh).

The right side of the inequality (3.53) is trivially satisfied. The left side of (3.53) is satisfied if:

$$r \leq \frac{1}{2\sin^2(\frac{\beta k}{2})} \quad for\ all\ (\beta h)$$

Thus a necessary condition for the stability of the explicit scheme is $r \leq \frac{1}{2}$.

Similarly, a stability analysis could be carried out for other schemes, by substituting for $Z_m^n = \exp(\alpha nk)\exp(i\beta mh)$ in their corresponding "error" equations, and determining the condition which is to be imposed on the mesh ratio 'r' to make $|\xi| \leq 1$. The application of the Fourier series method can be extended to investigate the stability of a *system* of finite difference equations.

To illustrate this, the Dufort-Frankel approximation:

$$(1+2r)U_m^{n+1} = (1-2r)U_m^{n-1} + 2r(U_{m+1}^n + U_{m-1}^n) \tag{3.54}$$

to equation (3.39) is rewritten as a system of two equations as:

$$\begin{aligned}(1+2r)U_m^{n+1} &= 2r(U_{m+1}^n + U_{m-1}^n) + (1-2r)V_m^n \\ V_m^{n+1} &= U_m^n \end{aligned} \tag{3.55}$$

By introducing a second variable $V_m^n = U_m^{n-1}$. As before the computational errors $\bar{Z}_m^n = \begin{bmatrix} u_m^n - U_m^n \\ v_m^n - V_m^n \end{bmatrix}$ at the mesh points on the initial time level ($t = 0$) are represented by a Fourier series as:

$$\bar{Z}_m^0 = \sum_{j=0}^{N+1} \bar{A}_j \exp(i\beta_j mh) \tag{3.56}$$

where $\bar{A}_j$ is a (2×1) vector. Again we study the growth or decay of one error mode $\exp(i\beta mh)$ by seeking a solution to the "error" system corresponding to (3.49) in the seperation-of-variables form, which reduces to $\exp(i\beta mh)$ for $t = 0$. Let this solution be:

$$\bar{Z}_m^n = (G)^n \exp(i\beta mh), \tag{3.57}$$

where the time dependence of this error mode is contained in the complex coefficient $(G)^n$. The superscript n implies that G is raised to the power n. G is a (2×2) matrix. Clearly from (3.57):

$$\bar{Z}_m^{n+1} = (G)\bar{Z}_m^n$$

Thus G is the *'amplification'* matrix.

The criterion for stability is:

$$\| G \| \leq 1$$

To find G for the Dufort-Frankel scheme, we write the "error" system corresponding to (3.55), (using the relation $(\bar{Z}_{m+1}^n + \bar{Z}_{m-1}^n) = 2\cos(\beta h)\bar{Z}_m^n$) as:

$$Q_1 \bar{Z}_m^{n+1} = Q_0 \bar{Z}_m^n$$

where $Q_1 = \begin{bmatrix} 1+2r & 0 \\ 0 & 1 \end{bmatrix}$, $Q_0 = \begin{bmatrix} 4r\cos(\beta h) & 1-2r \\ 1 & 0 \end{bmatrix}$.

$$Thus\ G = Q_1^{-1} Q_0 = \frac{1}{1+2r} \begin{bmatrix} 1 & 0 \\ 0 & 1+2r \end{bmatrix} \begin{bmatrix} 4r\cos(\beta h) & 1-2r \\ 1 & 0 \end{bmatrix}$$

$$i.e. \qquad G = \begin{bmatrix} \frac{4r\cos(\beta h)}{1+2r} & \frac{1-2r}{1+2r} \\ 1 & 0 \end{bmatrix}$$

The eigenvalues of G are:

$$\lambda_i = \frac{2r\cos(\beta h) \mp \sqrt{1-4r^2 sin^2 \beta h}}{1+2r} \tag{3.58}$$

$$and\ thus \qquad \rho(G) \leq 1 \tag{3.59}$$

which satisfies the *necessary condition* for stability.

Consideration of (3.58) shows that (3.59) is satisfied for every (βh).

If G is normal (i.e it commutes with its adjoints) then (3.59) is also a sufficient condition ([33] p:173).

The matrix method

The matrix method is applicable to initial boundary value problems, i.e equation (3.39) together with the initial conditions:

$$\tilde{u}(x,0) = g(x)$$

and the Dirichlet boundary conditions:

$$\tilde{u}(0,t) = 0 \quad and \quad \tilde{u}(1,t) = 0$$

After discretization, the number of mesh points at each time level interior to the domain is N. Thus at a certain time level, the set of algebraic equations representing the finite difference approximation at each mesh point can be written in matrix form as:

$$A\mathbf{U}^{n+1} = B\mathbf{U}^{n} + C\mathbf{U}^{n-1} + \bar{\mathbf{d}} \tag{3.60}$$

where A, B, and C are $N \times N$ matrices given below. $\mathbf{U}$ is an $N \times 1$ vector of the dependent variable i.e $\mathbf{U} = [U_1, U_2, \ldots U_N]^T$. The vector $\bar{\mathbf{d}}$ represents the boundary conditions - in the above example $\bar{\mathbf{d}} = \mathbf{0}$. The matrix A is:

$$A = I \quad (i.e\ N \times N\ \ identity\ \ matrix) \quad for\ the\ Explicit\ scheme$$

$$A = \begin{bmatrix} (2+2r) & -r & & & \\ -r & (2+2r) & -r & 0 & \\ & \ddots & \ddots & \ddots & \\ & 0 & \ddots & \ddots & -r \\ & & & -r & (2+2r) \end{bmatrix} \quad \textit{the Crank - Nicholson scheme.}$$

$A = (1+2r)I$ *for the Dufort - Frankel scheme.*

$$A = \begin{bmatrix} (1+2r) & -r & & & \\ -r & (1+2r) & -r & 0 & \\ & \ddots & \ddots & \ddots & \\ & 0 & \ddots & \ddots & -r \\ & & & -r & (1+2r) \end{bmatrix} \quad \textit{the Implicit scheme.}$$

$$A = \begin{bmatrix} (1+r) & -r & & & \\ & (1+r) & -r & 0 & \\ & & \ddots & \ddots & \\ & 0 & & \ddots & -r \\ & & & & (1+r) \end{bmatrix} \quad \textit{the Saul'yev I scheme.}$$

$$A = \begin{bmatrix} (1+r) & & & & \\ -r & (1+r) & & 0 & \\ & \ddots & \ddots & & \\ & 0 & \ddots & \ddots & \\ & & & -r & (1+r) \end{bmatrix} \quad \textit{the Saul'yev II scheme.}$$

The matrix B is given as:

$$B = \begin{bmatrix} (1-2r) & r & & & \\ r & (1-2r) & r & 0 & \\ & \ddots & \ddots & \ddots & \\ & 0 & \ddots & \ddots & r \\ & & & r & (1-2r) \end{bmatrix} \quad \textit{the Explicit scheme} \quad (3.61)$$

$$B = \begin{bmatrix} (2-2r) & r & & & \\ r & (2-2r) & r & 0 & \\ & \ddots & \ddots & \ddots & \\ & 0 & \ddots & \ddots & r \\ & & & r & (2-2r) \end{bmatrix} \quad \textit{the Crank - Nicholson scheme.}$$

$$B = \begin{bmatrix} 0 & 2r & & & \\ 2r & 0 & 2r & 0 & \\ & \ddots & \ddots & \ddots & \\ & 0 & \ddots & \ddots & 2r \\ & & & 2r & 0 \end{bmatrix} \quad \textit{the Dufort - Frankel scheme.}$$

$B = I$ *for the Implicit scheme.*

$$B = \begin{bmatrix} (1-r) & & & & \\ r & (1-r) & & 0 & \\ & \ddots & \ddots & & \\ & 0 & \ddots & \ddots & \\ & & & r & (1-r) \end{bmatrix} \quad \textit{the Saul'yev I scheme.}$$

$$B = \begin{bmatrix} (1-r) & r & & & \\ & (1-r) & r & 0 & \\ & & \ddots & \ddots & \\ & 0 & & \ddots & r \\ & & & & (1-r) \end{bmatrix} \quad \textit{the Saul'yev II scheme.}$$

The matrix C is an $(N \times N)$ zero matrix for all schemes except the Dufort-Frankel, which is a three time level scheme for which:

$$C = (1-2r)I$$

For all the two time level schemes stability can be investigated by writing (3.60) in the explicit form:

$$\mathbf{U}^{n+1} = A^{-1}B\mathbf{U}^n + A^{-1}\bar{\mathbf{d}} \tag{3.62}$$

and examining the eigenvalues of $A^{-1}B$.

The error vector ($\mathbf{z}^n = \mathbf{u}^n - \mathbf{U}^n$) satisfies:

$$\mathbf{z}^{n+1} = G\mathbf{z}^n$$

where $G = A^{-1}B$.

Thus, if $\mathbf{z}^0$ is the $N \times 1$ vector representing the perturbation in the initial conditions then:

$$\mathbf{z}^{n+1} = G^{n+1}\mathbf{z}^0 \tag{3.63}$$

$$and \quad \| \mathbf{z}^{n+1} \| \leq \| G^{n+1} \| \times \| \mathbf{z}^0 \|$$

The following inequality:

$$\rho^{n+1}(G) \leq \| G^{n+1} \| \leq (\| G \|)^{n+1}$$

where $\rho(G)$ is the spectral radius of G, can be used to deduce from (3.63) the following conditions for stability.

i) The spectral radius condition

$$\rho(G) \leq 1 \tag{3.64}$$

is a *necessary* condition for stability, since it guarantees that $G^n \rightarrow 0$, and consequently $\mathbf{z}^n \rightarrow 0$ as $n \rightarrow \infty$, but gives no indication of the magnitude of $\mathbf{z}^n$ for finite n.

ii) The norm condition

$$\| (G) \| \leq 1 \tag{3.65}$$

which is sufficient for stability and guarantees an ever-diminishing error as n increases ([34] p:41).

For the Classical explicit scheme:

$$G \equiv B \qquad (B \textit{ is given by equation } 3.61)$$

$$\textit{and } \| G \| = | 1 - 2r | + 2r \leq 1 \textit{ if } r \leq \frac{1}{2}$$

For the remaining two time level schemes G is given as:

$$G = A^{-1}B$$

Since A and B commute, the eigenvalues of G can be obtained using equation (2.10) and theorem (2.7) as:

$$\lambda_i(G) = \frac{\lambda_i(B)}{\lambda_i(A)} \tag{3.66}$$

Thus for the implicit and Crank-Nicholson schemes we have respectively

$$\lambda_s(G) = \frac{1}{(1+2r) + 2r\cos\frac{s\pi}{N+1}} \Big\} < 1 \quad \forall\, r, s \tag{3.67}$$

$$\textit{and } \lambda_s(G) = \frac{(2-2r) + 2r\cos\frac{s\pi}{N+1}}{(2+2r) + 2r\cos\frac{s\pi}{N+1}} \Big\} < 1 \quad \forall\, r, s \tag{3.68}$$

Since A and B are commutative and symmetric then $G = A^{-1}B$ is also symmetric. Therefore $\rho(G) = \| G \|_2$, and the spectral radius condition satisfied for the implicit and Crank-Nicholson schemes by (3.67) and (3.68) respectively is also a *sufficient* condition for the stability of the two schemes. As for the Saul'yev (I and II) schemes, since A is a upper/lower triangular matrix with diagonal elements (1+r), and B is a lower/upper triangular matrix with diagonal elements (1-r) then for both schemes the eigenvalues of G are $\lambda_s(G) = \frac{1-r}{1+r} \leq 1$ *for every* r, of multiplicity N. As for the Dufort-Frankel scheme the "error" equation corresponding to (3.60) can be written in explicit form as:

$$\mathbf{z}^{n+1} = \frac{1}{1+2r}[B\mathbf{z}^n + (1-2r)I\mathbf{z}^{n-1}] \tag{3.69}$$

To investigate the stability, (3.69) should be rewritten as the following system:

$$\begin{bmatrix} \mathbf{z}^{n+1} \\ \mathbf{z}^{n} \end{bmatrix} = \begin{bmatrix} (1+2r)^{-1}B & (1-2r)(1+2r)^{-1}I \\ I & 0 \end{bmatrix} \begin{bmatrix} \mathbf{z}^{n} \\ \mathbf{z}^{n-1} \end{bmatrix} \tag{3.70}$$

which is of the form:

$$E^{n+1} = WE^{n} \tag{3.71}$$

where $E^{n+1} = \begin{bmatrix} \mathbf{z}^{n+1} \\ \mathbf{z}^{n} \end{bmatrix}$.

Since the matrices $(1+2r)^{-1}B$, I and $(1-2r)(1+2r)^{-1}I$ are all symmetric and commmute with each other, they have a common set of linearly independent eigenvectors. Also, the eigenvalues of W are the eigenvalues of the matrices:

$$\begin{bmatrix} \frac{\lambda_k}{1+2r} & \frac{1-2r}{1+2r} \\ 1 & 0 \end{bmatrix}$$

where λ_k is the k^{th} eigenvalue of B given as:

$$\lambda_k = 2r\cos k\pi/(N+1), \quad k = (1,2,\ldots N).$$

Thus the eigenvalues (v_i) of W can be obtained by solving for:

$$det \begin{bmatrix} \frac{\lambda_k}{1+2r} - v & \frac{1-2r}{1+2r} \\ 1 & -v \end{bmatrix} = 0$$

which gives:

$$v(v - \frac{\lambda_k}{1+2r}) - \frac{1-2r}{1+2r} = 0$$

giving:

$$v(W) = \frac{1}{1+2r}\left\{2r\cos\frac{k\pi}{N+1} \mp \left[1 - 4r^2\sin^2\frac{k\pi}{N+1}\right]^{\frac{1}{2}}\right\} \quad k = (1,\ldots N) \tag{3.72}$$

It can be shown from (3.72) that the magnitudes of all the eigenvalues of W are less than unity for every r. This satisfies the spectral radius condition for stability given in (3.64).

The problem of finding a continuous exact solution for a PDE has now been transformed to finding the solution, at discrete points, given by its finite difference approximation. This involves solving a large system of algebraic equations (one at each point) for the whole mesh. This system may be explicit or implicit, depending on the difference scheme. The solution of the systems of equations arising from explicit finite difference schemes is straight-forward. i.e., matrix vector multiplication. However *implicit* difference schemes require the solution of large systems of algebraic difference equations which have the general form:

$$A\mathbf{u} = \mathbf{b} \tag{3.73}$$

where A is an $N \times N$ matrix, (N being the total number of gridpoints in the region of the solution). U is an $N \times 1$ vector representing the unknown values of the solution at the gridpoints, and **b** is an $N \times 1$ vector of known elements. Solutions at non gridpoints are interpolated from the surrounding gridpoint solutions.

Methods for solving (3.73) fall into two categories, namely *Direct methods* and *iterative methods.* These are the subjects of the following two sections.

3.4.3 Convergence

A finite difference scheme is said to be *convergent* if, at a fixed mesh point, the exact solution of the finite difference replacement and that of the differential equation get uniformly closer as the mesh is refined.

In the case of finite difference replacements for linear initial value problems, stability and consistency of the finite difference scheme guarantee convergence, as stated by the *Lax equivalence* theorem which states (see [39] p:45) "Given a properly posed *linear initial value* problem and a finite difference approximation to it that satisfies the consistency condition, stability is the necessary and sufficient condition for convergence".

For the more difficult initial *boundary value* problems and for nonlinear problems, establishing the convergence is generally very difficult. A theoretical convergence

analysis can be carried out only for simple cases where the differential equation is not very complicated and the difference scheme replacing it is simple. Examples of such theoretical treatment are the convergence analyses given for the classical explicit scheme replacement of the one dimensional heat conduction equation (3.39) in ([35] p:117-119) and the five point explicit scheme replacement of the wave equation (5.11) in ([47] p:146-148).

Convergence in some more difficult cases may be inferred numerically by examining whether the error of the subsequent computational solutions of the difference equations obtained on progressively refined grids is uniformly reduced. An example of such analysis of convergence is found in ([24] P:75-76).

Finally, the Lax equivalence theorem, for other than linear initial value problems, may be interpreted as providing a necessary condition, and not always sufficient, for convergence [24].

In the next two sections we introduce the methods for solving the the various systems of difference equations which has the typical form of (3.73).

3.5 Direct methods:

Direct methods yield an exact solution of (3.73) in the absence of round-off errors, in a finite number of numerical operations.

However, direct methods are not recommended, when the coefficient matrix A in (3.73) is arbitrarily sparse. This is because in these methods, the matrix A is altered during the computation process, and *fill in* by non zero elements may occur in the band of the matrix ([25] p:484), which requires large storage in the computer memory. All direct methods are considered variants of the Gauss elimination method, which is based on augmenting the right hand side vector in (3.73) by A, and performing some appropriate *elementary row* operations on the augmented matrix so that the elements below the diagonal of A are eliminated. The matrix A becomes upper triangular, and the solution is then obtained by a *back-substitution*

process. In the *Gauss-Jordan* method, instead of a *back- substitution* process, the elimination process proceeds to eliminate the elements above the diagonal of A in the augmented matrix as well. Then each row is divided by its diagonal element, thus transforming A to an identity matrix. The solution vector afterwards, is given by the rightmost column of the augmented matrix. In the above elimination methods, care is taken where possible, to avoid division by zero (The elimination method will fail if such division is unavoidable) and to ensure at each elimination step that the diagonal elements of A have the largest absolute value of all the elements below it in its column (this helps in reducing round off errors). This is achieved by appropriate interchanges in the rows of the augmented matrix in a process known as *maximal column pivoting* or *partial pivoting.* The detailed description of this method, including other intermediate procedures involved (e.g scaling), and illustrative examples are given in ([25]).

Other methods known as the LU decomposition methods are variants of the Gaussian elimination method. These are methods based on factoring the matrix A in terms of a lower triangular matrix L and an upper triangular matrix U, thus enabling the system Ax=b to be rewritten as:

$$LU\mathbf{x} = \mathbf{b} \tag{3.74}$$

where a condition is imposed on the diagonal elements in L, U or in both.

The equation (3.74) is factorized to two systems with the aid of an intermediate vector z and solved in the following two subsequent stages:

$$L\mathbf{z} = \mathbf{b} \tag{3.75}$$

followed by;

$$U\mathbf{x} = \mathbf{z} \tag{3.76}$$

where z is obtained in (3.75) by a *forward substitution* and the solution x is obtained from (3.76) by a *back-substitution* process. *Partial pivoting* may also be necessary

for the LU decomposition methods, but it is slightly different than pivoting in the Gauss and Gauss-Jordan methods, and demands extra computations. The way in which pivoting in the LU methods is employed is illustrated in Algorithm 3.5 below.

The most popular LU decomposition methods are, the *Dolittle* method, which imposes a condition that the diagonal elements of L be all unity, the *Crout* method where the diagonal elements of U are all unity, and the *Choleski* method where the condition $l_{ii} = u_{ii}$ is imposed. This last method is applicable only when A is positive definite, and decomposes A into LL^T (*i.e.*, $U = L^T$) making use of Theorem 2.11 thus eliminating the work of finding U.

A general algorithm for the factorization procedures, including partial pivoting, the forward and backward substitution processes for LU decomposition methods is (extracted from Burden's [2]) is given next.

Alogorithm for LU methods:

1. Input the dimension n of A, the entries a_{ij} and b_i in an augmented matrix $(n+1) \times n$ array A.

2. Input the diagonal elements of U or of L.

3. Let p be the smallest integer such that $1 \leq p \leq n$ and

$$|a_{p1}| = \max_{1 \leq j \leq n} |a_{j1}| \qquad (\textit{find the first pivot element}).$$

$$\textit{if } |a_{p1}| = 0, \qquad \textit{then Output ('No unique solution')}. \textbf{ STOP.}$$

4. If $p \neq 1$ then interchange rows p and 1 in (augmented) array A.

5. Select l_{11} and u_{11} satisfying $l_{11}u_{11} = a_{11}$.

6. For $j = 2 \ldots n$ set $U_{1j} = a_{1j}/l_{11}$ (Calculate the Ist row of U)
   & $l_{j1} = a_{j1}/u_{11}$ (Calculate the Ist row of L)

7. For $i = 2 \ldots n$ do steps 8–11

8. Let p be the smallest integer such that $i \leq p \leq n$ and

$$\left| a_{pi} - \sum_{k=1}^{i-1} l_{pk}u_{ki} \right| = \max_{i \leq j \leq n} \left| a_{ji} - \sum l_{jk}u_{kl} \right| \quad (find\ the\ i^{th}\ pivot\ element).$$

If the maximum is zero, **OUTPUT** 'No unique solution'.

9. If $p \neq i$ then interchange rows p and i in A and L.

10. Select l_{ii} and u_{ii} satisfying:

$$l_{ii}u_{ii} = a_{ii} - \sum_{k=1}^{i-1} l_{ik}u_{ki}$$

11. For $j = i+1, \ldots n$ Set:

$$u_{ij} = \frac{1}{l_{ii}} \left[a_{ij} - \sum_{k=1}^{i-1} l_{ik}u_{ki} \right] \quad (Calculate\ the\ i^{th}\ row\ of\ U)$$

$$l_{ji} = \frac{1}{u_{ii}} \left[a_{ji} - \sum_{k=1}^{i-1} l_{jk}u_{kj} \right] \quad (Calculate\ the\ i^{th}\ column\ of\ L)$$

12. Set $Hold = a_{nn} - \sum_{k=1}^{n-1} l_{nk}u_{kn}$;

If Hold=0 **OUTPUT** 'No solution exists' **STOP**

Select u_{nn} and l_{nn} satisfying $l_{nn}u_{nn} = Hold$.

(Steps 13 and 14 are to perform forward and backward substitutions).

13. $Set\ \ z_1 = a_{1,n+1}/l_{11}$; $For\ i = 2\ \ldots n, Set\ \ z_i = \frac{1}{l_{ii}}[a_{i,n+1} - \sum_{j=1}^{i-1} l_{nk}u_{kn}]$

14. $Set\ \ x_n = z_n/u_{nn} \quad i = n-1, \ldots 1 \quad Set\ \ x_i = \frac{1}{u_{ii}}[z_i - \sum_{j=i+1}^{n} u_{ij}x_j]$

OUTPUT ' $(x_1, \ldots\ldots, x_n)$' **STOP.**

The maximum computational cost for the methods mentioned above, is given in terms of the number of multiplication/division operations and addition/subtraction operations involved. For the *Gaussian* elimination and *Dolittle* methods ($\frac{1}{3}N^3 + \frac{1}{2}N^2 - \frac{N}{3}$) multiplications divisions and ($\frac{N^3}{3} + \frac{N^2}{2} - \frac{5}{6}N$) additions or subtractions are involved. *Choleski's* method requires $\frac{N^3}{6} + \frac{3}{2}N^2 - \frac{N}{3}$ multiplications/divisions

and $\frac{N^3}{6} + N^2 - \frac{7}{6}N$ additions/subtractions and N square roots. The number N is the order of the coefficient matrix A. However, A usually exhibits a special pattern which will keep the computational cost far below the maximum figures given above. In general for a banded matrix of bandwidth $2p+1$ we can obtain the number multiplication/division and addition/subtraction operations required by the Gaussian elimination, Crout, and Dolittle methods by a simple exercise. This gives for the total of multiplication/division operations to be:

$$count - mult = \frac{3(N-2)p^2 + (9N-4)p + 3N - 2p^3}{3} \tag{3.77}$$

and the number of addition/subtraction operations required is:

$$count - add = \frac{3(2N-3)p^2 + (12N-5)p - 4p^3}{6} \ . \tag{3.78}$$

The two figures given by (3.77) and (3.78) are thus of order $\mathcal{O}(Np^2)$. Upon substituting for p=N-1 (i.e. A is a full matrix) we retrieve the previous expressions given above. If A is tridiagonal (i.e. p=1), then substituting for p in (3.77) and (3.78) shows that 5n-4 multiplication/divisions operations and 3N-3 addition/subtraction operations are needed to solve the system $A\mathbf{x} = \mathbf{b}$. Finally, when A is block-tridiagonal of order MN, as sometimes is the case when a system of M coupled differential equations, in M unknowns, is discretized using central difference operators, the number of operations involved is of order $\frac{5}{3}NM^3$ ([24] p:189).

It remains to say that although the elimination methods are reliable and efficient, when A has a narrow band, they involve heavy dependencies in computation which makes them unsuitable for parallel computers.

3.6 Iterative methods

These are methods which solve the system (3.73), i.e., $A\mathbf{u} = \mathbf{b}$, by calculating a sequence of approximate solutions $\mathbf{u}$ that converges to $A^{-1}\mathbf{b}$ as k increases. The sequence is continually generated until successive solutions are sufficiently alike.

Iterative methods have the advantage over the direct methods in that round off errors do not accumulate, but rather decay with the number of iterations, and in that they are more economic to use when A exhibits a non easy pattern.

In this section a survey of some *linear, stationary* iterative methods of *first degree* is presented.

An iterative method for solving (3.73) can be constructed generally by decomposing A as:

$$A = N - P \tag{3.79}$$

such that (3.73) can be rewritten as:

$$(N - P)\mathbf{u} = \mathbf{b} \quad or \quad N\mathbf{u} = P\mathbf{u} + \mathbf{b} \tag{3.80}$$

where N is a nonsingular matrix, sometimes referred to as the *splitting matrix.* It is usually chosen to be easily solvable, (e.g diagonal, triangular).

If we add the superscript $(\mathbf{k+1})$ to u on the left hand side of (3.80) and the superscript (k) to u on the right hand side of (3.80) we obtain an iterative method.

Equation (3.80) can be rewritten as:

$$\mathbf{u^{k+1}} = G\mathbf{u^k} + \mathbf{f} \tag{3.81}$$

where $G = N^{-1}P$ is called the iteration matrix. Also the right side vector $\mathbf{f} = N^{-1}\mathbf{b}$.

For most well known methods, A is split into the form $(D - L - U)$ where L,U, and D are matrices including respectively only the elements of A which are strictly below, strictly above, and on, the diagonal of A. Thus $N - P = D - L - U$.

If we take N=D and P=L+U, we obtain the well known Jacobi (J) method given as:

$$D\mathbf{u^{k+1}} = (L + U)\mathbf{u^k} + \mathbf{b} \quad or \quad \mathbf{u^{k+1}} = D^{-1}(L + U)\mathbf{u^k} + D^{-1}\mathbf{b} \tag{3.82}$$

where the iteration matrix $G_J = B = D^{-1}(L + U)$ is known as the Jacobi matrix.

If N=D-L and P=U we obtain the Gauss-Seidel (GS) method given as:

$$D\mathbf{u}^{\mathbf{k+1}} = L\mathbf{u}^{\mathbf{k+1}} + U\mathbf{u}^{\mathbf{k}} + \mathbf{b} \; or \; \mathbf{u}^{\mathbf{k+1}} = (D-L)^{-1}U\mathbf{u}^{\mathbf{k}} + (D-L)^{-1}\mathbf{b} \quad (3.83)$$

with iteration matrix $G = (D-L)^{-1}U$.

The GS method almost always has a better convergence rate than the J method, but the J method has the advantage of being susceptible to significant improvement in its convergence rate by some acceleration techniques such as the *Chebyshev* acceleration and the *Conjugate Gradient* acceleration. This is because the eigenvalues of the iteration matrix of the J method are real. This is not the case for the GS method and its extrapolated version (i.e. the SOR method introduced next).

If a relaxation parameter ω is introduced into the Gauss-Seidel method such that $0 < \omega < 2$, a significant acceleration in the convergence rate of the method occurs, giving rise to a new method which is known as the *Successive Overrelaxation* (S.O.R) method.

This is more easily illustrated by considering the molecular form of equation (3.83) i.e. the equation for the elements of any vector in the sequence of approximate solutions. This is:

$$u_i^{k+1} = \frac{1}{a_{ii}}(b_i - \sum_{j=1}^{i-1} a_{ij}u_j^{k+1} - \sum_{j=i+1}^{n} a_{ij}u_j^k) \quad (3.84)$$

If u_i^k is both added and subtracted to the RHS of (3.84), it can be rewritten as:

$$u_i^{k+1} = u_i^k + \left[\frac{1}{a_{ii}}(b_i - \sum_{j=1}^{i-1} a_{ij}u_j^{k+1} - \sum_{j=i}^{n} a_{ij}u_j^k)\right] \quad (3.85)$$

where the term in the square brackets represents a *residual* or a *correction* term which tends to zero as the method converges. The SOR method is obtained by relaxing the correction term by a factor of ω such that $0 \leq \omega \leq 2$. Thus (3.85) is rewritten as:

$$u_i^{k+1} = u_i^k + \frac{\omega}{a_{ii}}(b_i - \sum_{j=1}^{i-1} a_{ij}u_j^{k+1} - \sum_{j=i}^{n} a_{ij}u_j^k) \quad (3.86)$$

This is equivalent to evaluating u_i^{k+1} as a weighted average of u_i^k and $(u_i^{k+1})_{GS}$,

$$i.e. \quad u_i^{k+1} = \omega u_{iGS}^{k+1} + (1-\omega)u_i^k$$

This can be written in matrix form as:

$$\mathbf{u}^{k+1} = (1-\omega)\mathbf{u}^k + \omega[D^{-1}(L\mathbf{u}^{k+1} + U\mathbf{u}^k + \mathbf{b})] \tag{3.87}$$

or

$$\mathbf{u}^{k+1} = L_\omega \mathbf{u}^k + (D-\omega L)^{-1}\omega\mathbf{b} \tag{3.88}$$

where $L_\omega = (D-\omega L)^{-1}[\omega U + (1-\omega)D]$.

The iteration matrix for the SOR method is $G_{SOR} = L_\omega$. The matrices N and P corresponding to (3.79) are $N = \omega^{-1}D - L$ and $P = U + (\omega^{-1}-1)D$.

For $\omega = 1$ the SOR method reduces to the GS method. The optimal relaxation factor for the SOR method lies always between 1 and 2. The convergence of the method is sensitive to the choice of the acceleration parameter, and an optimum choice ω' of the parameter is given as:

$$\omega' = \frac{2}{1+\sqrt{1-\rho^2(B)}} \tag{3.89}$$

where $\rho(B)$ is the spectral radius of the Jacobi matrix B corresponding to matrix A. This is often difficult to estimate because the spectral radius is not usually known in advance, and often the few first iterations are used to estimate $\rho(B)$ and consequently ω' follows ([29]).

The *Symmetric Successive Overrelaxation* (SSOR) method evaluates the $\mathbf{u}^{k+1}$ iterate in two half iterations representing two sweeps of the SOR iterations in opposite directions. The first sweep is a forward sweep where the values $u_1^{k+\frac{1}{2}}, u_2^{k+\frac{1}{2}}, \ldots, u_n^{k+\frac{1}{2}}$ are from the half iteration are obtained, while the values $u_n^{k+1}, u_{n-1}^{k+1}, \ldots, u_1^{k+1}$ for the complete iteration are obtained after a further backward sweep. The relaxation parameter of the SOR method used in both sweeps is the same. If different acceleration parameters were used for the two sweeps, the method is then referred to as the *unsymmetric SSOR* or (USSOR).

The two sweeps of the SSOR can be written as:

$$\mathbf{u}^{k+\frac{1}{2}} = L_\omega \mathbf{u}^k + (D - \omega L)^{-1}\omega \mathbf{b} \tag{3.90}$$

$$and \quad \mathbf{u}^{k+1} = U_\omega \mathbf{u}^{k+\frac{1}{2}} + (D - \omega U)^{-1}\omega \mathbf{b} \tag{3.91}$$

where $U_\omega = (D - \omega U)^{-1}[(1-\omega)D + \omega L]$.

Upon eliminating $\mathbf{u}^{k+\frac{1}{2}}$ we get:

$$\mathbf{u}^{k+1} = U_\omega L_\omega \mathbf{u}^{k+\frac{1}{2}} + \mathbf{f} \tag{3.92}$$

where,

$$\mathbf{f} = (D - \omega U)^{-1}[(1-\omega)D + \omega L](D - \omega L)^{-1}\omega \mathbf{b} + (D - \omega U)^{-1}\omega \mathbf{b} \tag{3.93}$$

$$= \omega(2-\omega)(D - \omega U)^{-1} D (D - \omega L)^{-1}\mathbf{b} \tag{3.94}$$

The proof of the equality:

$$(D - \omega U)^{-1}[(1-\omega)D + \omega L](D - \omega L)^{-1}\omega \mathbf{b} + (D - \omega U)^{-1}\omega \mathbf{b}$$

$$= \omega(2-\omega)(D - \omega U)^{-1} D (D - \omega L)^{-1}\mathbf{b} \tag{3.95}$$

of the right hand sides of (3.93) and (3.94) is referred to later and is therefore given below:

The LHS of (3.95) $= (D - \omega U)^{-1}\{\,[(1-\omega)D + \omega L](D - \omega L)^{-1} + I\}\omega \mathbf{b}$.

Therefore we only need to prove that:

$$[(1-\omega)D + \omega L](D - \omega L)^{-1} + I = (2-\omega)D(D - \omega L)^{-1} \tag{3.96}$$

$$\begin{aligned}
[(1-\omega)D + \omega L] \qquad & (D - \omega L)^{-1} + I \\
&= [(D - \omega L) - \omega D + 2\omega L](D - \omega L)^{-1} + I \\
&= I - \omega D(D - \omega L)^{-1} + 2\omega L(D - \omega L)^{-1} + I \\
&= (2D - 2\omega L - \omega D + 2\omega L)(D - \omega L)^{-1} \\
&= (2-\omega)D(D - \omega L)^{-1}
\end{aligned}$$

which verifies equation (3.96).

The splitting matrix N for the SSOR method is given as:

$$N = \frac{\omega}{2-\omega}(\frac{D}{\omega} - L)D^{-1}(\frac{D}{\omega} - U) \tag{3.97}$$

The SSOR method converges slower than the SOR method, but has the advantage that it can be accelerated by the techniques mentioned earlier.

There are other first order methods, some of which are very well known such as the *Simultaneous Displacement* method and the (nonstationary) *Richardson* method. These are given respectively by the following two equations:

$$\mathbf{u^{k+1}} = \mathbf{u^k} + \alpha\underbrace{(\mathbf{b} - \mathbf{A}\mathbf{u^k})}_{\mathbf{R^k}} = (\mathbf{I} - \alpha\mathbf{A})\mathbf{u^k} + \alpha\mathbf{b} \tag{3.98}$$

and

$$\mathbf{u^{k+1}} = \mathbf{u^k} + \alpha_k\underbrace{(\mathbf{b} - \mathbf{A}\mathbf{u^k})}_{\mathbf{R^k}} = (\mathbf{I} - \alpha_k\mathbf{A})\mathbf{u^k} + \alpha_k\mathbf{b} \tag{3.99}$$

where a constant factor α or a different choice α_k for every iteration is multiplied by the residual vector $\mathbf{R}^k$ and then added to the vector $\mathbf{u}^k$ of the present iterate.

We shall suffice ourselves with the above typical well known methods, and discuss next the properties of iterative methods with reference to them.

3.7 Consistency of iterative methods

In the analysis of iterative methods (3.81) and their convergence properties, it is important to consider the related linear system:

$$(I - G)\mathbf{u} = \mathbf{f} \tag{3.100}$$

and the relation between its solution and that of the system $A\mathbf{u} = \mathbf{b}$.

A basic requirement of an iterative method as (3.81) is that for a non-singular coefficient matrix A, when a solution is obtained of the system $A\mathbf{u} = \mathbf{b}$, all subsequent

iterates remain the same. This is known as the *Consistency condition.* Another requirement is that if the iterative method converges, it should converge only to the exact solution $\bar{\mathbf{u}} = A^{-1}\mathbf{b}$ of the above system and not to any other solution. This is the condition of *Reciprocal Consistency.* The following theorems give the *necessary and sufficient* conditions for consistency and reciprocal consistency of an iterative method.

Theorem 3.1 *If A is nonsingular, then an iterative method of the form (3.81) is consistent if and only if:*

$$\mathbf{f} = (I - G)A^{-1}\mathbf{b} \tag{3.101}$$

Theorem 3.2 *If (I-G) is nonsingular, then the iterative method (3.81) is reciprocally consistent if and only if:*

$$\mathbf{b} = A(I - G)^{-1}\mathbf{f} \tag{3.102}$$

Theorem 3.3 *If A is nonsingular, then an iterative method (3.81) is completely consistent if and only if it is consistent and (I-G) is nonsingular.*

If an iterative method is both consistent and reciprocally consistent then it is said to be *completely consistent.* Complete consistency implies that the exact solution of the system $A\mathbf{u} = \mathbf{b}$ and the related system are identical. More details, and proofs of the above theorems are given in ([50], p:65-66).

For the above mentioned methods it is easy to verify, when A is nonsingular and have nonzero diagonal elements that:

For the Jacobi method:

$$G = B = D^{-1}(L + U); \quad (I - G) = I - D^{-1}(L + U) = I - D^{-1}(D - A) = D^{-1}A$$

Therefore (I-G) is nonsingular, and since $\mathbf{f} = D^{-1}\mathbf{b}$,

$$\textit{then } \mathbf{f} = (I - G)A^{-1}\mathbf{b} \quad (\textit{i.e., the method is consistent})$$

Then by Theorem 3.3 the method is completely consistent.

Similarly

for the Gauss-Seidel method:

$$G = (D-L)^{-1}U \Rightarrow (I-G) = I-(D-L)^{-1}(D-L-A) = (D-L)^{-1}A.$$

Therefore (I-G) is nonsingular.

Also,

$\mathbf{f} = (D-L)^{-1}\mathbf{b} = (I-G)A^{-1}\mathbf{b}$, (i.e., the GS method is

For the SOR method:

$$G = (D-\omega L)^{-1}[\omega U + (1-\omega)D]$$

$$\Rightarrow (I-G) = (D-\omega L)^{-1}[D-\omega L-\omega U-(1-\omega)D]$$

$$\Rightarrow (I-G) = (D-\omega L)^{-1}[\omega A] = \omega(D-\omega L)^{-1}A \quad \text{(Hence I-G is nonsingular).}$$

Also

$\mathbf{f} = \omega(D-\omega L)^{-1}\mathbf{b} \Rightarrow \mathbf{f} = (I-G)A^{-1}\mathbf{b}$ (i.e., the SOR method is consistent).

For the SSOR method:

$$G = (D-\omega U)^{-1}[(1-\omega)D+\omega L](D-\omega L)^{-1}[(1-\omega)D+\omega U]$$

But from equation (3.95) we have:

$$(D-\omega U)^{-1}[(1-\omega)D+\omega L]\,(D-\omega L)^{-1}$$

$$= (2-\omega)(D-\omega U)^{-1}D(D-\omega L)^{-1} - (D-\omega U)^{-1}$$

Therefore,

$$\begin{aligned} I-G &= I-[(2-\omega)(D-\omega U)^{-1}D(D-\omega L)^{-1} - (D-\omega U)^{-1}][(1-\omega)D+\omega U] \\ &= I-(D-\omega U)^{-1}[(2-\omega)D(D-\omega L)^{-1} - I][D-\omega L-\omega A] \\ &= I-(D-\omega U)^{-1}[(2-\omega)D-\omega(2-\omega)D(D-\omega L)^{-1}A - D+\omega L+\omega A] \\ &= I-(D-\omega U)^{-1}[-\omega(2-\omega)D(D-\omega L)^{-1}A + D-\omega U] \\ &= \omega(2-\omega)(D-\omega U)^{-1}D(D-\omega L)^{-1}A \end{aligned}$$

Therefore (I-G) is nonsingular.

Also from (3.94) we have:

$$\begin{aligned} \mathbf{f} &= \omega(2-\omega)(D-\omega U)^{-1}D(D-\omega L)^{-1}\mathbf{b} \\ \Rightarrow \mathbf{f} &= (I-G)A^{-1}\mathbf{b}. \end{aligned}$$

Hence the SSOR method is completely consistent.

It follows then, also by Theorem 3.3 that the GS, the SOR, and the SSOR methods are completely consistent.

The complete consistency of the simultaneous displacement method and the Richardson method is obvious from equations (3.98) and (3.99) for every $\alpha \neq 0$ and $\alpha_k \neq 0$.

3.8 Convergence of iterative methods

Complete consistency is always assumed for any reasonable iterative method, however, a very important property, namely convergence is not always assumed. In this section, the basic theorems concerning convergence and the rate of convergence are outlined.

Definition 3.1 *An iterative method is said to be* convergent *if, for any initial vector* $\mathbf{u}^0$ *the sequence of vectors* $\mathbf{u}^k$ *converges to the exact solution* $\bar{\mathbf{u}} = A^{-1}\mathbf{b}$, *as* k *increases.*

$$i.e.\ \ if: \quad \lim_{k\to\infty} \mathbf{u}^k = \bar{\mathbf{u}} \tag{3.103}$$

Since the exact solution $\bar{\mathbf{u}} = A^{-1}\mathbf{b}$ also satisfies (3.81) and the related system (3.100) then the error vector defined as:

$$\mathbf{e}^k = \mathbf{u}^k - \bar{\mathbf{u}} \tag{3.104}$$

satisfies the relation:

$$\mathbf{e}^{k+1} = G\mathbf{e}^k = G^2\mathbf{e}^{k-1} = \ldots = G^{k+1}\mathbf{e}^0 \tag{3.105}$$

Therefore it follows that for convergence we require that:

$$\lim_{k\to\infty} \| G^k \| = 0 \tag{3.106}$$

and from the Definition 2.4 and Theorem 2.6 a necessary and sufficient condition for an iterative method (3.81) to be convergent is that:

$$\rho(G) \leq 1 \tag{3.107}$$

We can estimate the rate of convergence of an iterative method (3.81) by using the relation (3.105), from which we can write:

$$|| \mathbf{e}^k ||_p \leq || G^k ||_p \times || \mathbf{e}^0 ||_p \tag{3.108}$$

Thus if we require that the norm of the error vector $\mathbf{e}^0$ be reduced by a factor ϵ, we have to choose k large enough to satisfy:

$$|| G^k ||_p \leq \epsilon \tag{3.109}$$

Then the *average rate of convergence* is defined for k iterations, as:

$$R_k(G) = -\frac{1}{k} \log_e || G^k || \tag{3.110}$$

It is shown in ([50], P86-87) that:

$$\lim_{k \to \infty} (|| G^k ||_2)^{\frac{1}{k}} = \rho(G)$$

From which the asymptotic rate of convergence R(G) is defined as:

$$R(G) = \lim_{k \to \infty} R_k(G) = -\log_e \rho(G) \tag{3.111}$$

Thus for two convergent iterative methods I and II with iteration matrices G' and G'' respectively, the method I is faster than II if it has a larger asymptotic rate of convergence.

$$i.e. \quad if: \; -\log_e \rho(G') > -log_e \rho(G'')$$

$$or \quad if: \; \rho(G') < \rho(G'')$$

Evans has shown in [10] for several iterative methods that the asymptotic rate of convergence is inversely proportional to the condition number κ of the coefficient matrix A of (3.73) which is defined as:

$$\kappa_{(A)} = ||A|| \times ||A^{-1}|| \tag{3.112}$$

which when A is symmetric gives $\kappa_{(A)} = \frac{\max \lambda_A}{\min \lambda_A}$ where $\max \lambda_A$ and $\min \lambda_A$ are the maximum and minimum eigenvalues of A respectively.

Evans has also introduced the principle of *preconditioning* the original system (3.73) by transforming it into a new system $MA\mathbf{u} = M\mathbf{b}$ whose condition number can be minimized ($\kappa_{(MA)} \ll \kappa_{(A)}$), thus maximizing the rate of convergence. Detailed description of preconditioning techniques can be found in [11], [12], [18], and [17] among others.

The iteration procedure (3.81) is pursued until the exact solution $\bar{\mathbf{u}} = A^{-1}\mathbf{b}$ is reached within a certain prescribed tolerance β such that:

$$|| \mathbf{u}^k - \bar{\mathbf{u}} ||_p \leq \beta \tag{3.113}$$

Since usually $\bar{\mathbf{u}}$ is not known in advance, then the criterion for convergence is based on the norm of the residual vector $\mathbf{R}^k$ defined as:

$$|| \mathbf{R}^k || = || A\mathbf{u}^k - \mathbf{b} || \tag{3.114}$$

approaching zero with a prescribed sufficiently small tolerance.

Chapter 4

The ADI and the AGE Methods

4.1 The Alternating Direction Implicit (ADI) methods

It has been earlier demonstrated in chapter 3 that implicit finite difference methods exhibit unlimited stability, while explicit schemes are generally subject to a restricting condition on their mesh ratio, in the interest of numerical stability. This restriction becomes extremely more severe for multidimensional problems. However, for multidimensional problems, implicit central difference methods require the solution of a system of difference equations of the form $A\mathbf{u} = \mathbf{b}$, where the coefficient matrix A is no longer tridiagonal, but rather A is sparse, structurely banded and of large order. The solution of such systems becomes increasingly difficult and demanding in computer time and memory as the dimensions of the problem increase. These difficulties can be considerably alleviated by applying a class of methods, which possesses both the unconditional stability of the implicit methods, and maintains the simplicity of the one dimensional approach to the problem, even when applied to problems of several dimensions. This class is known as the Alternating Direction Implicit (ADI) methods. It may be applied iteratively to some problems (steady state problems) and applied either iteratively or directly in p steps to other (diffusion/convection) problems. In this section, we present some of these methods

as direct methods applied to the diffusion equation in two or more space dimensions. We consider the following two dimensional PDE:

$$\frac{\partial \tilde{u}}{\partial t} = L\tilde{u} \tag{4.1}$$

where L is a (linear) differential operator, and the equation is defined over a region $\Re = [0 \leq x_1, x_2 \leq 1],\ [t \geq 0]$ and subject to the initial condition:

$$\tilde{u}(x_1, x_2, 0) = g(x_1, x_2) \tag{4.2}$$

where $(x_1, x_2) \in \Re$ and the boundary condition:

$$\tilde{u}(x_1, x_2, t) = f(x_1, x_2, t) \quad where\ (x_1, x_2) \in \partial\Re. \tag{4.3}$$

where $\partial\Re$ is the boundary of $\Re$. The region $\Re$ is covered by a rectangular mesh having m_1 and m_2 internal points, in the x_1 and x_2 directions respectively, with the corresponding grid spacings h_1 and h_2 given by $h_1 = 1/(m_1+1)$ and $h_2 = 1/(m_2+1)$. The coordinates of any point P are given as $P(ih_1, jh_2)$ where $0 \leq i \leq m_1 + 1$ and $0 \leq j \leq m_2 + 1$, with the solution at P denoted by $\tilde{u}_{i,j}$.

If $L = D_1^2 + D_2^2$ (where $D_1 = \frac{\partial}{\partial x_1}$ and $D_2 = \frac{\partial}{\partial x_2}$) we get the heat equation in two dimensions given by :

$$\frac{\partial \tilde{u}}{\partial t} = (D_1^2 + D_2^2)\tilde{u} = \frac{\partial^2 \tilde{u}}{\partial x_1^2} + \frac{\partial^2 \tilde{u}}{\partial x_2^2} \tag{4.4}$$

An exact replacement of (4.4) is :

$$\begin{aligned} \tilde{u}_{i,j}^{n+1} &= exp[k(D_1^2 + D_2^2)]\tilde{u}_{i,j} \\ &= exp(k[\theta D_1^2 + (1-\theta)D_1^2 + \theta D_2^2 + (1-\theta)D_2^2])\tilde{u}_{i,j} \end{aligned} \tag{4.5}$$

$$\begin{aligned} \Longleftrightarrow \qquad & exp[k(-\theta D_1^2 - \theta D_2^2)]\tilde{u}_{ij}^{n+1} = \\ & exp[k((1-\theta)D_1^2 + (1-\theta)D_2^2)]\tilde{u}_{ij}^{n} \end{aligned} \tag{4.6}$$

from which, we can derive the following weighted central difference approximation:

$$[1 - \theta r\delta_{x_1}^2 - \theta r\delta_{x_2}^2]U_{ij}^{n+1} = [1 + (1-\theta)r\delta_{x_1}^2 + (1-\theta)r\delta_{x_2}^2]U_{ij}^{n} \tag{4.7}$$

where U_{ij} is now the finite difference solution, r is defined as $r = k/h^2$, k being the time increment, and $h = h_1 = h_2$, n is the index for the time coordinate, and θ is an extrapolation parameter $0 \leq \theta \leq 1$ to be chosen as follows:

If $\theta = 0$ we get the classic Explicit scheme.

If $\theta = 1/2$ we get the Crank Nicholson scheme.

and

If $\theta = 1$ we get the fully Implicit scheme.

Another way of writing (4.6) is :

$$exp[-\theta kD_1^2]exp[-\theta kD_2^2]U_{ij}^{n+1} = exp[\ (1-\theta)kD_1^2]exp[\ (1-\theta)kD_2^2]U_{ij}^{n} \tag{4.8}$$

The truncated expansion of (4.8) yields :

$$(1 - \theta r\delta_{x_1}^2)(1 - \theta r\delta_{x_2}^2)U_{ij}^{n+1} = (1 + (1-\theta)r\delta_{x_1}^2)(1 + (1-\theta)r\delta_{x_2}^2)U_{ij}^{n} \tag{4.9}$$

It can be seen that the different schemes arising from substituting the different values of θ in (4.9) are only perturbations of those schemes resulting from substituting the same θ values in (4.7). i.e for $\theta = 0, \frac{1}{2}, 1$ we get schemes which are perturbations of the Explicit, Crank-Nicholson's and Implicit schemes respectively. If however, the above weighted approximation (4.9) of (4.1) is applied to advance the solution implicitly in only one direction (the x_1 direction) from time $t = n$ to some intermediate[1] time n^*, then consecutively applied to advance the solution implicitly only in the other direction (the x_2 direction) to the time level t=n+1, we obtain an implicit scheme which, at each stage, only requires the solution of a simple tridiagonal system. Such schemes produce only conditionally stable solutions

[1]This intermediate time n^* level does not necessarily correspond to any time level between t=n and t=n+1

at intermediate 'time levels', but produce unconditionally stable solutions after full time steps. Such schemes are referred to as the Alternating Direction Implicit (ADI) methods. The first of such schemes is due to [38]. The two steps of this scheme are given as:

$$\begin{aligned}(1-\frac{1}{2}r\delta_{x_1}^2)U_{i,j}^{n*} &= (1+\frac{1}{2}r\delta_{x_2}^2)U_{i,j}^{n} \\ (1-\frac{1}{2}r\delta_{x_2}^2)U_{i,j}^{n+1} &= (1+\frac{1}{2}r\delta_{x_1}^2)U_{i,j}^{n*}\end{aligned} \tag{4.10}$$

If $U_{i,j}^{n*}$ is eliminated from the above equations we get the composite equation given by (4.9) with $\theta = \frac{1}{2}$.

The above scheme henceforth referred to as the P – R scheme has a truncation error of order $O(k^2, h^2)$. This can easily be verified by applying the Taylors' Series expansion to its composite formula. One setback for the P – R scheme is that it is only conditionally stable when extended to three dimensional problems.
Another scheme due to [7] which exhibits unconditional stability even when applied to a three dimensional problem is given as :

$$\begin{aligned}(1-\frac{1}{2}r\delta_{x_1}^2)U_{i,j}^{n*} &= (1+\frac{1}{2}r\delta_{x_1}^2)U_{i,j}^{n} + r\delta_{x_2}^2 U_{i,j}^{n} \\ (1-\frac{1}{2}r\delta_{x_2}^2)U_{i,j}^{n+1} &= U_{i,j}^{n*} - \frac{1}{2}r\delta_{x_2}^2 U_{i,j}^{n}\end{aligned} \tag{4.11}$$

The elimination of $U_{i,j}^{n*}$ from the above pair of equations leads to the recovery of (4.9), again with $\theta = \frac{1}{2}$. Thus the Douglas scheme is also a perturbation of the Crank-Nicholson scheme, and has also a truncation error of order $O(k^2, h^2)$. Also [8] formulated a scheme which is a perturbation of the fully Implicit scheme in two dimensions, and has truncation error of order $O(k, h^2)$. This is given as :

$$\begin{aligned}(1-r\delta_{x_1}^2)U_{i,j}^{n*} &= (1+r\delta_{x_2}^2)U_{i,j}^{n} \\ (1-r\delta_{x_2}^2)U_{i,j}^{n+1} &= U_{i,j}^{n*} - r\delta_{x_2}^2 U_{i,j}^{n}\end{aligned} \tag{4.12}$$

Upon eliminating $U_{i,j}^{n*}$ from the above pair of equations, we obtain a composite formula for the Douglas and Rachford (D-R) scheme which is given as:

$$(1-r\delta_{x_1}^2)(1-r\delta_{x_2}^2)U_{i,j}^{n+1} = (1+r^2\delta_{x_1}^2\delta_{x_2}^2)U_{i,j}^{n} \tag{4.13}$$

Another ADI method of second order accuracy in both the time and space direction is obtained by splitting (4.9) in a manner suggested by [9]. This split is given as :

$$\begin{aligned}(1-\frac{1}{2}r\delta_{x_1}^2)U_{i,j}^{n*} &= (1+\frac{1}{2}r\delta_{x_1}^2)(1+\frac{1}{2}\delta_{x_2}^2)U_{i,j}^{n} \\ (1-\frac{1}{2}r\delta_{x_2}^2)U_{i,j}^{n+1} &= U_{i,j}^{n*}\end{aligned} \tag{4.14}$$

The above schemes (with the exception of the P – R scheme) can be easily extended to provide unconditionally stable solutions to the heat equations in three dimensions, which is given as:

$$\frac{\partial\tilde{u}}{\partial t} = (D_1^2+D_2^2+D_3^2)\tilde{u} = \frac{\partial^2\tilde{u}}{\partial x_1^2}+\frac{\partial^2\tilde{u}}{\partial x_2^2}+\frac{\partial^2\tilde{u}}{\partial x_3^2} \tag{4.15}$$

defined over the domain given by $\Re_1 = [0 \leq x_1, x_2, x_3 \leq 1], t \geq 0]$, with the initial conditions :

$$\tilde{u}(x_1, x_2, x_3, 0) = g_1(x_1, x_2, x_3) \tag{4.16}$$

where $(x_1, x_2, x_3) \in \Re_1$, and the boundary conditions :

$$\tilde{u}(x_1, x_2, x_3, t) = f_1(x_1, x_2, x_3, t) \tag{4.17}$$

where $(x_1, x_2, x_3, t) \in \partial\Re_1 \times [t \geq 0]$.

If $\Re_1$ is covered with a uniform grid mesh with spacings h_1, h_2, and h_3 in the directions parallel to the axes x_1, x_2, and x_3 respectively, whereby for simplicity we take $h_1 = h_2 = h_3 = h$, and $r = k/h^2$, k being the time increment. The meshpoints indices in the x_1, x_2, and x_3 directions are i, j, and l respectively.

The extensions to the above mentioned ADI schemes are given as follows.

For the Douglas scheme, the split formulae are:

$$\begin{aligned}(1-\frac{1}{2}r\delta_{x_1}^2)U_{i,j,l}^{n*} &= (1+\frac{1}{2}r\delta_{x_1}^2+r\delta_{x_2}^2+r\delta_{x_3}^2)U_{i,j,l}^{n} \\ (1-\frac{1}{2}r\delta_{x_2}^2)U_{i,j,l}^{n**} &= U_{i,j,l}^{n*}-\frac{1}{2}r\delta_{x_2}^2U_{i,j,l}^{n} \\ (1-\frac{1}{2}r\delta_{x_3}^2)U_{i,j,l}^{n+1} &= U_{i,j,l}^{n**}-\frac{1}{2}r\delta_{x_3}^2U_{i,j,l}^{n}\end{aligned} \tag{4.18}$$

which gives upon eliminating the intermediate solutions $U^{n*}_{i,j,l}$ and $U^{n**}_{i,j,l}$ the following composite formula:

$$\begin{aligned}(1-\frac{1}{2}r\delta^2_{x_1}-\frac{1}{2}r\delta^2_{x_2}-\frac{1}{2}r\delta^2_{x_3})U^{n+1}_{i,j,l}\\ +\frac{r^2}{4}(\delta^2_{x_1}\delta^2_{x_2}+\delta^2_{x_1}\delta^2_{x_3}+\delta^2_{x_2}\delta^2_{x_3}-\frac{1}{2}r\delta^2_{x_1}\delta^2_{x_2}\delta^2_{x_3})U^{n+1}_{i,j,l} \;=\\ (1+\frac{1}{2}r\delta^2_{x_1}+\frac{1}{2}r\delta^2_{x_2}+\frac{1}{2}r\delta^2_{x_3})U^{n}_{i,j,l}\\ +\frac{r^2}{4}(\delta^2_{x_1}\delta^2_{x_2}+\delta^2_{x_1}\delta^2_{x_3}+\delta^2_{x_2}\delta^2_{x_3}-\frac{1}{2}r\delta^2_{x_1}\delta^2_{x_2}\delta^2_{x_3})U^{n}_{i,j,l}\end{aligned} \tag{4.19}$$

The scheme is thus similar to the Crank Nicholson scheme in three dimensions with the second group of terms on both sides of (4.19) representing a "perturbation". Equation (4.19) can be written as:

$$\prod_{i=1}^{3}(1-\frac{1}{2}r\delta^2_{x_i})U^{n+1}_{i,j,l}=\prod_{i=1}^{3}(1+\frac{1}{2}r\delta^2_{x_i})U^{n}_{i,j,l}-\frac{1}{4}r^3\prod_{i=1}^{3}\delta^2_{x_i}U^{n}_{i,j,l} \tag{4.20}$$

For the D-R scheme the formulae are:

$$\begin{aligned}(1-r\delta^2_{x_1})U^{n*}_{i,j,l} &= [1+r(\delta^2_{x_2}+\delta^2_{x_3})]U^{n}_{i,j,l}\\ (1-r\delta^2_{x_2})U^{n**}_{i,j,l} &= U^{n*}_{i,j,l}-r\delta^2_{x_2}U^{n}_{i,j,l}\\ (1-r\delta^2_{x_3})U^{n+1}_{i,j,l} &= U^{n**}_{i,j,l}-r\delta^2_{x_3}U^{n}_{i,j,l}\end{aligned} \tag{4.21}$$

which upon eliminating the intermediate solutions give:

$$\prod_{i=1}^{3}(1-r\delta^2_{x_i})U^{n+1}_{i,j,l}=\prod_{i=1}^{3}(1-r\delta^2_{x_i})U^{n}_{i,j,l}+r\prod_{i=1}^{3}\delta^2_{x_i}U^{n}_{i,j,l} \tag{4.22}$$

For the D'yakonov scheme, the split formulae are as follows:

$$\begin{aligned}(1-\frac{1}{2}r\delta^2_{x_1})U^{n*}_{i,j,l} &= (1+\frac{1}{2}r\delta^2_{x_2})(1+\frac{1}{2}r\delta^2_{x_3})U^{n}_{i,j,l}\\ (1-\frac{1}{2}r\delta^2_{x_2})U^{n**}_{i,j,l} &= U^{n*}_{i,j,l}-(1+\frac{1}{2}r\delta^2_{x_3})U^{n}_{i,j,l}\\ (1-\frac{1}{2}r\delta^2_{x_3})U^{n+1}_{i,j,l} &= U^{n**}_{i,j,l}+(1+\frac{1}{2}r\delta^2_{x_3})U^{n}_{i,j,l}\end{aligned} \tag{4.23}$$

which gives after the elimination of $U^{n*}_{i,j,l}$, and $U^{n**}_{i,j,l}$, the equation:

$$\prod_{i=1}^{3}(1-\frac{1}{2}r\delta^2_{x_i})U^{n+1}_{i,j,l}=\prod_{i=1}^{3}(1+\frac{1}{2}r\delta^2_{x_i})U^{n}_{i,j,l} \tag{4.24}$$

If any of the above *direct* ADI methods is used to solve (4.4), or (4.15), where the unknowns of the difference equations are ordered along the lines x_1, i.e the horizontal lines, then along the lines x_2 i.e the vertical lines, (then along the lines x_3 i.e the levels of the mesh, for the three dimensional problem) - which is known as the *Natural Ordering* - the ADI method will lead to solving a tridiagonal or a block tridiagonal system of equations: For the two dimensional problem these systems are given as:

$$H\mathbf{u} = \mathbf{b_1} \tag{4.25}$$

$$V\mathbf{u} = \mathbf{b_2} \tag{4.26}$$

where b_1 and b_2 are known vectors. H and V are square matrices of order m^2, with H being block diagonal, and V block tridiagonal. They are given as:

$H = diag(H_1)$ and $V = diag(V_b, V_c, V_t)$ where $H_1 = diag(a_2, b, a_1)$, $V_t = diag(a_1)$, $V_b = diag(a_2)$; and $V_c = diag(b)$.

H_1, V_b, V_b, and V_c are square matrices of order m. For the D-R scheme: $a_1 = a_2 = r$ and $b = 1+2r$ while for the remaining ADI schemes mentioned above: $a_1 = a_2 = \frac{-r}{2}$, and $b = 1 + r$. It is customary to solve (4.25) and (4.26) using *direct* LU decomposition methods.

The ADI methods as direct difference schemes can be similarly applied to hyperbolic problems. Such applications are presented in treatments in subsequent chapters.

In the following section, the application of the ADI strategy applied as an iterative method is presented.

4.1.1 Intermediate boundary conditions

It is to be noted that for the application of any of the above ADI schemes, it is necessary to determine the intermediate values of the solution at some boundaries.

These values can be approximated from (4.3) and equation (4.17) to be as:

$$\left.\begin{array}{l} U^{n*}_{i,j} = f^{n+1}_{i,j} \text{ the 2 dimensional problem} \\ \left.\begin{array}{l} U^{n*}_{i,j,l} = f^{n+1}_{i,j,l} \\ U^{n**}_{i,j,l} = f^{n+1}_{i,j,l} \end{array}\right\} \text{ the 3 dimensional problem.} \end{array}\right\} \text{ at the appropriate boundaries.}$$

However the above formulae lead to a loss of accuracy in the respective ADI scheme.

The way to conserving accuracy is by deriving these intermediate boundary values from the equations of the respective ADI schemes. Thus for the two dimensional ADI-PR scheme we get:

$$U^{n*}_{i,j} = \frac{1}{2}(1 - \frac{1}{2}r\delta^2_{x_2})U^{n+1}_{i,j} + \frac{1}{2}(1 + \frac{1}{2}r\delta^2_{x_2})U^{n}_{i,j} \tag{4.27}$$

This can be written using (4.3) as:

$$U^{n*}_{i,j} = \frac{1}{2}(1 - \frac{1}{2}r\delta^2_{x_2})f^{n+1}_{i,j} + \frac{1}{2}(1 + \frac{1}{2}r\delta^2_{x_2})f^{n}_{i,j} \qquad \textit{for the } \mathsf{P-R} \textit{ scheme.} \tag{4.28}$$

Similarly the following formulae are derived for other ADI schemes described above:

$$U^{n*}_{i,j} = (1 - \frac{1}{2}r\delta^2_{x_2})f^{n+1}_{i,j} \qquad \textit{for the D'Yakonov scheme} \tag{4.29}$$

$$U^{n*}_{i,j} = (1 - r\delta^2_{x_2})f^{n+1}_{i,j} + r\delta^2_{x_2}f^{n}_{i,j} \qquad \textit{for the } \mathsf{D-R} \textit{ scheme.} \tag{4.30}$$

As for the three dimensional methods the intermediate boundary values as derived from the respective schemes are:

$$\left.\begin{array}{l} U^{n*}_{i,j,l} = f^{n}_{i,j,l} + (1 - r\delta^2_{x_2})(1 - r\delta^2_{x_3})(f^{n+1}_{i,j,l} - f^{n}_{i,j,l}) \\ \textit{and} \\ U^{n**}_{i,j,l} = f^{n}_{i,j,l} + (1 - r\delta^2_{x_3})(f^{n+1}_{i,j,l} - f^{n}_{i,j,l}) \end{array}\right\} \textit{for the } \mathsf{D-R} \textit{ scheme.} \tag{4.31}$$

$$\left.\begin{array}{l} U^{n*}_{i,j,l} = f^{n}_{i,j,l} + (1 - \frac{1}{2}r\delta^2_{x_2})(1 - \frac{1}{2}r\delta^2_{x_3})(f^{n+1}_{i,j,l} - f^{n}_{i,j,l}) \\ \textit{and} \\ U^{n**}_{i,j,l} = f^{n}_{i,j,l} + (1 - \frac{1}{2}r\delta^2_{x_3})(f^{n+1}_{i,j,l} - f^{n}_{i,j,l}) \end{array}\right\} \textit{for the Douglas scheme.} \tag{4.32}$$

$$\left.\begin{aligned} U^{n*}_{i,j,l} &= \tfrac{1}{2}(1-\tfrac{1}{2}r\delta^2_{x_2})(1-\tfrac{1}{2}r\delta^2_{x_3})f^{n+1}_{i,j,l} \\ &\quad +\tfrac{1}{2}(1+\tfrac{1}{2}r\delta^2_{x_2})(1+\tfrac{1}{2}r\delta^2_{x_3})f^{n}_{i,j,l} \\ \textit{and} & \\ U^{n**}_{i,j,l} &= \tfrac{1}{2}(1-r\delta^2_{x_3})f^{n+1}_{i,j,l} - \tfrac{1}{2}(1+\tfrac{1}{2}\delta^2_{x_3})f^{n}_{i,j,l}) \end{aligned}\right\} \textit{for the D'yakonov scheme.} \tag{4.33}$$

4.1.2 Other alternating direction methods

The ADI method is the method which is used as a component in the EAD method to be presented in the next chapter, however it will be also demonstrated that other alternating direction methods can be that component. To mention briefly other alternating direction methods, we consider equation (4.1) again. If this equation has g space dimensions then a *fractional splitting* method applied to its solution involves splitting the operator L into g simpler operators such that:

$$L = \sum_{i=1}^{g} L_i$$

with each operator L_i being operating over a time interval $\frac{k}{g}$. Thus over each interval we have:

$$g\frac{\partial \tilde{u}}{\partial t} = L_i\tilde{u} \tag{4.34}$$

Then the equivalence of equation (4.1) is taken to be the integral of equations (4.34) over the whole interval k. If all the operators L_i involve derivatives in only one space dimension, (i.e if $L_1 = \frac{\partial^2}{\partial x_1^2}$, $L_2 = \frac{\partial^2}{\partial x_2^2}$ ...) then the fractional splitting method is referred to as the *locally one dimensional* (LOD) method.

Further details and examples of the finite difference schemes derived for these methods, as well as the relations between these schemes and the ADI schemes can be found in [33].

4.2 The ADI iterative methods

Consider the heat equations (4.4) and (4.15) with Dirichlet boundary conditions that are independent of time. If we apply any of the above ADI schemes successively until the solution reaches its steady state, then the applied ADI scheme can be taken to represent an iteration procedure which converges when:

$$U_{i,j}^{n+1} = U_{i,j}^{n} = U_{i,j} \qquad for\ all\ i,j \tag{4.35}$$

Thus the ADI *iterative* methods for *elliptic* differential equation are by-products of the above ADI "direct" schemes that were developed for the parabolic equations.

If we substitute the values in (4.35) in any of the corresponding ADI schemes given in the previous section for solving (4.4) and (4.15) we get respectively the following equations:

$$(\delta_{x_1}^2 + \delta_{x_2}^2)U_{i,j} = 0 \tag{4.36}$$

and

$$(\delta_{x_1}^2 + \delta_{x_2}^2 + \delta_{x_3}^2)U_{i,j,l} = 0 \tag{4.37}$$

which are respectively the five point, and seven point difference replacements of the Laplace equation in two and three dimensions which are given by equating to zero the LHSs of equations (4.4) and (4.15) respectively. The quantity 'r' is now an iteration parameter which may be varied from iteration to iteration. The above iterative process can be shown to be convergent for any positive 'r', with an accelerated rate of convergence when 'r' is varied according to an optimal sequence (see [34] p:149-152).

Another way of demonstrating how the ADI strategy can be applied to solve the Laplace equation in two dimensions iteratively is given as follows. Consider the system of equations, resulting from applying the five point difference formula (4.36) to the totality of mesh points (m_2) given as:

$$(H + V)\mathbf{u} = \mathbf{f} \tag{4.38}$$

where now H and V are two constituents of the coefficient matrix, with H arising from the $\delta^2_{x_1} U_{i,j}$ approximation and V from the $\delta^2_{x_2} U_{i,j}$ approximation in equation (4.36). The vector **f** is associated with the terms of $(\delta^2_{x_1} + \delta^2_{x_2})U$ involving boundary values. Now equation (4.38) is equivalent, to each of the following equations, for any matrices D and E.

$$(H + D)\mathbf{u} = \mathbf{f} - (V - D)\mathbf{u} \tag{4.39}$$

$$(V + E)\mathbf{u} = \mathbf{f} - (H - E)\mathbf{u} \tag{4.40}$$

provided that $(H + D)$ and $(V + E)$ are non-singular. The Peaceman-Rachford iterative procedure for solving (4.38) can be derived by consecutively substituting for the D and E matrices a sequence of matrices $D_k = r_1^k I$ and $E_k = r_2^k I$, to calculate the sequence of vectors $\mathbf{u}^{k+1/2}$ and $\mathbf{u}^{k+1}$ by using the following formulae:

$$\begin{aligned} (H + r_1^k I)\mathbf{u}^{k+1/2} &= \mathbf{f} - (V - r_1^k I)\mathbf{u}^k \\ (V + r_2^k I)\mathbf{u}^{k+1} &= \mathbf{f} - (H - r_2^k I)\mathbf{u}^{k+1/2} \end{aligned} \tag{4.41}$$

where r_1^k and r_2^k are two parameter sequences that are chosen so as to maximize the rate of convergence .

If $r_1^k = r_1$ and $r_2^k = r_2$ for all iterations k, then we have the case of the stationary P-R iterative method, with two parameters. If further $r_1 = r_2 = r$, then we have the P-R iterative method with one acceleration parameter. A method due to Douglas and Rachford for solving iteratively equation (4.38) is given by:

$$\begin{aligned} (H + r_1^k I)\mathbf{u}^{k+1/2} &= \mathbf{f} - (V - r_1^k I)\mathbf{u}^k \\ (V + r_1^k I)\mathbf{u}^{k+1} &= V\mathbf{u}^k + r_2^k \mathbf{u}^{k+1/2} \end{aligned} \tag{4.42}$$

The equations (4.41), (4.42) can be given in a more general formula as:

$$\begin{aligned} (H + r_1^k I)\mathbf{u}^{k+1/2} &= \mathbf{f} - (V - r_1^k I)\mathbf{u}^k \\ (V + r_1^k I)\mathbf{u}^{k+1} &= (V - (1-\omega) r_2^k I)\mathbf{u}^k + (2-\omega) r_2^k \mathbf{u}^{k+1/2} \end{aligned} \tag{4.43}$$

where ω is a parameter that takes the values 0 and 1, for the $\mathsf{P-R}$ and $\mathsf{D-R}$ methods respectively. In this thesis we shall be concerned only with the stationary

ADI iterative methods, and in particular with the ADI iterative methods with a single parameter.

The ADI technique can also be applied as an *iterative method* for solving systems of equations which arise from applying conventional difference schemes (e.g Crank Nicholsòn, fully implicit $\cdots$ etc.) to time dependent problems. This is illustrated by considering the Crank Nicholson difference approximation to (4.4) given as:

$$(1 - \frac{1}{2}r\delta_{x_1}^2 - \frac{1}{2}r\delta_{x_2}^2)U_{i,j}^{n+1} = (1 + \frac{1}{2}r\delta_{x_1}^2 + \frac{1}{2}r\delta_{x_2}^2)U_{i,j}^{n} \tag{4.44}$$

when equation (4.44) is applied to the totality of internal points of the square grid, we get the structured system of equations of the form:

$$A_1\mathbf{u}^{k+1} = \mathbf{b_1} \tag{4.45}$$

where A_1 is a block tridiagonal matrix given as: $A_1 = diag(SS, DD, SS)$, with $DD = diag(\frac{-r}{2}, 1 + 2r, \frac{-r}{2})$ and $SS = diag(\frac{-r}{2})$. $\mathbf{b}$ is a known vector derived from the values associated with the values of the solution at the explicit time level as well as the values at the boundaries. An ADI iterative method for solving this system can be obtained by splitting A_1 to $H_1 + V_1$ such that H_1 arises from the term $(\frac{1}{2} - \frac{1}{2}r\delta_{x_1}^2)$ and V_1 from the term $(\frac{1}{2} - \frac{1}{2}r\delta_{x_2}^2)$ i.e we split A_1 into H_1 and V_1 such that we include in H_1 the coefficients associated with the central difference operator in the x_1 direction, while the coefficients associated with the central difference operator in the x_2 direction are included in V_1. Then the system (4.45) is solved as a two step iterative procedure, the first represents a sweep through the components of the solution vector $\mathbf{u}$ along the horizontal lines of the mesh, and is equivalent to applying an iterative method having a matrix splitting equal to: $N = (H_1 + r_1^k I)$, and the second step represent another sweep through the components of the solution vector along the vertical lines of the mesh and is equivalent to applying an iterative method with the matrix splitting equal to: $N = (V_1 + r_2^k I)$. Each of these two sweeps constitute a half iteration in each single ADI iteration. The first sweep will generate new values $\mathbf{u_j}$ (where $\mathbf{u_j} = u_{1,j}, u_{2,j}, \ldots u_{m,j}$) for $j = 1, 2 \ldots m$, which

we shall denote $\mathbf{u}^{k+1/2}$ and the second sweep will generate new values $\mathbf{u_i}$ (where $\mathbf{u_i} = u_{i,1}, u_{i,2}, \ldots u_{i,m}$) for $i = 1, 2 \ldots m$, which we shall denote $\mathbf{u}^{k+1}$. The formulae of these two sweeps are given as:

$$\begin{aligned} (H_1 + r_1^k I)\mathbf{u}^{k+1/2} &= \mathbf{b} - (V_1 - r_1^k I)\mathbf{u}^k \\ (V_1 + r_2^k I)\mathbf{u}^{k+1} &= \mathbf{b} - (H_1 - r_2^k I)\mathbf{u}^{k+1/2} \end{aligned} \tag{4.46}$$

The above presented ADI iterative method can be implemented to solve systems arising from the application of various finite difference schemes to other time dependent problems, (Parabolic or Hyperbolic) with the convergence properties being dependent on the form of the coefficient matrix, which is dictated by the particular problem. The different convergence properties of the ADI method will be discussed in further chapters when these different problems are discussed.

4.3 Consistency of the ADI method

The composite form of an ADI iterative method for solving (4.45) is obtained by eliminating $\mathbf{u}^{k+1/2}$ from the pair of equations (4.46) and is given as:

$$\mathbf{u}^{k+1} = G_{ADI}\mathbf{u}^k + \mathbf{k} \tag{4.47}$$

where G_{ADI} is the iteration matrix for the ADI method, given as:

$$G_{ADI} = (V_1 + r_2 I)^{-1}(H_1 - r_2 I)(H_1 + r_1 I)^{-1}(V_1 - r_1 I) \tag{4.48}$$

and

$$\begin{aligned} \mathbf{k} &= (V_1 + r_2 I)^{-1}[I - (H_1 - r_2 I)(H_1 + r_1 I)^{-1}]\mathbf{b} \\ &= (V_1 + r_2 I)^{-1}[I - [(H_1 + r_1 I) - (r_1 + r_2)I](H_1 + r_1 I)^{-1}]\mathbf{b} \qquad (4.49) \\ &= (r_1 + r_2)(V_1 + r_2 I)^{-1}(H_1 + r_1 I)^{-1}\mathbf{b} \qquad (4.50) \end{aligned}$$

Lemma 4.1 *The ADI iterative method as defined in equations 4.46 is completely consistent.*

Proof: The iteration matrix of (4.47) given by (4.48) can be rewritten in the form:

$$\begin{aligned}
G_{ADI} &= (V_1 + r_2 I)^{-1}[H_1 + r_1 I - (r_1 + r_2)I](H_1 + r_1 I)^{-1}(A_1 - H_1 - r_1 I) \\
&= (V_1 + r_2 I)^{-1}[I - (r_1 + r_2)(H_1 + r_1 I)^{-1}](A_1 - H_1 - r_1 I) \\
&= (V_1 + r_2 I)^{-1}[A_1 - H_1 - r_1 I - (r_1 + r_2)A_1 + (r_1 + r_2)I] \\
&= (V_1 + r_2 I)^{-1}[V_1 + r_2 I - (r_1 + r_2)I \\
&\quad -(r_1 + r_2)(H_1 + r_1 I)^{-1}A + (r_1 + r_2)I] \\
&= (V_1 + r_2 I)^{-1}[(V_1 + r_2 I) - (r_1 + r_2)(H_1 + r_1 I)^{-1}A] \\
&= I - (r_1 + r_2)(V_1 + r_2 I)^{-1}(H_1 + r_1 I)^{-1}A \\
I - G_{ADI} &= -(r_1 + r_2)(V_1 + r_2 I)^{-1}(H_1 + r_1 I)^{-1}A \qquad (4.51)
\end{aligned}$$

Therefore $(I - G_{ADI})$ is equal to the product of nonsingular matrices and is thus nonsingular. Also:

$$(I - G_{ADI})A^{-1}\mathbf{b} = -(r_1 + r_2)(V_1 + r_2 I)^{-1}(H_1 + r_1 I)^{-1}\mathbf{b} = \mathbf{k}$$

Hence the proof follows from theorems 3.1 and 3.3.

4.4 The Alternating Group Explicit (AGE) Iterative method

4.4.1 A historical review

The Alternating Group Explicit (AGE) iterative method was first introduced in [14] for the solution of Parabolic differential equations in one space dimension. This method is based on applying a splitting strategy alternately, at each half (intermediate) iteration , to the tridiagonal systems of stable implicit difference schemes. The method therefore is not a difference scheme itself, but rather an iterative procedure

for solving the tridiagonal systems of difference equations which are traditionally solved directly by using elimination methods. Therefore its analysis is analogous to that of the ADI *iterative* method described earlier. The convergence analysis of the method, when applied to the one dimensional parabolic heat equation is given by [41], together with several numerical experiments. Sahimi proved theoretically that the method is convergent when applied to the symmetric difference systems of equations arising from the one dimensional parabolic problem, with Dirichlet boundary conditions. Sahimi also carried out several experiments of the method, including its application to hyperbolic problems. His experiments with the hyperbolic problems which produce unsymmetric difference systems of equations showed that the method is also convergent in this case, but his attempts to justify the convergence theoretically was unsuccessful and he concluded that the failure of his attempt "does not necessarily imply the non-convergence of the AGE iterative process but only confirms the theoretical difficulty that arises when dealing with unsymmetric matrices". The AGE method was later extended to systems of difference equations arising from the two and three dimensional heat equations (see [20] and [21]). Also another variant of the method was given in [19] for the one dimensional parabolic problem. Another work by [16] included the reorganization of the the AGE procedure to, which resulted in a saving of about 25% of the computational work involved, at the expense of one extra vector storage. The authors also exposed the suitability of the method for parallel computers and showed that it exhibits an almost linear *speed up*.

In the following subsections the method is presented for the one dimensional heat problem together with its convergence analysis as given in [22].

In the remaining sections of this chapter the AGE method is presented for two dimensional problems. The added developments and theoretical justifications for the method are given in the next chapter.

4.4.2 The Alternating Group Explicit (AGE) method

Consider the one dimensional heat problem given by:

$$\frac{\partial \tilde{u}}{\partial t} = \frac{\partial^2 \tilde{u}}{\partial x^2} \qquad 0 \leq x \leq 1 \quad 0 \leq t \leq T \tag{4.52}$$

subject to the initial conditions: $\tilde{u}(x,0) = f(x) \quad 0 \leq x \leq 1$ and the Dirichlet boundary conditions given by:

$$\tilde{u}(0,t) = g(t); \quad \tilde{u}(1,t) = h(t). \quad 0 < t \leq T$$

We discretize the above problem on a uniformly spaced mesh whose points are defined by the coordinates $x_i = ih$, and $t = jk$, with i=0,1, ...m,m+1 (where, without loss of generality, m is assumed to be odd) and j=0,1,2 ...,n where $h = \frac{1}{m+1}$ and $k = \frac{T}{n+1}$. The mesh ratio $r = \frac{k}{h^2}$. A weighted difference approximation to the above problem will produce a symmetric system of equations which can be written in matrix form as:

$$\tilde{A}\mathbf{u}^{j+1} = \tilde{\mathbf{f}} \tag{4.53}$$

where the matrix $\tilde{A}$ has the form:

$$\tilde{A} = \begin{pmatrix} \tilde{a} & \tilde{b} & & & 0 \\ \tilde{c} & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \tilde{b} \\ 0 & & & \tilde{c} & \tilde{a} \end{pmatrix}$$

with $\tilde{a} = (1 + 2r\theta); \;\; \tilde{b} = \tilde{c} = -r\theta$

where θ is a weighting parameter which is equal to 0,1, or 0.5 depending respectively on whether the explicit, implicit, or the Crank Nicholson scheme is applied. The vector $\tilde{\mathbf{f}}$ is a known vector of order m, consisting of boundary values as well as the values at the time level j. $\mathbf{u}^{j+1}$ is the required solution vector at time level j+1.
If the system (4.53) is divided by $r\theta$ ($\theta \neq 0$) then we get the following equivalent system:

$$A\mathbf{u}^{j+1} = \mathbf{f} \tag{4.54}$$

where $\mathbf{f} = \frac{1}{r\theta}\tilde{\mathbf{f}}$ and $A = diag(c, a, b)$ with $b = c = -1$ and $a = \frac{\tilde{a}}{r\theta}$.

The AGE iterative method for solving (4.54) consists of splitting the coefficient matrix as:

$$A = G_1 + G_2 \tag{4.55}$$

where G_1 and G_2 are given as:

$$G_1 = \begin{pmatrix} a/2 & & & 0 \\ & C & & \\ & & \ddots & \\ 0 & & & C \end{pmatrix}_{m\times m} ; G_2 = \begin{pmatrix} C & & & 0 \\ & \ddots & & \\ & & \ddots & \\ 0 & & & a/2 \end{pmatrix}_{m\times m} \tag{4.56}$$

where

$$C = \begin{bmatrix} a/2 & b \\ c & a/2 \end{bmatrix}_{2\times 2}$$

where it is assumed that the following conditions are satisfied:

1. $(G_1 + sI)$ and $(G_2 + sI)$ are nonsingular for any $s > 0$

2. For any vectors $\mathbf{f_1}$ and $\mathbf{f_2}$ the systems:

   $(G_1 + sI)\mathbf{u_1} = \mathbf{f_1}$

   and

   $(G_2 + sI)\mathbf{u_2} = \mathbf{f_2}$

   are more easily solved in explicit form, since they consist of simple (2×2) subsystems which can be inverted by inspection.

Equation (4.54) can thus be written as:

$$(G_1 + G_2)\mathbf{u} = f \tag{4.57}$$

The stationary AGE method with a single parameter consists of writing (4.57) as a pair of equations given as:

$$(G_1 + sI)\mathbf{u}^{p+1/2} = (sI - G_2)\mathbf{u}^p + \mathbf{f} \tag{4.58}$$

and

$$(G_2 + sI)\mathbf{u}^{p+1} = (sI - G_1)\mathbf{u}^{p+1/2} + \mathbf{f} \quad , \quad p \geq 0 \tag{4.59}$$

where $\mathbf{u}^0$ is an initial approximation of $\mathbf{u}^{j+1}$, (taken as the solution vector $\mathbf{u}^j$ at the j^{th} time level), and s is an acceleration parameter which is chosen to maximize the rate of convergence of the method. After the elimination of the intermediate values $\mathbf{u}^{p+1/2}$, the method can be expressed, as a single equation in the form,

$$\mathbf{u}^{p+1} = G\mathbf{u}^p + \mathbf{k} \tag{4.60}$$

where $\mathbf{k}$ is a known vector given as:

$$\begin{aligned} \mathbf{k} &= (G_2 + sI)^{-1}[(sI - G_1)(G_1 + sI)^{-1} + I]\mathbf{f} \\ &= 2s(G_2 + sI)^{-1}(G_1 + sI)^{-1}\mathbf{f} \end{aligned} \tag{4.61}$$

and G is the iteration matrix of the method given as:

$$G = (G_2 + sI)^{-1}(G_1 - sI)(G_1 + sI)^{-1}(G_2 - sI) \tag{4.62}$$

The equations that apply at each mesh point to compute the solution of (4.54) by the AGE method with a fixed parameter can thus be derived from (4.58) and (4.59) as follows: Let

$$\tilde{G} = \begin{bmatrix} r_2 & b \\ c & r_2 \end{bmatrix} \quad and \quad \tilde{G}_m = \begin{bmatrix} r_1 & -b \\ -c & r_1 \end{bmatrix}$$

where $r_2 = s + \frac{a}{2}$ and $r_1 = s - \frac{a}{2}$

then we can write:

$$(G_1 + sI) = \begin{pmatrix} r_2 & & & 0 \\ & \tilde{G} & & \\ & & \ddots & \\ 0 & & & \tilde{G} \end{pmatrix} ; \quad (G_2 + sI) = \begin{pmatrix} \tilde{G} & & & 0 \\ & \ddots & & \\ & & \ddots & \\ 0 & & & r_2 \end{pmatrix}$$

Since also $(G_1 + sI)^{-1}$ and $(G_2 +_s I)^{-1}$ are block diagonal matrices with square blocks being at most of order 2. Then they can be easily determined by inverting the block diagonal entries of $(G_1 + sI)$ and $(G_2 +_s I)$. Hence we can write:

$$(G_1 + sI)^{-1} = \begin{pmatrix} \frac{1}{r_2} & & & 0 \\ & \tilde{G}^{-1} & & \\ & & \ddots & \\ & & & \ddots \\ 0 & & & \tilde{G}^{-1} \end{pmatrix}_{m \times m}$$

and

$$(G_2 + sI)^{-1} = \begin{pmatrix} \tilde{G}^{-1} & & & 0 \\ & \ddots & & \\ & & \ddots & \\ & & \tilde{G}^{-1} & \\ 0 & & & \frac{1}{r_2} \end{pmatrix}_{m \times m}$$

where

$$\tilde{G}^{-1} = \frac{1}{\Delta} \begin{bmatrix} r_2 & -b \\ -c & r_2 \end{bmatrix} \; ; \quad and \;\; \Delta = r_2^2 - bc \tag{4.63}$$

Similarly $(sI - G_1)$ and $(sI - G_2)$ are given as:

$$(sI - G_1) = \begin{pmatrix} r_1 & & & 0 \\ & \tilde{G}_m & & \\ & & \ddots & \\ & & & \ddots \\ 0 & & & \tilde{G}_m \end{pmatrix} ;$$

and

$$(sI - G_2) = \begin{pmatrix} \tilde{G}_m & & & 0 \\ & \ddots & & \\ & & \ddots & \\ & & \tilde{G}_m & \\ 0 & & & r_1 \end{pmatrix}$$

Now equations (4.58) and (4.59) can be expressed explicitly as:

$$\mathbf{u}^{p+1/2} = (G_1 + sI)^{-1}[(sI - G_2)\mathbf{u}^p + \mathbf{f}] \tag{4.64}$$

and

$$\mathbf{u}^{p+1} = (G_2 + sI)^{-1}[(sI - G_1)\mathbf{u}^{p+1/2} + \mathbf{f}] \qquad , p \geq 0 \tag{4.65}$$

or:

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ \vdots \\ u_{m-2} \\ u_{m-1} \\ u_m \end{bmatrix}^{p+\frac{1}{2}} = \frac{1}{\Delta} \begin{bmatrix} \frac{\Delta}{r_2} & & & & & & \\ & r_2 & -b & & & & 0 \\ & -c & r_2 & & & & \\ & & & \ddots & & & \\ & & & & \ddots & & \\ & & & & & \ddots & \\ & 0 & & & & r_2 & -b \\ & & & & & -c & r_2 \end{bmatrix} \times$$

$$\left(\begin{bmatrix} r_1 & -b & & & & & \\ -c & r_1 & & & & & 0 \\ & & \ddots & & & & \\ & & & \ddots & & & \\ & & & & \ddots & & \\ & & & & & r_1 & -b \\ & 0 & & & & -c & r_1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ \vdots \\ u_{m-2} \\ u_{m-1} \\ u_m \end{bmatrix}^p + \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ \vdots \\ f_{m-2} \\ f_{m-1} \\ f_m \end{bmatrix} \right) \tag{4.66}$$

and

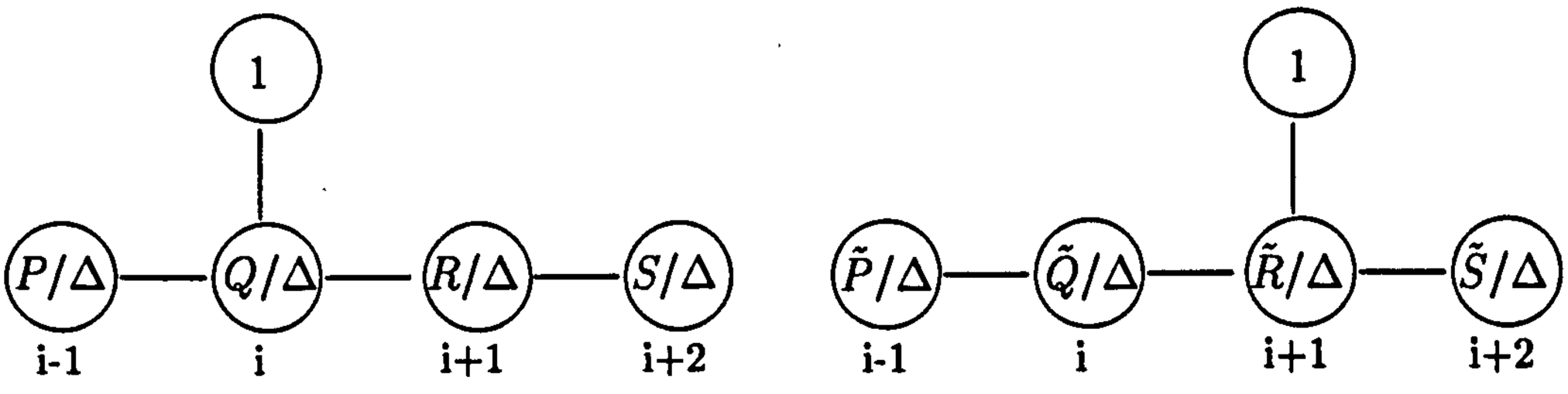


Figure 4.1: The computational molecules for the AGE-1D method at each pair of grouped points.

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ \vdots \\ u_{m-2} \\ u_{m-1} \\ u_m \end{bmatrix}^{p+1} = \frac{1}{\Delta} \begin{bmatrix} r_2 & -b & & & & & \\ -c & r_2 & & & & 0 & \\ & \ddots & \ddots & & & & \\ & & \ddots & \ddots & & & \\ & & & & r_2 & -b & \\ & 0 & & & -c & r_2 & \\ & & & & & & \frac{\Delta}{r_2} \end{bmatrix} \times$$

$$\left(\begin{bmatrix} r_1 & & & & & \\ & r_1 & -b & & & \\ & -c & r_1 & & 0 & \\ & & \ddots & \ddots & & \\ & & & \ddots & & \\ & 0 & & & r_1 & -b \\ & & & & -c & r_1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ \vdots \\ u_{m-2} \\ u_{m-1} \\ u_m \end{bmatrix}^{p} + \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ \vdots \\ f_{m-2} \\ f_{m-1} \\ f_m \end{bmatrix} \right) \tag{4.67}$$

Equations (4.66) and (4.67) show that the solutions $\mathbf{u}^{p+1/2}$ and $\mathbf{u}^{p+1}$ can be obtained by applying at each consecutive pair of mesh points x_i and x_{i+1} the corresponding two computational molecules of figure 4.1, representing the following two formulae:

$$u_i^q = (Pu_{i-1}^v + Qu_i^v + Ru_{i+1}^v + Su_{i+2}^v + T_i)/\Delta \tag{4.68}$$

$$u_{i+1}^{q} = (\tilde{P}u_{i-1}^{v} + \tilde{Q}u_{i}^{v} + \tilde{R}u_{i+1}^{v} + \tilde{S}_{i+2}^{v} + \tilde{T}_{i})/\Delta \tag{4.69}$$

At the half (intermediate) iteration level (i.e for $q = p + 1/2$ and v=p) the above equations are applied at the mesh points x_i (for $i = 2, 4, \cdots, m-1$) and the coefficients are given as:

$$P = -cr_2;\ \ Q = r_1 r_2;\ \ R = -br_1;\ \ T_i = r_2 f_i - b f_{i+1};\ \ and\ \ S = \begin{cases} 0 & \text{for i=m-1} \\ b^2 & otherwise. \end{cases}$$

$$\tilde{P} = -cr_2;\ \ \tilde{Q} = r_1 r_2;\ \ \tilde{R} = -br_1;\ \ \tilde{T}_i = r_2 f_i - b f_{i+1};\ \ and\ \ \tilde{S} = \begin{cases} 0 & \text{for i=m-1} \\ -br_2 & otherwise. \end{cases}$$

For the meshpoint x_1 the following computational formula is used:

$$\mathrm{u}_1^{p+1/2} = (r_1 \mathrm{u}_1^{p} - b\mathrm{u}_2^{p} + f_1)/r_2 \tag{4.70}$$

At the complete iteration level (i.e $q = p + 1$ and $v = p + \frac{1}{2}$), the equations (4.68) and (4.69) are applied at the mesh points x_i ($i = 1, 3, \cdots, m-2$) where the coefficients are now given by:

$$P = \begin{cases} 0 & for\ i = 1 \\ -cr_2 & for\ i \neq 1 \end{cases};\ \ Q = r_1 r_2\ ;\ R = -br_1\ \ S = b^2\ ; and\ T_i = -cf_i + r_2 f_{i+1}$$

$$\tilde{P} = \begin{cases} 0 & for\ i = 1 \\ c^2 & for\ i \neq 1 \end{cases};\ \ \tilde{Q} = cr_1\ ;\ \tilde{R} = r_1 r_2\ \ \tilde{S} = br_2\ ; and\ \tilde{T}_i = -cf_i + r_2 f_{i+1}$$

For the single meshpoint x_m, the following computational formula is used:

$$\mathrm{u}_m^{p+1} = (-c\mathrm{u}_{m-1}^{p+1/2} + r_1 \mathrm{u}_m^{p+1/2} + f_m)/r_2 \tag{4.71}$$

Thus at each level ($q = p + \frac{1}{2}, p$) the mesh points are organized in groups of two, at which the solution can be obtained by the appropriate explicit formulae (4.68)

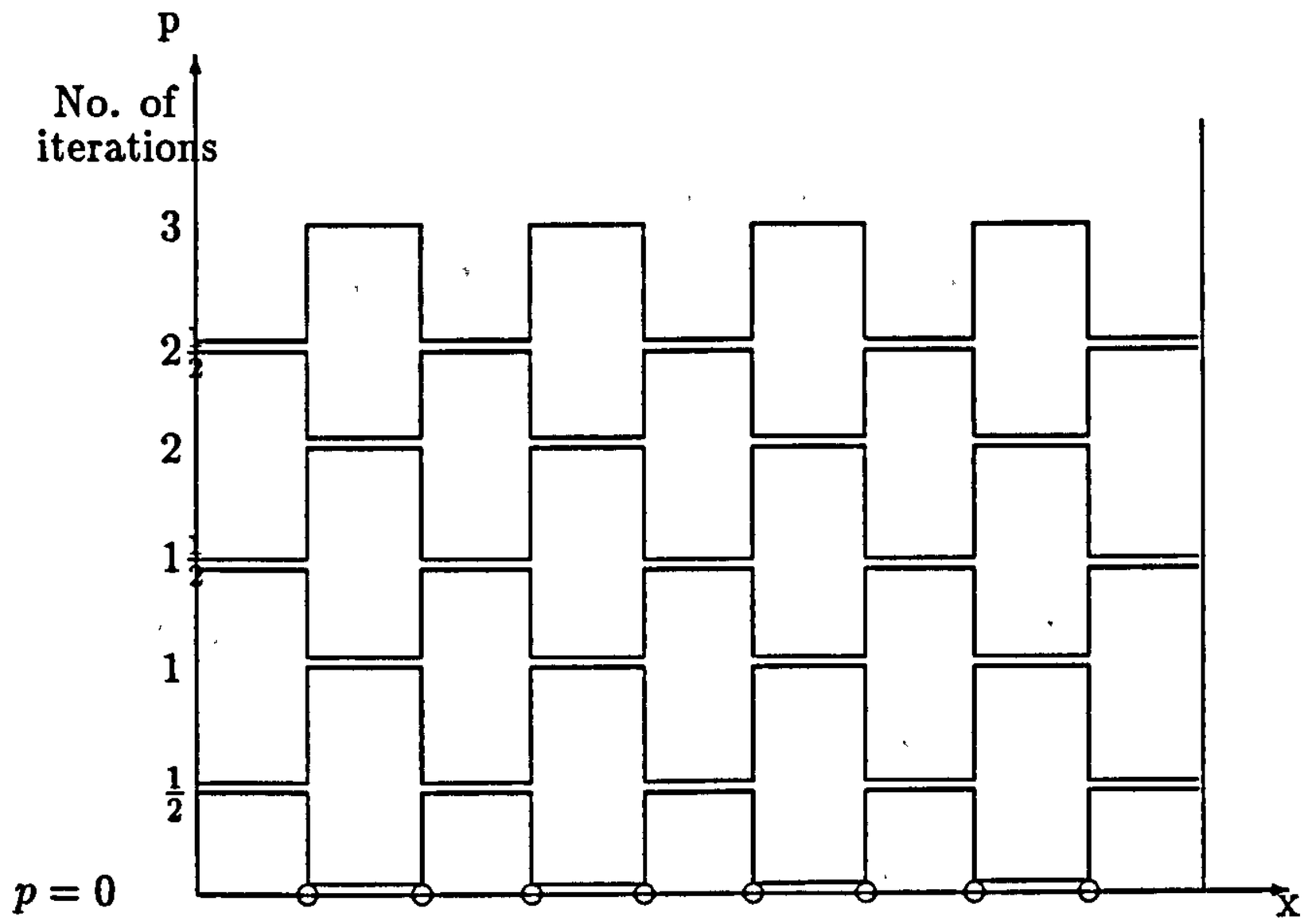


Figure 4.2: The Alternation in the order of meshpoints grouping every half iteration

and (4.69). In the grouping of points we alternate between leaving the first point ungrouped, and leaving the last point ungrouped every half iteration level as shown in figure 4.2. Such grouping of points, the alternation in forming the groups, and the explicit nature of the formulae used for obtaining the solution give the method its name, viz. the Alternating Group Explicit (AGE) iterative method.

It can also be seen by examining the matrix equations (4.66) and (4.67) that obtaining the solution at all the m meshpoints require $3m$ multiplications and $3m-2$ additions (note that $b = c = -1$) for each half iteration. i.e 6m multiplications and 6m additions are needed for every complete single iteration. The more efficient organization of the AGE algorithm given by [16] requires only $4m$ multiplications and $5m - 4$ additions. The above illustrated variant of the AGE method is called the AGE Peaceman-Rachford method due to the analogy of the method with the ADI-PR *iterative* method. Another variant of the method which is also due to Evans and Sahimi is the AGE Douglas-Rachford, (AGE-DR) method which uses a different formula than AGE-PR for the second half iteration. This is given by:

$$(G_1 + sI)\mathbf{u}^{p+1/2} = (sI - G_2)\mathbf{u}^{p} + \mathbf{f}$$

$$(G_2 + sI)\mathbf{u}^{p+1} = G_2\mathbf{u}^p + s\mathbf{u}^{p+1/2} \tag{4.72}$$

It is noted that all Sahimi's experiments (see [41]) show that the AGE-PR has faster convergence than the AGE-DR method. His experiments show also that the AGE-DR method often takes more than twice the number of iterations that the AGE-PR method needs to converge.

4.4.3 Convergence analysis of the AGE method as applied to symmetric matrix systems

The convergence analysis of the stationary AGE-PR method, with a single parameter, as applied to the above heat diffusion problem with Dirichlet boundary conditions is given here after [22]. The AGE-PR method is written in composite form in the following single equation as:

$$\mathbf{u}^{p+1} = G(s)\mathbf{u}^p + \mathbf{k}(s) \tag{4.73}$$

where

$$G(s) = (G_2 + sI)^{-1}(sI - G_1)(G_1 + sI)^{-1}(sI - G_2) \tag{4.74}$$

is the iteration matrix. We now define an error vector $\mathbf{e}$ representing the difference between the computed solution after each iteration $\mathbf{u}^p$ and the exact solution $\overline{\mathbf{u}} = A^{-1}\mathbf{f}$ of (4.54) i.e., we define:

$$\mathbf{e}^p = \mathbf{u}^p - \overline{\mathbf{u}} \quad and \quad \mathbf{e}^{p+1} = \mathbf{u}^{p+1} - \overline{\mathbf{u}}$$

then it can be shown that :

$$\mathbf{e}^p = G^p(s)\mathbf{e}^0, \qquad p \geq 1$$

where

$$\mathbf{e}^0 = \mathbf{u}^0 - \overline{\mathbf{u}}$$

($\mathbf{u}^0$ being the starting vector). Thus, for the method to converge we need to prove that: $\rho(G(s)) < 1$

Proof: If we apply a similarity transformation to $G(s)$ such that :

$$\begin{aligned} \tilde{G}(s) &= (G_2 + sI)G(s)(G_2 + sI)^{-1} \\ &= (sI - G_1)(G_1 + sI)^{-1}(sI - G_2)(G_2 + sI)^{-1} \end{aligned} \tag{4.75}$$

then

$$\begin{aligned} \rho(G(s)) = \quad & \rho(\tilde{G}(s)) \leq || \tilde{G}(s) ||_2 \\ & \leq || (sI - G_1)(G_1 + sI)^{-1} ||_2 \times || (sI - G_2)(G_2 + sI)^{-1} ||_2 \end{aligned} \tag{4.76}$$

where $\rho(G(s))$ and $\rho(\tilde{G}(s))$ are the spectral radii of $G(s)$ and $\tilde{G}(s)$ respectively. It can be seen from equation (4.56) that both G_1 and G_2 are symmetric positive definite (SPD) matrices, thus having positive eigenvalues μ_i and η_i respectively. Since $(sI - G_1)$ and $(G_1 + sI)^{-1}$ commute, and G_1 has positive eigenvalues μ_i, we can write:

$$\begin{aligned} || (sI - G_1)(G_1 + sI)^{-1} ||_2 \quad &= \rho[(sI - G_1)(G_1 + sI)^{-1}] \\ &= \max_{1 \leq i \leq m} \left| \frac{s - \mu_i}{s + \mu_i} \right| < 1 \ (since\ \mu_i > 0) \end{aligned} \tag{4.77}$$

Likewise, $(G_2 + sI)^{-1}$ and $(sI - G_2)$ commute and G_2 has positive eigenvalues η_i and we can write:

$$\begin{aligned} || (sI - G_2)(G_2 + sI)^{-1} ||_2 \quad &= \rho[(sI - G_2)(G_2 + sI)^{-1}] \\ &= \max_{1 \leq i \leq m} \left| \frac{s - \eta_i}{s + \eta_i} \right| < 1 \ (since\ \eta_i > 0) \end{aligned} \tag{4.78}$$

Hence;

$$\begin{aligned} \rho(G(s)) &= \rho(\tilde{G}(s)) \\ &\leq \rho[(sI - G_1)(G_1 + sI)^{-1}]\rho[(sI - G_2)(G_2 + sI)^{-1}] < 1 \end{aligned} \tag{4.79}$$

and convergence is assured.

The optimum parameter $\hat{s}$ for AGE-PR, which minimizes the upper bound of $\rho(M(s))$ can be obtained if bounds α and β are found for the eigenvalues of G_1 and G_2 such that:

$$\alpha \leq \mu_i, \eta_i \leq \beta \tag{4.80}$$

We now write the relation (4.79) as:

$$\begin{aligned}\rho[M(s)] &\leq \left\{\max_{1\leq i\leq m}\left|\frac{s-\eta_i}{s+\eta_i}\right|\right\}\left\{\max_{1\leq i\leq m}\left|\frac{s-\mu_i}{s+\mu_i}\right|\right\} \\ &\leq \left\{\max_{\alpha\leq i\leq\beta}\left|\frac{s-z}{s+z}\right|\right\}^2 = \left\{\max_{\alpha\leq z\leq\beta}\mid\phi(z;s)\mid\right\}^2 \qquad (4.81)\end{aligned}$$

where $\phi(z;s)$ is a (decreasing) function of z and therefore its maximum is at one of the endpoints of the interval.

Hence:

$$\max_{\alpha\leq z\leq\beta}\left|\frac{s-z}{s+z}\right| = \max\left(\left|\frac{s-\alpha}{s+\alpha}\right|,\left|\frac{s-\beta}{s+\beta}\right|\right) \qquad (4.82)$$

When $s=\sqrt{\alpha\beta}$ we have:

$$\frac{s-\alpha}{s+\alpha} = \left|\frac{s-\alpha}{s+\alpha}\right| = \left|\frac{s-\beta}{s+\beta}\right| = \frac{\sqrt{\beta}-\sqrt{\alpha}}{\sqrt{\beta}+\sqrt{\alpha}} \qquad (4.83)$$

For $0<s\leq\sqrt{\alpha\beta}$, it can be verified that:

$$\max_{\alpha\leq z\leq\beta}\left|\frac{s-z}{s+z}\right| = \left|\frac{s-\beta}{s+\beta}\right| = \frac{\beta-s}{s+\beta}$$

and that:

$$\frac{\beta-s}{s+\beta}-\frac{\sqrt{\beta}-\sqrt{\alpha}}{\sqrt{\beta}+\sqrt{\alpha}} = \frac{2\sqrt{\beta}(\sqrt{\alpha\beta}-s)}{(s+\beta)(\sqrt{\beta}+\sqrt{\alpha})} > 0 \qquad (4.84)$$

Similarly for $s\geq\sqrt{\alpha\beta}$ it can be verified that:

$$\max_{\alpha\leq z\leq\beta}\left|\frac{s-z}{s+z}\right| = \left|\frac{s-\alpha}{s+\alpha}\right| = \frac{s-\alpha}{s+\alpha}$$

and that:

$$\frac{s-\alpha}{s+\alpha}-\frac{\sqrt{\beta}-\sqrt{\alpha}}{\sqrt{\beta}+\sqrt{\alpha}} = \frac{2\sqrt{\alpha}(\sqrt{s-\alpha\beta})}{(s+\beta)(\sqrt{\beta}+\sqrt{\alpha})} > 0 \qquad (4.85)$$

Therefore

$$\begin{aligned}\min_{s>0}\max_{\alpha\leq z\leq\beta}\phi(z;s) &= \max_{\alpha\leq z\leq\beta}\left|\phi(z;\sqrt{\alpha\beta})\right| \\ &= \left|\phi(\alpha;\sqrt{\alpha\beta})\right| = \left|\phi(\beta;\sqrt{\alpha\beta})\right| = \phi(\alpha,\sqrt{\alpha\beta})\end{aligned}$$

Therefore the optimum acceleration parameter is:

$$\widehat{s} = \sqrt{\alpha\beta} \tag{4.86}$$

If the AGE algorithm is applied to solve the system 4.54 then the upper (β) and lower (α) bounds of the eigenvalues of G_1 and G_2 are given respectively as:

$$\beta = \frac{a}{2} + 1 \qquad and \qquad \alpha = \frac{a}{2} - 1$$

This makes the formula for the optimum parameter (4.86) be:

$$\widehat{s} = \sqrt{(\frac{a}{2})^2 - 1)} \tag{4.87}$$

For the non stationary AGE method, Sahimi pointed out that it is very difficult to obtain the optimum parameter sequence for the AGE method which will improve its convergence, except in the very special case where G_1 and G_2 commute. This is possible in the above problem if the boundary conditions are periodic and if G_1 and G_2 were of order 4, which is a very restrictive problem.

We finally note from the above analysis that when $s = \widehat{s}$ the spectral radius of the iteration of the AGE method is:

$$\rho(G) = \left(\frac{\sqrt{\beta} - \sqrt{\alpha}}{\sqrt{\beta} + \sqrt{\alpha}}\right)^2 = \left(\frac{\sqrt{P} - 1}{\sqrt{P} + 1}\right)^2 \tag{4.88}$$

where $P = \frac{\beta}{\alpha}$ is the condition number for G_1 and G_2.

This means that the asymptotic rate of convergence for the AGE method is:

$$R_\infty = -\log\left(\frac{1 - P^{\frac{-1}{2}}}{1 + P^{\frac{-1}{2}}}\right)^2 \tag{4.89}$$

If $P \gg 1$ we get:

$$\begin{aligned} R_\infty &= -\log\left(\frac{1 - P^{\frac{-1}{2}}}{1 + P^{\frac{-1}{2}}}\right)^2 \simeq -\log(1 - P^{\frac{-1}{2}})^4 \simeq -\log(1 - 4P^{\frac{-1}{2}}) \\ &\simeq \frac{4}{\sqrt{P}}. \end{aligned} \tag{4.90}$$

4.5 The AGE method for two dimensional parabolic problems

The AGE method was also extended by [20] to solve the heat diffusion problem in two dimensions given by:

$$\frac{\partial \tilde{u}}{\partial t} = \frac{\partial^2 \tilde{u}}{\partial x_1^2} + \frac{\partial^2 \tilde{u}}{\partial x_2^2} + q(x_1, x_2, t) \ , \quad (x_1, x_2, t) \ \in \Re \times t \geq 0 \tag{4.91}$$

where $\Re$ is a rectangular closed region defined by:

$$\Re = (x_1, x_2); \ 0 \leq x_1 \leq L \ and \ 0 \leq x_2 \leq M \ ,$$

subject to the initial conditions:

$$\tilde{u}(x_1, x_2, 0) = f(x_1, x_2) \ \ ; \ \ (x_1, x_2, t) \in \Re \times 0$$

and where $\tilde{u}(x_1, x_2, t)$ is specified at the boundary $\partial\Re$ of $\Re$ as:

$$\tilde{u}(x_1, x_2, t) = g(x_1, x_2, t) \in \partial\Re \times (t \geq 0)$$

To obtain the numerical solution of the above problem we cover the domain with a uniform mesh where the coordinates of the meshpoints P are (x_{1_i}, x_{2_j}, t_l) where $x_{1_i} = ih_1$, $x_{2_j} = jh_2$, and $t_l = lk$, $(0 \leq i \leq m+1)$, $(0 \leq j \leq n+1)$, and $h_1 = L/(m+1)$ and $h_2 = M/(n+1)$, and k is the increment in time. For simplicity, we choose m and n such that $h_1 = h_2 = h$. The mesh ratio is therefore $r = k/h^2$.
A weighted finite difference approximation given by:

$$[1 - \theta r\delta_{x_1}^2 - \theta r\delta_{x_2}^2]U_{ij}^{n+1} = [1 + (1-\theta)r\delta_{x_1}^2 + (1-\theta)r\delta_{x_2}^2]U_{ij}^{n} + kq(x_1, x_2, t) \tag{4.92}$$

is used to replace (4.91) at each meshpoint. Here θ is a weighting parameter which is equal to $\frac{1}{2}$ and 1 for the Crank-Nicholson and the Implicit schemes respectively. This results in a system of equations of order $n \times m$ for the totality of meshpoints at each time step. This system can again be written in the form:

$$A\mathbf{u}^{l+1} = \mathbf{f} \tag{4.93}$$

where **f** is a known vector of order $m \times n$ consisting of the boundary values, the source term values at each point, and the solution values at the time level $\mathbf{u}^l$. The vector $\mathbf{u}^{l+1}$ is the solution vector which is to be calculated for the time level $l+1$. The matrix A now has the form:

$$A = \begin{pmatrix} T & a_1 I & & \\ a_2 I & \ddots & \ddots & 0 \\ & \ddots & \ddots & \ddots \\ 0 & \ddots & \ddots & a_1 I \\ & & a_2 I & T \end{pmatrix}_{mn \times mn} \tag{4.94}$$

where I is an $m \times m$ identity matrix and T is an $m \times m$ matrix given as:

$$T = diag(a_2, b, a_1) \; with \; b = (1 + 4r\theta) \; and \; a_1 = a_2 = -r\theta$$

where θ is a weighting parameter of the difference scheme.

If equation (4.92) is divided by $r\theta$, we obtain a system which is similar to (4.93) but the coefficients b, a_1, and a_2 become:

$$b = 4 + \frac{1}{r\theta} = 4(1+\beta) \; where \; \beta = \frac{1}{4r\theta} > 0, \quad a_1 = a_2 = -1 \tag{4.95}$$

This latter form of the coefficient matrix makes the application of AGE-2D method, which is introduced next, to solve (4.93) more economic.

The AGE iterative method which was formulated by [20] for solving (4.93) consists of splitting A into four matrices G_1, G_2, G_3, and G_4 such that:

$$G_1 + G_2 = \begin{pmatrix} T_1 & & \\ & \ddots & 0 \\ & & \ddots \\ 0 & & \ddots \\ & & & T_1 \end{pmatrix}_{mn \times mn} \tag{4.96}$$

where $T_1 = diag(a_2, b/2, a_1)$.

and

$$G_3 + G_4 = \begin{pmatrix} \frac{b}{2}I & a_1 I & & \\ a_2 I & \ddots & \ddots & 0 \\ & \ddots & \ddots & \ddots \\ & & \ddots & \ddots & a_1 I \\ & 0 & & a_2 I & \frac{b}{2}I \end{pmatrix}_{mn \times mn} \tag{4.97}$$

The matrices T_1 and I are of order m, since a natural ordering of the components of the solution vector is assumed. Hence $(G_1 + G_2)$ is tridiagonal and $(G_3 + G_4)$ is block tridiagonal. The AGE method for such two dimensional problems, henceforth referred to as AGE-2D is given in four steps as:

$$\begin{aligned} (G_1 + sI)\mathbf{u}_r^{p+1/4} &= (sI - G_1 - 2G_2 - 2G_3 - 2G_4)\mathbf{u}_r^p + 2\mathbf{f} \equiv \mathbf{v_1} \\ (G_2 + sI)\mathbf{u}_r^{p+1/2} &= G_2\mathbf{u}_r^p + s\mathbf{u}_r^{p+1/4} \equiv \mathbf{v_2} \\ (G_3 + sI)\mathbf{u}_r^{p+3/4} &= G_3\mathbf{u}_r^p + s\mathbf{u}_r^{p+1/2} \equiv \mathbf{v_3} \\ (G_4 + sI)\mathbf{u}_r^{p+1} &= G_4\mathbf{u}_r^p + s\mathbf{u}_r^{p+3/4} \equiv \mathbf{v_4} \end{aligned} \tag{4.98}$$

where the suffix 'r' under the solution vector indicates a row-wise ordering of its components, s is the acceleration parameter. The first equation of (4.98) can also be written as:

$$(G_1 + sI)\mathbf{u}_r^{p+1/4} = (G_1 + sI - 2A)\mathbf{u}_r^p + 2\mathbf{f} \tag{4.99}$$

Again, without loss of generality, the size of the matrix is assumed to be odd. i.e n and m are odd. Then G_1 and G_2 can be given as:

$$G_1 = \begin{bmatrix} C_1 & & \\ & \ddots & 0 \\ & 0 & \ddots \\ & & & C_1 \end{bmatrix}_{mn \times mn} \quad and\, G_2 = \begin{bmatrix} C_2 & & \\ & \ddots & 0 \\ & 0 & \ddots \\ & & & C_2 \end{bmatrix}_{mn \times mn} \tag{4.100}$$

or alternatively as:

$$G_1 = \begin{bmatrix} C_1 & & & & \\ & C_2 & & 0 & \\ & & C_1 & & \\ & & & \ddots & \\ & & & & \ddots \\ & & 0 & & & C_2 \\ & & & & & & C_1 \end{bmatrix}_{mn \times mn} \tag{4.101}$$

and

$$G_2 = \begin{bmatrix} C_2 & & & & \\ & C_1 & & 0 & \\ & & C_2 & & \\ & & & \ddots & \\ & & & & \ddots \\ & & 0 & & & C_1 \\ & & & & & & C_2 \end{bmatrix}_{mn \times mn} \tag{4.102}$$

where

$$C_1 = \begin{pmatrix} c & & & \\ & \begin{bmatrix} c & a_1 \\ a_2 & c \end{bmatrix} & & \\ & & \ddots & \\ & & & \begin{bmatrix} c & a_1 \\ a_2 & c \end{bmatrix} \end{pmatrix} \tag{4.103}$$

and

$$C_2 = \begin{pmatrix} \begin{bmatrix} c & a_1 \\ a_2 & c \end{bmatrix} & & & \\ & \ddots & & \\ & & \ddots & \\ & & & \begin{bmatrix} c & a_1 \\ a_2 & c \end{bmatrix} \\ & & & & c \end{pmatrix} \tag{4.104}$$

with $c = b/4$. The choice of G_1 and G_2 in equation (4.100) seems to be the natural choice, but for reasons not discussed by the above authors, they chose G_1 and G_2 as given by equations (4.101) and (4.102). Numerical experiments carried out by the author showed that Sahimi and Evans' choice is marginally better than the seemingly natural choice given by equations (4.100), because it produces improved convergence. Equations (4.103) and (4.104) correspond to the different grouping (alternating) of the mesh points along each mesh line with each subiteration level as shown in figure 4.2.

Equations (4.100) correspond to no alternation in space (figure 4.3 - right) while equations (4.101) and (4.102) correspond to alternating in space (figure 4.3 - left).

The matrices G_3 and G_4 are given as:

$$G_3 = \begin{pmatrix} T_2 & & & \\ & \ddots & 0 & \\ & & \ddots & \\ & & & T_2 \\ & 0 & & & \frac{b}{4}I \end{pmatrix}_{mn \times mn} \tag{4.105}$$

Alternation in the grouping of the meshpoints in space (along the x_2 dimension), as well as with iteration levels.

Alternation in the grouping of the meshpoints only with iteration levels, and not along the x_2 dimension.

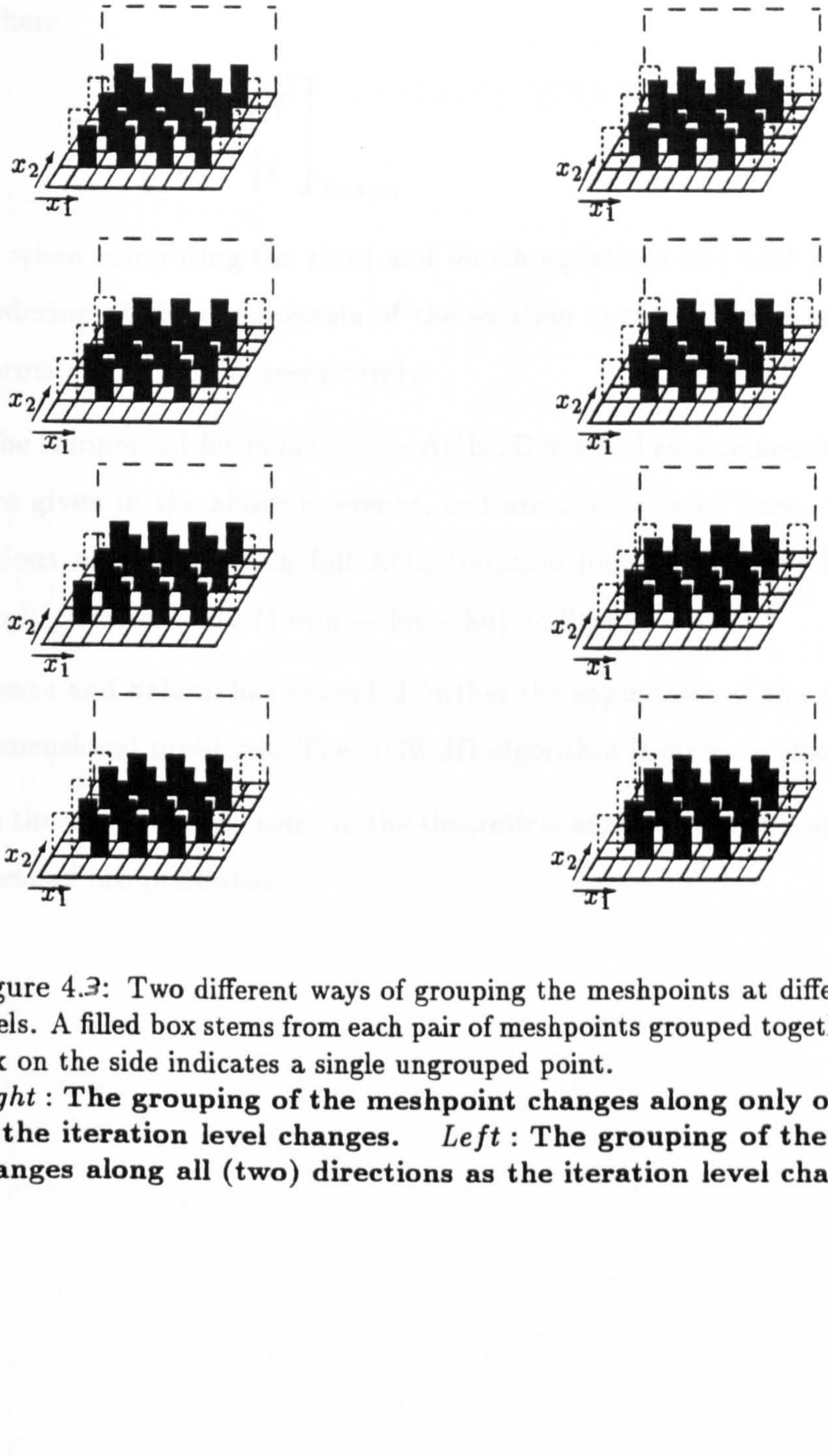


Figure 4.3: Two different ways of grouping the meshpoints at different iteration levels. A filled box stems from each pair of meshpoints grouped together. A dashed box on the side indicates a single ungrouped point.
Right : **The grouping of the meshpoint changes along only one direction as the iteration level changes.** *Left* : **The grouping of the meshpoints changes along all (two) directions as the iteration level changes**

and

$$G_4 = \begin{pmatrix} \frac{b}{4}I & & & \\ & T_2 & & 0 \\ & & \ddots & \\ & & & \ddots \\ & 0 & & T_2 \end{pmatrix}_{mn \times mn} \tag{4.106}$$

where

$$T_2 = \begin{bmatrix} \frac{b}{4}I & a_1 I \\ a_2 I & \frac{b}{4}I \end{bmatrix}_{2m \times 2m}$$

If when computing the third and fourth equations of (4.98) we use a column-wise ordering of the components of the solution vector, G_3 and G_4 will have the same forms as G_1 and G_2 respectively.

The component formulae for the AGE-2D method as obtained from equations (4.98) are given in the above reference, and are not repeated here. The number of operations needed for each full AGE iteration (equations 4.98) is $(16mn - 2m - 4n)$ multiplications and $(15mn - 2m - 8n)$ additions.

Evans and Sahimi has extended further the application of the AGE method to three dimensional problems. The AGE-3D algorithm is given in chapter 6.

In the next chapter, some of the theoretical analyses which is still missing about the method are presented.

Chapter 5

Further developments of the AGE methods

5.1 Introduction

In this chapter we consider the consistency of the AGE-1D and AGE-2D algorithms and the convergence of the AGE-1D method when applied to the hyperbolic linear advection equation. A theoretical examination of whether a Chebyshev acceleration of the AGE-1D method is possible is also given together with other notes on the applications of the AGE method.

5.2 Consistency analysis of the AGE method

The consistency analysis for the AGE method is omitted in the literature presently available on the method, and is therefore given in this section for the AGE-PR, and AGE-DR variants in one dimension as well as the AGE-2D method for two dimensional problems. The AGE-PR algorithm for solving the tridiagonal system:

$$A\mathbf{u} = \mathbf{f} \tag{5.1}$$

where A is a tridiagonal Toeplitz matrix, is given earlier in chapter 4.4. The composite form (4.73) of the AGE-PR algorithm is again given here by:

$$\mathbf{u}^{p+1} = G(s)\mathbf{u}^{p} + \mathbf{k}(s) \tag{5.2}$$

where s is the acceleration parameter. The iteration matrix G and the vector k are given as:

$$G(s) = (G_2 + sI)^{-1}(sI - G_1)(G_1 + sI)^{-1}(sI - G_2) \tag{5.3}$$

and

$$\begin{aligned} \mathbf{k} &= (G_2 + sI)^{-1}[(sI - G_1)(G_1 + sI)^{-1} + I]\,\mathbf{f} \\ &= 2s(G_2 + sI)^{-1}(G_1 + sI)^{-1}\mathbf{f} \end{aligned}$$

The exact solution for (5.1) is $\bar{\mathbf{u}} = A^{-1}\mathbf{f}$. Thus for (5.2) to be consistent with (5.1), the following condition (of theorem 3.1):

$$A^{-1}\mathbf{f} = GA^{-1}\mathbf{f} + \mathbf{k} \Rightarrow \mathbf{k} = (I - G)A^{-1}\mathbf{f}$$

must be satisfied.

If also $(I-G)$ is nonsingular, then by (theorem 3.3) the method is *completely consistent.*

Lemma 5.1 *The AGE-PR method is completely consistent.*

Proof The iteration matrix G given above can be written as:

$$\begin{aligned} G &= (G_2 + sI)^{-1}[(G_1 + sI - 2sI)(G_1 + sI)^{-1}(A - G_1 - sI)] \\ &= (G_2 + sI)^{-1}\{\left(I - 2s(G_1 + sI)^{-1}\right)[A - (G_1 + sI)]\} \\ &= (G_2 + sI)^{-1}\{(A - G_1 - sI) - 2s(G_1 + sI)^{-1}A + 2sI\} \\ &= (G_2 + sI)^{-1}\{(G_2 + sI) - 2sI - 2s(G_1 + sI)^{-1}A + 2sI\} \\ &= I - 2s(G_2 + sI)^{-1}(G_1 + sI)^{-1}A \end{aligned} \tag{5.4}$$

$$\text{Now} \quad (I-G)A^{-1}\mathbf{f} = [I - I + 2s(G_2+sI)^{-1}(G_1+sI)^{-1}A]A^{-1}\mathbf{f}$$
$$= 2s(G_2+sI)^{-1}(G_1+sI)^{-1}\mathbf{f}$$
$$= \mathbf{k}$$

Therefore the AGE-PR method is consistent.

Also from (5.4) we have:

$$(I-G) = 2s(G_2+sI)^{-1}(G_1+sI)^{-1}A$$

which implies that $(I-G)$ is the product of nonsingular matrices and is therefore nonsingular.

Thus the AGE-PR method is also *reciprocally consistent* and hence the proof is completed.

Similarly we can state and verify the following lemma.

Lemma 5.2 *The AGE-DR method is completely consistent.*

Proof The AGE-DR method given by the pair of equations (4.72) can also be written in the composite form (5.27) where now:

$$G = G_{DR} = (G_2+sI)^{-1}(G_1+sI)^{-1}[(G_1+sI)G_2 + s(sI-G_2)]$$
$$and\ \mathbf{k} = s(\ G_2+sI)^{-1}(G_1+sI)^{-1}\mathbf{f}$$

Thus,

$$G = (G_2+sI)^{-1}(G_1+sI)^{-1}[(G_1+sI)G_2 + s(sI+G_1-A)]$$
$$= (G_2+sI)^{-1}(G_1+sI)^{-1}[(G_1+sI)(G_2+sI) - sA)]$$
$$= I - s(G_2+sI)^{-1}(G_1+sI)^{-1}A \qquad (5.5)$$

Therefore $(I-G)$ is nonsingular.

Also $(I-G)A^{-1}\mathbf{f} = \mathbf{k}$. Hence the AGE-DR method is *completely consistent.*

Lemma 5.3 *The AGE-2D method given by equations (4.98) for two dimensional problems is completely consistent.*

The set of equations (4.98), can be written in the composite form of (5.27) where G is now given as:

$$\begin{aligned} G &= (G_4+sI)^{-1}G_4 + s(G_4+sI)^{-1}(G_3+sI)^{-1}G_3 \\ &\quad +s^2\prod_{i=4}^{2}(G_i+sI)^{-1}G_2 \\ &\quad +s^3\prod_{i=4}^{2}(G_i+sI)^{-1} - 2s^3\prod_{i=4}^{1}(G_i+sI)^{-1}A \end{aligned} \tag{5.6}$$

and

$$\mathrm{k} = 2s^3\prod_{i=4}^{1}(G_i+sI)^{-1}\mathrm{f} \tag{5.7}$$

The iteration matrix G can be manipulated as:

$$\begin{aligned} G &= (G_4+sI)^{-1}(G_4+sI-sI) + s(G_4+sI)^{-1}(G_3+sI)^{-1}(G_3+sI-sI) \\ &\quad +s^2\prod_{i=4}^{2}(G_i+sI)^{-1}(G_2+sI-sI) + s^3\prod_{i=4}^{2}(G_i+sI)^{-1} \\ &\quad -2s^3\prod_{i=4}^{1}(G_i+sI)^{-1}A \end{aligned}$$

which reduces after some cancellations to:

$$G = I - 2s^3\prod_{i=4}^{1}(G_i+sI)^{-1}A \tag{5.8}$$

from which we can easily deduce that $(I-G)$ is nonsingular and that:

$$(I-G)A^{-1}\mathrm{f} = \mathrm{k}$$

which concludes the proof.

5.2.1 A comment on the AGE-PR and the AGE-DR methods

It has been mentioned earlier that according to Sahimi's [41] experiments the AGE-PR method proved always to converge faster than the AGE-DR method. These

experimental results are justified theoretically here, by comparing the spectral radii of the iteration matrices of the two methods.

The iteration matrices G_{PR} and G_{DR} are given respectively by the last equation in (5.4 and 5.5), from which we can write:

$$G_{DR} = \frac{1}{2}[I + G_{PR}] \tag{5.9}$$

Therefore, (when G_1 and G_2 are both positive definite) we can write:

$$\begin{aligned} \rho(G_{DR}) &= \frac{1}{2} + \frac{1}{2}\rho(G_{PR}) \\ \rho(G_{DR}) &= \frac{1 + \rho(G_{PR})}{2} \end{aligned} \tag{5.10}$$

Thus it can be easily seen that for $\rho(G_{PR}) = 1$, we have $\rho(G_{DR}) = 1$, while for any $\rho(G_{PR}) < 1$ we always have

$$\rho(G_{DR}) > \rho(G_{PR})$$

and hence the AGE-PR method is always faster to converge . Moreover AGE-DR requires one more multiplication operation at each meshpoint per iteration than the AGE-PR method. It is therefore clear that the AGE-PR is superior to the AGE-DR method.

5.3 The AGE method for hyperbolic problems

It has been mentioned earlier that the AGE method has been successfully applied to hyperbolic problems. These problems include the second order wave equation, and first order advection equation.

For the second order wave equation: we have

$$\frac{\partial^2 \tilde{u}}{\partial t^2} = \frac{\partial^2 \tilde{u}}{\partial x^2} \quad 0 \le x \le 1, 0 \le t < T \tag{5.11}$$

subject to the following auxiliary conditions,

$$\left.\begin{aligned} \tilde{u}(x,0) &= g_1(x) \\ \tfrac{\partial \tilde{u}}{\partial t}(x,0) &= g_2(x) \end{aligned}\right\} \quad 0 \le x \le 1 \tag{5.12}$$

$$\left.\begin{array}{l} U(0,t) = E_1(t) \\ U(1,t) = E_2(t) \end{array}\right\} 0 \leq t \leq T \qquad (5.13)$$

and

Evans and Sahimi [21] applied a general three level implicit approximation which leads to solving a symmetric tridiagonal system as in the case of solving the parabolic heat problem, described in section 4.4. Therefore in this case the same convergence analysis given in the that section applies.

However, first order equations lead to the solution of unsymmetric difference systems of equations when implicit schemes based on central differences are applied. In this section, the theoretical proof of the convergence of the AGE method for such problems is provided, together with a formula determining the choice of the 'best' parameter for the method.

Consider the linear advection hyperbolic equation.

$$\frac{\partial \tilde{u}}{\partial t} + c\frac{\partial \tilde{u}}{\partial x} = 0 \qquad (5.14)$$

in the domain $\Re = a \leq x \leq b,\ t \geq 0$ satisfying the following initial conditions:

$$\tilde{u}(x,0) = g(x) \qquad (5.15)$$

with Dirichlet (I) and periodic (II) boundary conditions,

$$\left.\begin{array}{ll} Problem I: & \tilde{u}(a,t) = q_1(t), t > 0 \\ & \tilde{u}(b,t) = q_2(t), t > 0 \end{array}\right\} \qquad (5.16)$$

$$Problem II: \quad \tilde{u}(a,t) = \tilde{u}(b,t), t > 0 \qquad (5.17)$$

where c is a constant which stands for the speed of advection.

For the numerical solution of the above problem, we cover its domain with a mesh, having a uniform spacing h in the x direction, and a uniform time increment k along the t direction. If at each point $(i, j + \frac{1}{2})$ we replace (5.14) by the following weighted central difference approximation:

$$-eU_{i-1,j+1} + U_{i,j+1} + eU_{i+1,j+1} = -qU_{i-1,j} + U_{i,j} + qU_{i+1,j} \qquad (5.18)$$

where

$$e = \frac{\theta ck}{2h} \quad and \quad q = \frac{(1-\theta)ck}{2h} \tag{5.19}$$

where $\theta = \frac{1}{2}$ for the Crank Nicholson (CN) type scheme.

where $\theta = 1$ for the fully Implicit scheme.

we obtain the following linear system to be solved at each time level $(j+1)k$:

$$Au_{j+1} = d_j \tag{5.20}$$

where d_j is a known vector of order n, and the matrix A has the form:

$$A = \begin{bmatrix} 1 & e & & \\ -e & \ddots & \ddots & 0 \\ & \ddots & \ddots & \ddots \\ & 0 & \ddots & \ddots & e \\ & & & -e & 1 \end{bmatrix}_{n\times n} \quad for\ problem I \tag{5.21}$$

and

$$A = \begin{bmatrix} 1 & e & & & -e \\ -e & \ddots & \ddots & 0 & \\ & \ddots & \ddots & \ddots & \\ & 0 & \ddots & \ddots & e \\ e & & & -e & 1 \end{bmatrix}_{n\times n} \quad for\ problem II \tag{5.22}$$

Now to solve (5.20) we split A as:

$$A = G_1 + G_2 \tag{5.23}$$

where assuming n is even, G_1 is given by:

$$G_1 = \begin{bmatrix} c & & & \\ & c & & 0 \\ & & c & \\ & 0 & & c \\ & & & & c \end{bmatrix}_{n\times n} \tag{5.24}$$

and G_2 is given as:

$$G_2 = \begin{bmatrix} 0.5 & & & \\ & C & 0 & \\ & & C & \\ & 0 & & C \\ & & & 0.5 \end{bmatrix}_{n\times n} \quad for \quad problem\ (I) \tag{5.25}$$

and

$$G_2 = \begin{bmatrix} 0.5 & & & -e \\ & C & 0 & \\ & & C & \\ & 0 & & C \\ e & & & 0.5 \end{bmatrix}_{n\times n} \quad for \quad problem\ (II) \tag{5.26}$$

where $C = \begin{bmatrix} 0.5 & e \\ -e & 0.5 \end{bmatrix}$

Now the AGE method, as presented in section 4.4 is applied to solve (5.20). The method in its composite form is given by:

$$\mathbf{u}^{p+1} = G(s)\mathbf{u}^p + \mathbf{k}(s) \tag{5.27}$$

where s is the acceleration parameter. The iteration matrix G and the vector k are given as:

$$G(s) = (G_2 + sI)^{-1}(G_1 - sI)(G_1 + sI)^{-1}(G_2 - sI)$$

and $\mathbf{k} = 2s(G_2 + sI)^{-1}(G_1 + sI)^{-1}\mathbf{d}_j$

5.3.1 A note on using central difference approximations for advection problems

Before proceeding with our convergence analysis in this section, we make the following note on the finite difference approximations of the first order hyperbolic advection equation. The truncation error expressions for most of the first and second

order finite difference replacements of the advection equation (5.14) have leading terms which are products of the second and/or third order spatial differences i.e $\frac{\partial^2 \tilde{u}}{\partial x^2}$ and $\frac{\partial^3 \tilde{u}}{\partial x^3}$ (see [37] p:67 and [24] p:278-279). This means that the solutions obtained by such methods for the advection equation are actually solutions for the transport equation given by :

$$\frac{\partial \tilde{u}}{\partial t} = c\frac{\partial \tilde{u}}{\partial x} + \mathcal{D}_{diff}\frac{\partial^2 \tilde{u}}{\partial x^2} \tag{5.28}$$

or the linear Korteweg de Vries equation given by:

$$\frac{\partial \tilde{u}}{\partial t} = c\frac{\partial \tilde{u}}{\partial x} + \mathcal{D}_{disp}\frac{\partial^3 \tilde{u}}{\partial x^3} \tag{5.29}$$

The terms $\frac{\partial^2 \tilde{u}}{\partial x^2}$ and $\frac{\partial^3 \tilde{u}}{\partial x^3}$ are referred to respectively as the diffusion term and the dispersion term. The effect of the diffusion term on a wavelike solution is to dampen its amplitude, while the effect of the dispersion term when the solution consists of different waves, is to advance at a differential speed the different waves of the solution, so that waves with short wavelengths will have much smaller *phase speeds* than those of large wavelengths, thus leading to an oscillatory behaviour in the solution (for illustration see the above two references). $\mathcal{D}_{diff}$ and $\mathcal{D}_{disp}$ are respectively the coefficients of the leading diffusion and dispersion terms in the truncation error expressions of the finite difference approximations to the advection equation, and are functions of k and h the time and space steps of the numerical solution.

In contrast, an exact wavelike solution for (5.14) would be non attenuative and non dispersive. Schemes such as the *upwind* scheme given in the above references suffer from the artificial diffusivity introduced by its truncation error expression and hence represents more an approximation to the transport equation (5.28) rather than to equation (5.14). Such a scheme is much preferred due to its explicit nature and ability to provide a very good representation of the phase speed of the solution. It is of first order accuracy, and like all other explicit difference schemes has a stability condition which coincides with the *Courant-Friedrichs-Lewy* (CFL) condition, (the CFL condition requires that a particle of fluid should not travel more than one spatial step-size h in one time step k) i.e. it requires that $c\frac{k}{h} \leq 1$. On the

other hand, a scheme like the Crank-Nicholson scheme is of second order accuracy and is unconditionally stable (neutral stability). It is however a dispersive scheme that advances different modes of the solution at different speeds, and is more an approximation of equation (5.29) rather than equation (5.14). This means that a severe upper limit on the time step (to reduce $\mathcal{D}_{disp}$) should be imposed to sufficiently cut down the dispersion if all the modes of the solution are to be adequately represented. This restriction eliminates the advantage of the unconditional stability of the scheme. However for some problems, like in weather forecasting models, high-speed oscillations (corresponding to short wavelength components of the solution, such as gravity waves ... etc.) are unimportant, so that the deceleration of their phase speed poses no problem [23].

We note finally that the choice of a difference scheme foran advection problem depends on the particular nature of the solution sought and involves mostly a trade off between the effects of artificial diffusion and dispersion introduced by various finite difference approximations. We now proceed with the following convergence analysis.

5.3.2 Convergence analysis

Now we study the convergence of the AGE method which is governed by its iteration matrix G.

Since G is similar to the matrix:

$$M = (G_1 - sI)(G_1 + sI)^{-1}(G_2 - sI)(G_2 + sI)^{-1} \tag{5.30}$$

then

$$\rho(G) = \rho(M) \leq || M ||_2 \tag{5.31}$$

By the definitions of G_1 and G_2 (see equations 5.24, 5.25, 5.26), it can be shown that:

$$(G_1 - sI)(G_1 + sI)^{-1} = \begin{bmatrix} D & & & \\ & D & & 0 \\ & & D & \\ & 0 & & D \\ & & & & D \end{bmatrix}_{n \times n} \tag{5.32}$$

where

$$D = (C - sI)(C + sI)^{-1}$$

and $(G_2 - sI)(G_2 + sI)^{-1}$ is given as:

$$(G_2 - sI)(G_2 + sI)^{-1} = \begin{bmatrix} \alpha & & & \\ & D & & 0 \\ & & D & \\ & 0 & & D \\ & & & & \alpha \end{bmatrix}_{n \times n} \quad \textit{for } G_2 \textit{ having the form in } (5.25). \tag{5.33}$$

with $\alpha = \dfrac{0.5 - s}{0.5 + s}$ *for* G_2 *having the form* (5.25) *and that :*

$$PG_2P = \begin{bmatrix} C^T & & & \\ & C & & 0 \\ & & \ddots & \\ & 0 & & \ddots \\ & & & & C \end{bmatrix} \quad \textit{for } G_2 \textit{ having the form } (5.26). \tag{5.34}$$

where P is a permutation matrix given as:

$$P = \begin{bmatrix} 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 1 \\ & & 0 & 1 & 0 \\ & & & & \\ 0 & 1 & & 0 & \end{bmatrix} \tag{5.35}$$

Hence in the latter case, we have:

$$P(G_2 - sI)(G_2 + sI)^{-1}P = \begin{bmatrix} D^T & & & \\ & D & & 0 \\ & & \ddots & \\ & 0 & & \ddots \\ & & & & D \end{bmatrix} \tag{5.36}$$

From the inequality (5.31) we can write:

$$\begin{aligned} \rho(G) &\leq \| (G_1 - sI)(G_1 + sI)^{-1} \|_2 \| (G_2 - sI)(G_2 + sI)^{-1} \|_2 \\ &= \| D \|_2 \| P(G_2 - sI)(G_2 + sI)^{-1}P \|_2 \end{aligned}$$

Therefore,

$$\rho(G) = \| D \|_2 \times \max\{| \alpha |, \| D \|_2\} \tag{5.37}$$

for G_2 having the form in (5.25).

$$\textit{and}\ \ \rho(G) = \| D \|_2 \| D^T \|_2 = \| D \|_2^2 \tag{5.38}$$

for G_2 having the form in (5.26).

Note that:

$$\| D \|_2^2 \leq \| C - sI \|_2^2 \| (C + sI)^{-1} \|_2^2 = \frac{(0.5 - s)^2 + e^2}{(0.5 + s)^2 + e^2} < 1 \tag{5.39}$$

$$\textit{and}\ \ | \alpha | = \left| \frac{0.5 - s}{0.5 + s} \right| < 1.$$

Hence by (5.37 and 5.38) the AGE method is convergent for any $s > 0$.

We can further determine a "good choice" s^* of s as follows:

Since $| \alpha | \leq \| D \|_2$ for any real number e, therefore

$$\rho(G) \leq F(s, e) = \frac{(0.5 - s)^2 + e^2}{(0.5 + s)^2 + e^2} \tag{5.40}$$

for each case. Note that:

$$\frac{dF}{ds} = 2\frac{[s^2 - (0.25 + e^2)]}{[(0.5 + s)^2 + e^2]^2} \tag{5.41}$$

Hence,

$$\frac{dF}{ds} = \begin{cases} < 0 & if \quad 0 < s < s^* \\ = 0 & if \quad s = s^* \\ > 0 & if \quad s > s^* \end{cases}$$

where s^* is given by:

$$s^* = \sqrt{0.25 + e^2} \tag{5.42}$$

Thus, by substituting for s by s^* in (5.40) we get:

$$\rho(G) \leq F(s^*, e) = \frac{\left(0.5 - \sqrt{0.25 + e^2}\right)^2 + e^2}{\left(0.5 + \sqrt{0.25 + e^2}\right)^2 + e^2} = \frac{2\sqrt{0.25 + e^2} - 1}{2\sqrt{0.25 + e^2} + 1} \tag{5.43}$$

The parameter s^* is defined in terms of the parameter $e = \frac{\theta ck}{2h}$, which is governed by the chosen mesh sizes and can be determined ' á priòri'. The following numerical results in table 5.1 show that the above choice for $s = s^*$ is best. They also show that the AGE method is not too sensitive to the choice of s, i.e there is an interval $(s_L, s_R) \ni s^*$ such that the AGE method with any s in this interval has the same convergence rate.

5.3.3 Numerical results

The following problem of linear advection i.e:

$$\frac{\partial \tilde{u}}{\partial t} + \frac{\partial \tilde{u}}{\partial x} = 0 \tag{5.44}$$

defined in the region $\Re = 0 \leq x \leq 2\pi; t \geq 0$, with periodic boundary conditions and initial conditions obtained from the analytical solution given as:

$$u = \cos(x - t) \tag{5.45}$$

dt=0.01			dx=0.01			e=0.48			$s^* = 0.707$	
s	0.2	0.4	0.707*	1.2	1.3	1.9	2.7	3.1	3.7	4.0
NIT	6	4	4	4	5	6	7	8	8	9
dt=0.02			dx=0.03			e=0.33			$s^* = 0.6$	
s	0.2	0.4	0.6*	0.9	1.2	1.7	2.3	2.7	3.1	3.7
NIT	6	4	4	4	5	6	7	8	9	10
dt=0.08			dx=0.03			e=1.3			$s^* = 1.39$	
s	0.2	0.4	0.6	0.8	1.2	1.39*	1.9	2.7	3.1	4.0
NIT	26	16	12	11	10	10	10	11	12	14
dt=0.08			dx=0.014			e=2.85			$s^* = 2.89$	
s	0.2	0.6	0.8	1.3	1.9	2.7	2.89*	3.4	3.7	4.5
NIT	79	36	29	22	19	17	17	17	18	19
dt=0.04			dx=0.14			e=0.142			$s^* = 0.52$	
s	0.1	0.2	0.3	0.5	0.52*	0.8	1.0	1.2	1.9	3.1
NIT	9	5	4	3	3	4	4	5	7	10

Table 5.1: This table shows how the number of iterations (NIT) needed to obtain convergence for the AGE Algorithm varies with the acceleration parameter 's'. It also shows how the optimum value for 's' agrees with the *best parameter* s^* calculated using equation (5.42) for different values of e.

was solved by the AGE method. The fully implicit scheme was used in this example for the discretization of equation 5.44. Thus the variable e given by equation (5.19) is now $e = k/2h$.

The value of e was varied by varying the time and space steps k and h. The Number of iterations (NIT) needed for the convergence of the AGE method was obtained for runs with different values of the acceleration parameter s, and the results are presented in table 5.1. It can be seen from table 5.1 that the choice of the parameter s^* given by equation (5.43) is the best choice. The table also illustrates that the AGE method is not too sensitive to the choice of s.

5.4 Convergence analysis of the AGE-2D method

In this section the convergence of the AGE-2D method for the two dimensional heat conduction problem of chapter 4 is considered.

The iteration matrix G of the AGE-2D algorithm (4.98) can be written (see equation (5.8) as:

$$G_{AGE-2D} = I - 2s^3 \prod_{i=4}^{1} (G_i + sI)^{-1} A \tag{5.46}$$

The convergence of the method is governed by the spectral radius of G, and is guaranteed if

$$\rho(G) < 1 \, . \tag{5.47}$$

However it can be seen from (5.46) that very little can be deduced about the spectral radius of the iteration matrix G. We therefore use the numerical results below to verify the convergence of the algorithm and indicate the best range of the acceleration parameter to obtain the most rapid convergence of the method.

In the following results the algorithm is used to solve the system:

$$A\mathbf{u} = \mathbf{b}$$

where the coefficient matrix A is given by equation 4.94 and the coefficients of A are given equation 4.95. The right hand side vector **b** for the model problem is chosen such that it makes the components of the solution vector equal to unity.

The numerical results for a wide range of values of $r\theta$ and different values of the acceleration parameter s are shown in figure 5.1. The numerical results indicate that the algorithm converges well for increasing values of $r\theta > 0$ for every $1 < s < 4.0$. It can be seen from figure 5.1 that for all values of $r\theta$ the optimum value/s of s lie in the above interval with the rate of convergence decreasing towards the sides of the

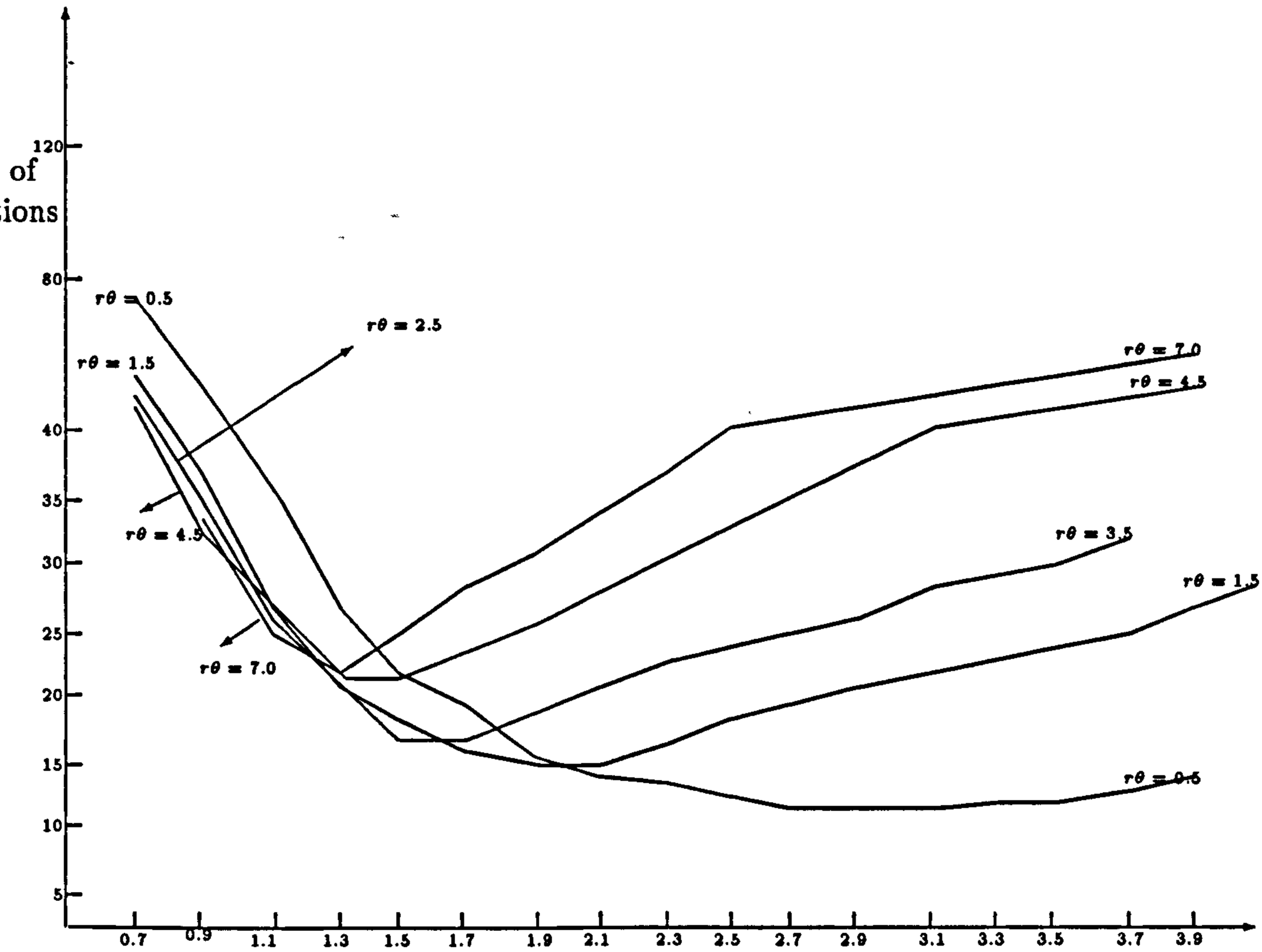


Figure 5.1: This figure shows how the number of iterations needed for the AGE-2D method to converge vary with the acceleration parameter s. Notice that when the values of $r\theta$ increase the curve gets steeper around the best range of values of s.

interval. The position of this optimum value/s gets shifted to the left of the interval (to around s=1.3) as the values of $r\theta$ become large.

It is also noted that as the value of $r\theta$ (i.e., the mesh ratio) increases the convergence curve gets steeper around the best value/s of s, which is quite unfavourable for the practical estimation of s. However, the curves show that an over estimate of s is preferable.

5.5 The AGE-1D for block symmetric systems

In this section we consider the convergence of the AGE-1D method as applied to a block symmetric system of equations. The system in consideration can arise from a finite difference approximation to a coupled system of equations like the one considered in section 7.2. This is given as:

$$A\mathbf{w} = \mathbf{b} \tag{5.48}$$

where b is a known vector and A is a block symmetric matrix of the form:

$$A = \begin{pmatrix} E & -I_2 & & \\ -I_2 & \ddots & \ddots & 0 \\ & \ddots & \ddots & \ddots \\ & 0 & \ddots & \ddots & -I_2 \\ & & & -I_2 & E \end{pmatrix}_{2m\times 2m}, \qquad m \text{ is even;}$$

where

$$I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \text{ and } E = \begin{bmatrix} 2a & 2b \\ -2b & 2a \end{bmatrix} \quad a > 0.$$

We now consider the convergence of the AGE-1D method when applied to the solution of this system. Let

$$C = \begin{bmatrix} \frac{1}{2}E & -I_2 \\ -I_2 & \frac{1}{2}E \end{bmatrix}_{4x4} = \begin{bmatrix} \begin{pmatrix} a & b \\ -b & a \end{pmatrix} & -I_2 \\ -I_2 & \begin{pmatrix} a & b \\ -b & a \end{pmatrix} \end{bmatrix}_{4x4} \tag{5.49}$$

If A is split as follows:

$$A = G_1 + G_2 \tag{5.50}$$

where,

$$G_1 = \begin{pmatrix} C & & 0 \\ & \ddots & \\ 0 & & \ddots \\ & & C \end{pmatrix}; \text{ and } G_2 = \begin{pmatrix} \frac{1}{2}E & & 0 \\ & C & \\ & & \ddots \\ 0 & & C \\ & & & \frac{1}{2}E \end{pmatrix} \tag{5.51}$$

The AGE algorithm for solving system (5.48) can be written as:

$$\begin{aligned} (G_1 + sI)\mathbf{w}^{p+\frac{1}{2}} &= \mathbf{b} - (G_2 - sI)\mathbf{u}^p \equiv \mathbf{y} \\ \mathbf{w}^{p+1} &= (G_2 + sI)^{-1}[\mathbf{b} - (G_1 - sI)(G_1 + sI)^{-1}\ ] \\ \mathbf{w}^{p+1} &= T_s\mathbf{w}^p + \mathbf{k} \end{aligned} \tag{5.52}$$

where s is the acceleration parameter of the AGE method, and T_s is the iteration matrix given as:

$$T_s = (G_2 + sI)^{-1}(G_1 - sI)(G_1 + sI)^{-1}(G_2 - sI)$$

which can be shown to be similar to :

$$\tilde{T}_s = (G_1 - sI)(G_1 + sI)^{-1}(G_2 - sI)(G_2 + sI)^{-1}$$

we note that:

$$(\mathrm{G_1 - sI}) = \begin{pmatrix} C - sI & & 0 \\ & \ddots & \\ 0 & & \ddots \\ & & C - sI \end{pmatrix}; \text{and } (\mathrm{G_1 + sI}) = \begin{pmatrix} C + sI & & 0 \\ & \ddots & \\ 0 & & \ddots \\ & & C + sI \end{pmatrix}$$

and

$$(\mathrm{G_2 - sI}) = \begin{pmatrix} \frac{1}{2}E - sI & & 0 \\ & C - sI & \\ & & \ddots \\ 0 & & C - sI \\ & & & \frac{1}{2}E - sI \end{pmatrix}$$

; and

$$(\mathrm{G}_2 + \mathrm{sI}) = \begin{pmatrix} \frac{1}{2}E + sI & & & \\ & C + sI & & 0 \\ & & \ddots & \\ & 0 & & \frac{1}{2}E + sI \end{pmatrix}$$

Therefore

$$(G_1 - sI)(G_1 + sI)^{-1} = \begin{pmatrix} D & & & \\ & \ddots & & 0 \\ & & \ddots & \\ & 0 & & D \end{pmatrix}$$

and;

$$(\mathrm{G}_2 - \mathrm{sI})(\mathrm{G}_2 + \mathrm{sI})^{-1} = \begin{pmatrix} F & & & \\ & D & & 0 \\ & & \ddots & \\ & 0 & & D \\ & & & F \end{pmatrix}$$

where $D = (C - sI)(C + sI)^{-1}$ and $F = (\frac{1}{2}E - sI)(\frac{1}{2}E + sI)^{-1}$.

Thus we have the following relations:

$$\rho(T_s) = \rho(\tilde{T}_s) \leq \| (\tilde{T}_s) \|_2 \tag{5.53}$$

$$\Rightarrow \rho(T_s) \leq \| (G_1 - sI)(G_1 + sI)^{-1} \|_2 \| (G_2 - sI)(G_2 + sI)^{-1} \|_2 \tag{5.54}$$

$$= \| D \|_2 \max\{\| D \|_2, \| F \|_2\} \tag{5.55}$$

Let $\frac{1}{2}E = G = \begin{pmatrix} a & b \\ -b & a \end{pmatrix}$

The eigenvalues of G are $a + ib$ and $a - ib$.

Also

$$G\begin{pmatrix} 1 \\ i \end{pmatrix} = (a + ib)\begin{pmatrix} 1 \\ i \end{pmatrix}, \text{ and } \begin{pmatrix} 1 \\ -i \end{pmatrix} = (a - ib)\begin{pmatrix} 1 \\ -i \end{pmatrix},$$

Therefore

$$x = \begin{pmatrix} 1 \\ -i \end{pmatrix} \text{ and } y = \begin{pmatrix} 1 \\ i \end{pmatrix}$$

are the eigenvectors of G. They are orthogonal since

$$\bar{x}^T y = (1\ i)\begin{pmatrix} 1 \\ i \end{pmatrix} = 0$$

Hence, there exists an unitary orthogonal matrix U so that:

$$U^{-1}GU = \begin{pmatrix} a+ib & 0 \\ 0 & a-ib \end{pmatrix} = \Lambda \quad and\ U^H U = I$$

If we let: $Q = \begin{pmatrix} U & 0 \\ 0 & U \end{pmatrix}$ we then have:

$$Q^{-1}DQ = Q^{-1}(C-sI)QQ^{-1}(C+sI)^{-1}Q \tag{5.56}$$

$$= \left\{\begin{bmatrix} \Lambda & -I_2 \\ -I_2 & \Lambda \end{bmatrix} - sI\right\}\left\{\begin{bmatrix} \Lambda & -I_2 \\ -I_2 & \Lambda \end{bmatrix} + sI\right\}^{-1} \tag{5.57}$$

$$= \begin{bmatrix} \Lambda - sI_2 & -I_2 \\ -I_2 & \Lambda - sI_2 \end{bmatrix}\begin{bmatrix} \Lambda + sI_2 & -I_2 \\ -I_2 & \Lambda + sI_2 \end{bmatrix}^{-1} \tag{5.58}$$

Further there is a permutation maxtrix P such that:

$$P\begin{bmatrix} \Lambda \mp sI_2 & -I_2 \\ -I_2 & \Lambda \mp sI_2 \end{bmatrix}P = \begin{bmatrix} \lambda_1^{\mp} & -1 & 0 & 0 \\ -1 & \lambda_1^{\mp} & 0 & 0 \\ 0 & 0 & \lambda_2^{\mp} & -1 \\ 0 & 0 & -1 & \lambda_2^{\mp} \end{bmatrix} \tag{5.59}$$

where

$$\lambda_1^{\mp} = a+ib\mp s = a\mp s+ib$$

$$\lambda_2^{\mp} = a-ib\mp s = a\mp s-ib$$

Again there exists an unitary orthogonal matrix U_1 so that :

$$\begin{bmatrix} U_1 & \\ & U_1 \end{bmatrix}^{-1}\begin{bmatrix} \lambda_1^{\mp} & -1 & 0 & 0 \\ -1 & \lambda_1^{\mp} & 0 & 0 \\ 0 & 0 & \lambda_2^{\mp} & -1 \\ 0 & 0 & -1 & \lambda_2^{\mp} \end{bmatrix}\begin{bmatrix} U_1 & \\ & U_1 \end{bmatrix} = \begin{bmatrix} \lambda_1^{\mp}+1 & 0 & 0 & 0 \\ 0 & \lambda_1^{\mp}-1 & 0 & 0 \\ 0 & 0 & \lambda_2^{\mp}+1 & 0 \\ 0 & 0 & 0 & \lambda_2^{\mp}-1 \end{bmatrix}$$

Hence:

$$\| D \|_2 = \| Q^{-1}DQ \|_2$$

$$= \left\| \begin{bmatrix} \lambda_1^- +1 & & & \\ & \lambda_1^- -1 & & 0 \\ & & \lambda_2^- +1 & \\ & 0 & & \lambda_2^- -1 \end{bmatrix}\begin{bmatrix} \lambda_1^+ +1 & & & \\ 0 & \lambda_1^+ -1 & & 0 \\ 0 & & \lambda_2^+ +1 & \\ & & & \lambda_2^+ -1 \end{bmatrix}^{-1} \right\|_2$$

$$= \max\left\{\left|\frac{\lambda_1^- +1}{\lambda_1^+ +1}\right|, \left|\frac{\lambda_1^- -1}{\lambda_1^+ -1}\right|, \left|\frac{\lambda_2^- +1}{\lambda_2^+ +1}\right|, \left|\frac{\lambda_2^- -1}{\lambda_2^+ -1}\right|\right\}$$

Since

$$\lambda_1^- = a - s + ib \quad ; \lambda_1^+ = a + s + ib$$

$$\lambda_2^- = a - s - ib \quad ; \lambda_2^+ = a + s - ib$$

Therefore

$$\left|\frac{\lambda_1^- + 1}{\lambda_1^+ + 1}\right| = \left[\frac{(a-s+1)^2+b^2}{(a+s+1)^2+b^2}\right]^{\frac{1}{2}} = \left[\frac{(a+1-s)^2+b^2}{(a+1+s)^2+b^2}\right]^{\frac{1}{2}}$$

$$\left|\frac{\lambda_1^- - 1}{\lambda_1^+ - 1}\right| = \left[\frac{(a-s-1)^2+b^2}{(a+s-1)^2+b^2}\right]^{\frac{1}{2}} = \left[\frac{(a-1-s)^2+b^2}{(a-1+s)^2+b^2}\right]^{\frac{1}{2}}$$

$$\left|\frac{\lambda_2^- + 1}{\lambda_2^+ + 1}\right| = \left[\frac{(a-s+1)^2+b^2}{(a+s+1)^2+b^2}\right]^{\frac{1}{2}} = \left[\frac{(a+1-s)^2+b^2}{(a+1+s)^2+b^2}\right]^{\frac{1}{2}}$$

$$\left|\frac{\lambda_2^- - 1}{\lambda_2^+ - 1}\right| = \left[\frac{(a-s-1)^2+b^2}{(a+s-1)^2+b^2}\right]^{\frac{1}{2}} = \left[\frac{(a-1-s)^2+b^2}{(a-1+s)^2+b^2}\right]^{\frac{1}{2}}$$

we also have

$$\| F \|_2 = \sqrt{\frac{(a-s)^2+b^2}{(a+s)^2+b^2}} \quad < 1 \quad \text{for every } s > 0 \tag{5.60}$$

Thus it can be seen from (5.55) that:

$$\rho(T_s) \leq \max\left\{\left[\frac{(a-1-s)^2+b^2}{(a-1+s)^2+b^2}\right]^{\frac{1}{2}}, \left[\frac{(a+1-s)^2+b^2}{(a+1+s)^2+b^2}\right]^{\frac{1}{2}}\right\}$$

$$\times \max\left\{\left[\frac{(a-1-s)^2+b^2}{(a-1+s)^2+b^2}\right]^{\frac{1}{2}}, \left[\frac{(a-s)^2+b^2}{(a+s)^2+b^2}\right]^{\frac{1}{2}}, \left[\frac{(a+1-s)^2+b^2}{(a+1+s)^2+b^2}\right]^{\frac{1}{2}}\right\} \tag{5.61}$$

Hence

$$\rho(T_s) \leq 1 \quad if\ a \geq 1 \qquad for\ any\ s > 0 \tag{5.62}$$

The experimental results given in section 7.2 indicate that the inequality in (5.62) is strict and that for such block symmetric systems the method is convergent for any $s > 0$.

5.6 Chebyshev acceleration of the AGE-1D method.

In this section we consider whether or not in theory, the convergence of the AGE-1D iterative procedure may be accelerated by using the Chebyshev polynomials, when AGE-1D is applied for one dimensional problems or as component of the *Explicit Alternating Direction (EAD)* method (see chapter 6) method for multidimensional problems.

We therefore consider here the AGE-1D method for solving the system:

$$A_1\mathbf{x} = \mathbf{b}_1 \tag{5.63}$$

where

$$A_1 = \begin{bmatrix} e & -c & & & \\ -c & e & -c & 0 & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -c \\ & 0 & & -c & e \end{bmatrix}_{m\times m} \quad ; \; m \; is \; even,$$

or alternately the normalized equivalent system:

$$A\mathbf{x} = \mathbf{b} \tag{5.64}$$

with

$$A = \begin{bmatrix} a & -1 & & & \\ -1 & a & -1 & 0 & \\ & \ddots & \ddots & \ddots & \\ & 0 & \ddots & \ddots & -1 \\ & & & -1 & a \end{bmatrix} \quad ; \; a = \frac{e}{c}$$

System (5.63) or (5.64) are the type of systems which arise from two point boundary value problems, 1D parabolic heat conduction problems, and also as a

subsystem in higher dimensional parabolic (heat conduction) and elliptic (Laplace, Poisson) problems, where the AGE-1D method may be applied as a component of the EAD method.

If we apply the AGE-1D method for solving (5.64), then we have:

$$\mathbf{x}^{p+1} = T_s \mathbf{x}^p + \mathbf{b}' \tag{5.65}$$

where $\mathbf{b}' = 2s(G_2 + sI)^{-1}(G_1 + sI)^{-1}\,\mathbf{b}$, and

$$T_s = (G_2 + sI)^{-1}(G_1 - sI)(G_1 + sI)^{-1}(G_2 - sI)$$

is the iteration matrix. G_1 and G_2 are given as:

$$G_1 = \begin{bmatrix} C_1 & & & \\ & \ddots & & 0 \\ & & \ddots & \\ & 0 & & C_1 \end{bmatrix} \quad and \quad G_1 = \begin{bmatrix} \frac{a}{2} & & & \\ & C_1 & & 0 \\ & & \ddots & \\ & 0 & & C_1 \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \frac{a}{2} \end{matrix}$$

with $C_1 = \begin{pmatrix} \frac{a}{2} & -1 \\ -1 & \frac{a}{2} \end{pmatrix}$.

To accelerate the convergence of the AGE-1D method by using the Chebyshev polynomials it is required that the eigenvalues of the iteration matrix T_s be real. We now consider the conditions posed by this requirement.

The matrix T_s is similar to $F_1.F_2$ where

$$F_1 = (G_1 - sI)(G_1 + sI)^{-1} \quad ; \quad F_2 = (G_2 - sI)(G_2 + sI)^{-1}$$

Therefore T_s and $F_1.F_2$ have the same eigenvalues.

$$Let \;\; \hat{C} = (C_1 - sI)(C_1 + sI)^{-1} = \frac{1}{(\frac{a}{2} + s)^2 - 1} \begin{bmatrix} (\frac{a}{2})^2 - s^2 - 1 & -2s \\ -2s & \frac{a}{2})^2 - s^2 - 1 \end{bmatrix}$$

$$= \frac{1}{\lambda\mu} \begin{bmatrix} (\frac{a}{2})^2 - s^2 - 1 & -2s \\ -2s & \frac{a}{2})^2 - s^2 - 1 \end{bmatrix}$$

where $\lambda = \frac{a}{2} + s + 1$ and $\mu = \frac{a}{2} + s - 1$.

Also we define the matrices $E = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$, *and* $D = \begin{bmatrix} E & & \\ & \ddots & \\ 0 & & \ddots \\ & & & E \end{bmatrix}$

We now have

$$\hat{C}E = \frac{1}{\lambda\mu}\begin{bmatrix} \alpha\lambda & \beta\mu \\ \alpha\lambda & -\beta\mu \end{bmatrix} \quad where \quad \begin{matrix} \alpha = \frac{a}{2} - s - 1 \\ \beta = \frac{a}{2} - s + 1 \end{matrix}$$

$$and \quad E\begin{bmatrix} y & 0 \\ 0 & z \end{bmatrix} = \frac{1}{\lambda\mu}\begin{bmatrix} \alpha\lambda & \beta\mu \\ \alpha\lambda & -\beta\mu \end{bmatrix} \quad where\ y = \frac{\alpha}{\mu}\ ;\ z = \frac{\beta}{\lambda}$$

We therefore have:

$$F_1 D = D\Lambda \Rightarrow \ F_1 = D\Lambda D^{-1}$$

where

$$\Lambda = \begin{bmatrix} y & & & \\ & z & & 0 \\ & & \ddots & \\ & 0 & & y \\ & & & & z \end{bmatrix}$$

If, and only if:

$$y > 0 \tag{5.66}$$

$$z > 0 \tag{5.67}$$

then the matrix

$$\Lambda^{\frac{1}{2}} = \begin{bmatrix} \sqrt{y} & & & \\ & \sqrt{z} & & 0 \\ & & \ddots & \\ & 0 & & \sqrt{y} \\ & & & & \sqrt{z} \end{bmatrix}$$

is real positive definite.

We then have $F_1 = D\Lambda^{\frac{1}{2}}\Lambda^{\frac{1}{2}}D^{-1} = (D\Lambda^{\frac{1}{2}}D^{-1})(D\Lambda^{\frac{1}{2}}D^{-1}) = F_1^{\frac{1}{2}}F_1^{\frac{1}{2}}$.

Under the conditions of (5.66) and (5.67) F_1 and $F_1^{\frac{1}{2}}$ are not singular. Therefore,

$$F_1F_2 = F_1^{\frac{1}{2}}F_1^{\frac{1}{2}}F_2 = F_1^{\frac{1}{2}}(F_1^{\frac{1}{2}}F_2F_1^{\frac{1}{2}})F_1^{\frac{-1}{2}}$$

Thus, F_1F_2 and $(F_1^{\frac{1}{2}}F_2F_1^{\frac{1}{2}})$ has the same eigenvalues. It can be easily seen that $(F_1^{\frac{1}{2}}F_2F_1^{\frac{1}{2}})$ is symmetric and therefore has only real eigenvalues. Hence under the conditions given by the inequalities (5.66) and (5.67) the requirement that all the eigenvalues of T_s be real is satisfied.

However the above conditions are very restrictive, and therefore using the Chebyshev polynomial to accelerate the AGE-1D method may not be practical for the above stated problem.

5.7 The various forms of the AGE method and the computational requirements

There exist various forms for the formulation of the AGE-1D method some of which were referred to earlier, e.g the usual formulation of (4.58), (4.59) and the composite form of (4.60). The computational cost of some of these formulations are less than others. The computational cost also is usually less for normalized systems such as (4.54) than for non-normalized systems such as (4.53). The various forms of the AGE-PR method for solving these systems are listed together with their computational cost *per iteration* in the table below. But first some of the variants not given earlier are introduced next.

To start with we rewrite here the above mentioned two formulations which are referred to as Formulation I and Formulation II respectively.

Formulation I

$$\begin{aligned}(G_1 + sI)\mathbf{u}^{p+1/2} &= \mathbf{f} + (G_2 - sI)\mathbf{u}^p = \mathbf{z_1} \\ (G_2 + sI)\mathbf{u}^{p+1} &= \mathbf{f} + (G_1 - sI)\mathbf{u}^{p+1/2} = \mathbf{z_2}\end{aligned}$$

Formulation II

$$\mathbf{u}^{p+1} = G\mathbf{u}^p + \mathbf{k}$$

For this formulation we note that the vector k is given by

$$\mathbf{k} = 2s(G_2 + sI)^{-1}(G_1 + sI)^{-1}\mathbf{f}$$

and is calculated once for each system (i.e not every iteration) at the cost of (4m-4) multiplications and (3m-4) additions. and G is the iteration matrix of the method given as:

$$G = (G_2 + sI)^{-1}(G_1 - sI)(G_1 + sI)^{-1}(G_2 - sI) \tag{5.68}$$

The iteration matrix G has the following form:

$$G = \begin{bmatrix}
a' & b' & c' & d' & & & & & & \\
e' & f' & g' & h' & & & & & & \\
h' & g' & i' & j' & c' & d' & & & & \\
d' & c' & j' & i' & g' & h' & & & & \\
 & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & & \\
 & & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \\
 & & & & h' & g' & i' & j' & c' & d' \\
 & & & & d' & c' & j' & i' & g' & h' \\
 & & & & & & h' & g' & f' & e' \\
 & & & & & & d' & c' & b' & a'
\end{bmatrix}$$

where $a' \ldots j'$ are terms obtained after the product of the matrices given in equation 5.68 is carried out.

Formulation III

$$\begin{aligned}
(G_1 + sI)\mathbf{u}^{p+1/2} &= \mathbf{f} + (G_2 - sI)\mathbf{u}^p = \mathbf{z}_1 \\
(G_2 + sI)\mathbf{u}^{p+1} &= (G_2 - sI)\mathbf{u}^p + 2s\mathbf{u}^{p+1/2} = \mathbf{z}_2
\end{aligned}$$

This variant of the AGE-PR method can be easily derived by manipulating the second equation of Formulation I. It is less costly in its computational work.

Formulation IV

$$\begin{aligned}(G_1 + sI)\mathbf{u}^{p+1/2} &= \mathbf{f} + (G_2 - sI)\mathbf{u}^p = \mathbf{z_1} \\ (G_2 + sI)\mathbf{u}^{p+1} &= \mathbf{f} + (G_1 - sI)\underbrace{(G_1 + sI)^{-1}\mathbf{z_1}}_{\mathbf{u}^{p+1/2}}\end{aligned}$$

This formulation is due to Evans and Li [16]. Here the half iterate of the solution is never obtained but rather only the right hand side $\mathbf{z_1}$ of the first equation is obtained and used in computing the full iterate solution in the second equation.

By this formulation the solution may be obtained at a lower computational cost per iteration than by any of the above formulations. The second equation of this formulation is computed according to [16] as:

$$\mathbf{u}^{p+1} = \mathbf{f}' + \widehat{(G_2 + sI)}\,\underbrace{\Delta\,[(G_1 - sI)(G_1 + sI)^{-1}\mathbf{z_1}]}\qquad (5.69)$$

where Δ is the determinent of the 2x2 submatrices of $(G_2 + sI)$ and is as given in (4.63), and :

$$\widehat{(G_2 + sI)} = \Delta(G_2 + sI)^{-1}$$

the underbrace indicates the term which is to be calculated first. The vector $\mathbf{f}'$ needs to be computed and stored. It is given as:

$$\mathbf{f}' = (G_2 + sI)^{-1}\mathbf{f}$$

which has to be calculated once only and not every iteration. It is computed at the cost of $2m$ multiplications and m additions.

A slight improvement can still be obtained if the second equation is computed as:

$$\mathbf{u}^{p+1} = \widehat{(G_2 + sI)}\,[\mathbf{f}' + \Delta(G_1 - sI)(G_1 + sI)^{-1}\mathbf{z_1}]\qquad (5.70)$$

where $\widehat{\mathbf{f}'}$ is now defined as:

$$\widehat{\mathbf{f}'} = \Delta\mathbf{f}$$

The AGE-1D variant	Cost[a]	Normalized system: No. of +/- op. per iteration	Normalized system: No. of × op. per iteration	Non-normalized system: No. of +/- op. per iteration	Non-normalized system: No. of × op. per iteration	storage requirement
Formulation I		6m-4	6m	6m-4	8m	5 vectors
Formulation II		5m-8	6m-8	5m-8	6m-8	3 vectors
Formulation III		5m-4	6m	5m-4	7m	5 vectors
Formulation IV		5m-4	4m	5m-4	6m-4	5 vectors

[a]The figures given are for systems of order m where m is supposed even. They are almost identical to the case when m is odd

Table 5.2: This table shows the computational requirements when using any of the four main variants of the AGE-1D for solving tridiagonal systems which have off diagonal elements of identical modulus.

This cuts down the cost of computing f′ by m multiplications and m additions at no other extra cost.

Table 5.2 shows the computational work per iteration and the storage requirements of each of the variants of the AGE-1D algorithm for normalized systems and non-normalized systems.

Chapter 6

The Explicit Alternating Direction methods (EAD)

6.1 Introduction

For many real time problems, parallelism is the only way forward for obtaining any dramatic progress. A high degree of parallelism - defined as the number of independent operations that may be performed simultaneously - must persist throughout the different stages of solving any particular problem. These stages proceed in sequence from choosing the suitable algorithm, to expressing the algorithm in a high level language, compiling the language into a machine-readable object code, and finally executing the code on the target machine. Ideally the parallelism at any of these stages should be greater or equal to that in the subsequent stages, if the parallelism of the target computer is to be fully exploited. This places a greater emphasis on the parallelism of the first stage, namely the choice of a parallel algorithm.

The Alternating Group Explicit (AGE) iterative method was an important step in this context. However, it required, especially for two or more dimensional problems, a considerable greater computational work per iteration and an even larger number of iterations to converge. This meant that the advantages of the parallelism of the method should outweigh the disadvantages only when the systems being solved are

of a very large order.

The Explicit Alternating Direction (EAD) methods presented in this chapter, represent further developments on the AGE method for two or more dimensional problems, which require much less computational work than the the corresponding AGE-2D or AGE-3D methods and are even faster to converge, thus also improving on the parallelism of the AGE-2D and AGE-3D methods. The EAD methods are based on combining the use of the ADI techniques with the AGE-1D algorithm. The resulting EAD method will be referred to as an EAD *fully iterative* method if the AGE method is combined with an ADI iterative method. Otherwise if the use of the AGE method is combined with an ADI direct scheme, the resulting method is then referred to simply as the EAD method (see figure 6.1). Like the AGE method, the main positive features to be stressed in the formulation of the EAD methods is that it is an *explicit* algorithm for solving what is actually an implicit system, thus allowing maximum parallelism in its application. Another not important advantage is that the intermediary boundary values which have to be obtained for the standard ADI schemes are not required when the EAD *fully iterative* method is applied. The methods are introduced next, and a comparison between the EAD methods and the AGE methods is presented.

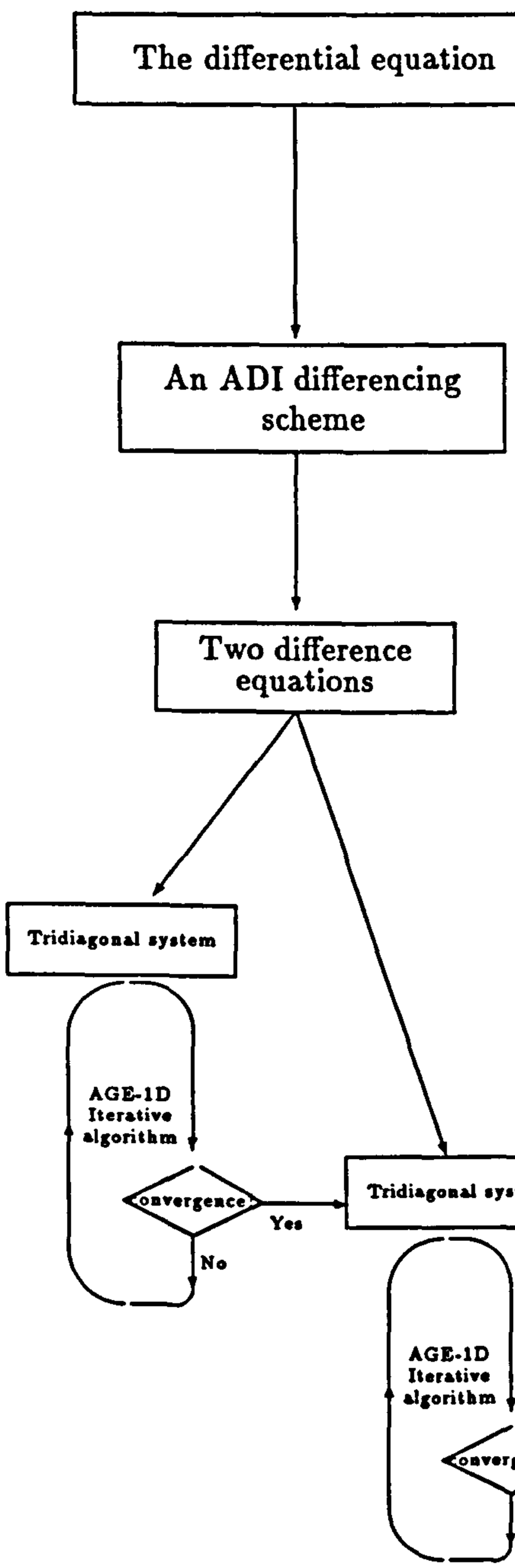
6.2 The EAD method.

6.2.1 A parabolic problem

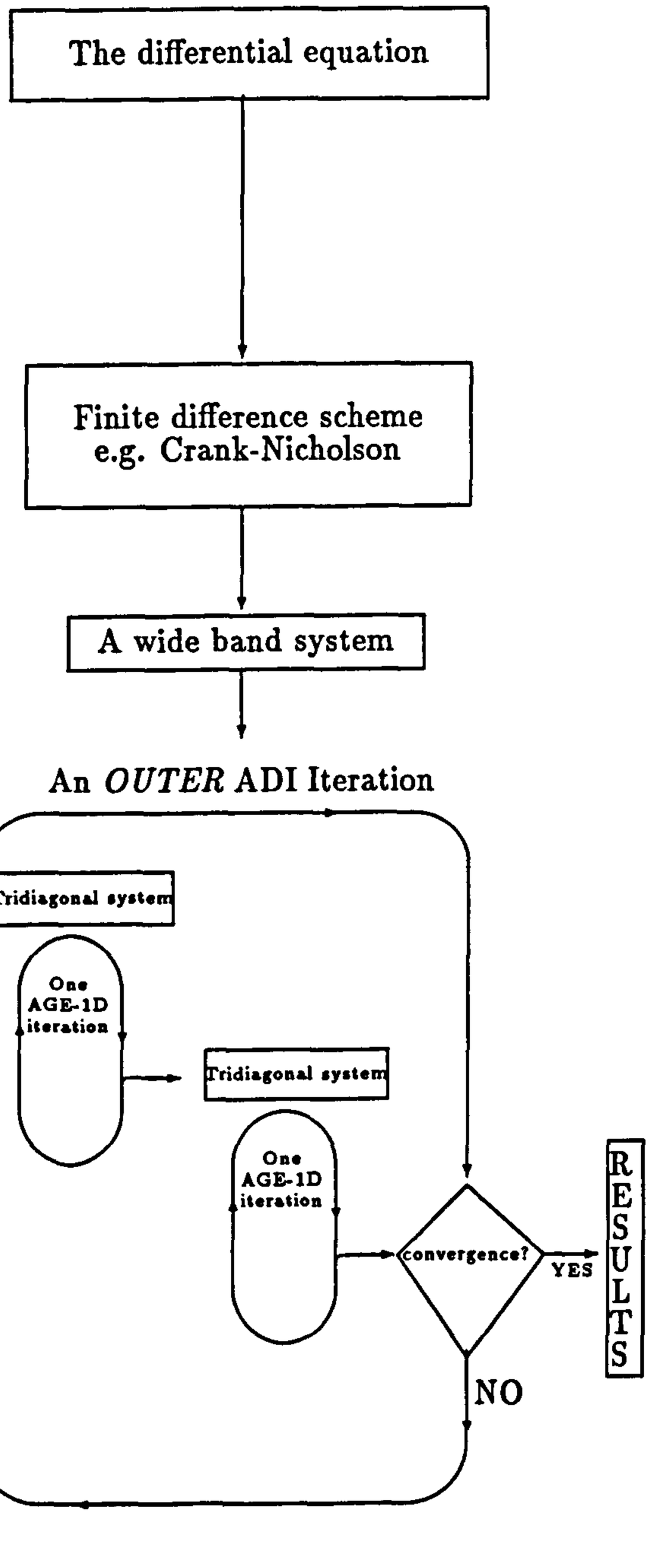
In this section the EAD method (fig. 6.1-left) is presented by solving the two dimensional heat problem given by equation (4.91) together with the auxiliary conditions as presented in subsection 4.5. We also use the same mesh described thereof.

The following ADI-PR and ADI-DR schemes can be used to approximate equation (4.91). These are given respectively by:

$$(1-\frac{1}{2}r\delta_{x_1}^2)U_{i,j}^{n*} \;=\; (1+\frac{1}{2}r\delta_{x_2}^2)U_{i,j}^{n}+\frac{1}{2}kq(x_1,x_2,t+\frac{1}{2})$$



The EAD direct method or
The EAD method



The EAD fully Iterative method

Figure 6.1: A schematic representation of the EAD techniques.

$$(1-\frac{1}{2}r\delta_{x_2}^2)U_{i,j}^{n+1} = (1+\frac{1}{2}r\delta_{x_1}^2)U_{i,j}^{n*} + \frac{1}{2}kq(x_1,x_2,t+\frac{1}{2}) \quad \textbf{(PR)} \qquad (6.1)$$

and

$$\begin{aligned}(1-r\delta_{x_1}^2)U_{i,j}^{n*} &= (1+r\delta_{x_2}^2)U_{i,j}^{n} + q(x_1,x_2,t+\frac{1}{2}) \\ (1-r\delta_{x_2}^2)U_{i,j}^{n+1} &= U_{i,j}^{n*} - r\delta_{x_2}^2 U_{i,j}^{n} \qquad \textbf{(DR)}\end{aligned} \qquad (6.2)$$

When applied to the totality of the meshpoints, equations (6.1) and (6.2) give rise to two systems given by (4.25) and (4.26) which are to be solved *in sequence.* If these systems were normalized (i.e. divided by $\frac{1}{2}r$ for the PR scheme, or by r for the DR scheme) we then get the following equivalent systems:

$$H_1\mathbf{u} = \mathbf{d}_1 \qquad (6.3)$$

and

$$V_1\mathbf{u} = \mathbf{d}_2 \qquad (6.4)$$

where H_1 and V_1 has the same form as H and V of equations (4.25) and (4.26) but with:

$$a_1 = a_2 = -1 \quad \textit{and} \quad b = \frac{(1+2\theta r)}{(\theta r)} = 2 + \frac{1}{\theta r}$$

where $\theta = \frac{1}{2}$ and 1, for the PR and DR schemes respectively. Each of the above systems is then solved using the AGE-1D iterative method, as applied to one dimensional problems. This is given using the economic Formulation IV of the algorithm as:

$$\begin{aligned}(G_1+sI)\mathbf{u}^{p+1/2} &= \mathbf{d}_1 - (G_2-sI)\mathbf{u}^{p} \equiv \mathbf{z}_1 \\ (G_2+sI)\mathbf{u}^{p+1} &= [\mathbf{d}_1 - (G_1-sI)(G_1+sI)^{-1}\mathbf{z}_1] \equiv \mathbf{z}_2\end{aligned} \qquad (6.5)$$

where s is the AGE-1D acceleration parameter. we use this algorithm, at the first level, to solve the system (6.3) whereby we iterate until the solution vector is obtained. This is then used in evaluating the RHS of equation (6.4) as well as a

starting vector when applying the AGE algorithm again, at the second level to solve (6.4). At the second level the AGE algorithm is again given by (6.5), but by replacing G_1, G_2, and $\mathbf{d_1}$ by G_3, G_4, and $\mathbf{d_2}$ respectively.

The two pairs of matrices G_1, G_2 and G_3, G_4 are the constituent matrices into which H_1 and V_1 are split respectively. They have exactly the same structure as given in subsection 4.5.

To compare the computational costs of the AGE-2D method and the EAD method we note the following: On one hand the AGE-2D algorithm for solving system (4.93) requires $(16mn-2m-4n)$ multiplications and $(15mn-2m-8n)$ additions for each full AGE-2D iteration. The calculation of the R.H.S of system (4.93) itself requires mn multiplications and $4mn+2(m+n)$ additions if a Crank-Nicholson (CN) scheme is used, and just mn multiplications and $2(m+n)$ additions if the *fully implicit* scheme is used. This is if we assume the source term q is zero. On the other hand the above AGE-1D algorithm used in the EAD method at each ADI level requires $4mn$ multiplications and $5mn-4n$ additions per iteration. The calculation of the R.H.Ss of equations (6.3) and (6.4) require $2mn$ multiplications and $4mn+4n$ additions if the Peaceman Rachford scheme is used, and $3mn$ multiplications and $5mn+4n$ additions if the DR scheme is used. We also require for the EAD method to calculate the intermediate values of the solution at the two boundaries parallel to the x_2 direction. This is done by the appropriate formulae given in section 4.1 at a trivial cost of $6n$ multiplications and $10n$ additions for either of the PR or DR schemes. If the source/sink term is not zero then an additional cost of evaluating the R.H.Ss of equation (4.93) and equations (6.3) and (6.4) is incurred.

Experiments were carried out for the heat conduction problem given above, with and without a heat source. Comparisons are made between, on one hand, the AGE-2D method as applied to the fully implicit scheme and the Crank-Nicholson scheme, and on the other hand the EAD method using the (ADI) PR and DR schemes. The results are given, for the case without a heat source in *tables* 6.1...6.4, and for the case of a heat source in *tables* 6.5... 6.10. The figures for the total number of +/* operations given in the tables, include the evaluation of R.H.S vector/s in each case.

a) $x_2 = .02$; *mesh ratio* = 0.5, $\Delta x_1 = \Delta x_2 = .02$, $\Delta t = .0002$, $\epsilon = 10^{-4}$

	$x_1 =$	.02	.30	.58	.86	1.14	1.42	1.70	1.98
scheme									
IMP	EAD-LOD	3.3E-06	2.2E-05	1.1E-05	1.8E-05	1.7E-05	1.1E-05	2.2E-05	3.4E-06
	EAD-DR	1.8E-06	2.0E-05	7.4E-06	1.7E-05	1.4E-05	1.2E-05	1.8E-05	4.4E-06
	AGE-2D	5.2E-06	2.0E-05	1.6E-05	1.4E-05	2.1E-05	6.2E-06	2.4E-05	1.2E-06
CN	EAD	8.1E-07	5.3E-06	2.7E-06	4.3E-06	4.3E-06	2.7E-06	5.3E-06	8.3E-07
	AGE-2D	1.4E-06	9.9E-06	5.3E-06	7.9E-06	8.2E-06	4.8E-06	1.0E-05	1.3E-06
Exact solution		.007201	.054642	-.027678	-.044269	.044269	.027678	-.054642	-.007201

b) $x_2 = .50$; *mesh ratio* = 0.5, $\Delta x_1 = \Delta x_2 = .02$, $\Delta t = .0002$, $\epsilon = 10^{-4}$

	$x_1 =$	.02	.30	.58	.86	1.14	1.42	1.70	1.98
scheme									
IMP	EAD-LOD	4.9E-05	.3.3E-04	1.6E-04	2.7E-04	2.7E-04	1.7E-04	3.3E-04	5.3E-05
	EAD-DR	3.8E-05	3.0E-04	1.4E-04	2.5E-04	2.3E-04	1.7E-04	2.9E-04	6.1E-05
	AGE-2D	6.7E-05	5.3E-04	2.6E-04	4.4E-04	4.2E-04	2.8E-04	3.9E-04	6.9E-05
CN	EAD	1.3E-05	8.2E-05	4.1E-05	6.6E-05	6.6E-05	4.1E-05	8.1E-05	1.3E-05
	AGE-2D	2.2E-05	1.7E-04	8.3E-05	1.3E-04	1.3E-04	8.5E-05	1.6E-04	2.4E-05
Exact solution		.114680	.870221	-.440807	-.705023	.705023	.440807	-.870221	-.114680

c) $x_2 = .98$; *mesh ratio* = 0.5, $\Delta x_1 = \Delta x_2 = .02$, $\Delta t = .0002$, $\epsilon = 10^{-4}$

	$x_1 =$	.02	.30	.58	.86	1.14	1.42	1.70	1.98
scheme									
IMP	EAD-LOD	3.9E-06	2.8E-05	1.4E-05	2.2E-05	2.2E-05	1.4E-05	2.7E-05	4.3E-06
	EAD-DR	3.1E-06	1.9E-05	9.8E-06	1.5E-05	1.5E-05	9.2E-06	1.9E-05	3.2E-05
	AGE-2D	4.0E-06	5.4E-05	1.9E-05	4.6E-05	3.7E-05	3.3E-05	4.9E-05	1.1E-05
CN	EAD	1.1E-06	7.5E-06	3.8E-06	6.1E-06	6.1E-06	3.8E-06	7.5E-06	1.1E-06
	AGE-2D	1.4E-06	1.2E-05	5.5E-06	9.7E-06	9.2E-06	6.3E-06	1.1E-05	1.8E-06
Exact solution		.007201	.054642	-.027678	-.044269	.044269	.027678	-.054642	-.007201

Table 6.1: The absolute errors of the solutions to the model 2D heat problem, as obtained by the AGE-2D method and the EAD method, at $t = .0018$.

Method	IMP			CN	
	EAD-LOD	EAD-DR	AGE-2D	EAD-PR	AGE-2D
Average of all absolute errors	8.56E-05	7.6E-05	2.33E-04	2.13E-05	7.26E-05
Number of iterations	3/2	3/1	4	2/2	4
No. (in mxn) of + & * operations	25+; 20*	20+; 16*	60+; 76*	20+; 16*	60+; 76*
Total (in mxn) of +/* operations	47	44	137	42	141
Computational cost w.r.t. AGE-2D	34 %	32 %	100 %	30 %	100 %
No. of points of synchronization	17	14	33	14	33

Table 6.2: Average of absolute errors, number of iterations, and the computational work involved in the experiments of table 6.1.

a) $x_2 = .02$; *mesh ratio* = 1.0, $\Delta x_1 = \Delta x_2 = .02$, $\Delta t = .0004$, $\epsilon = 10^{-4}$

	x_1 =	.02	.30	.58	.86	1.14	1.42	1.70	1.98
scheme									
IMP	EAD-LOD	1.1E-05	6.5E-05	3.5E-05	5.2E-05	5.4E-05	3.2E-05	6.6E-05	1.0E-05
	EAD-DR	1.5E-05	5.5E-05	3.0E-05	4.2E-05	4.6E-05	2.4E-05	5.3E-05	3.5E-05
	AGE-2D	3.8E-05	1.2E-04	8.6E-05	1.6E-04	2.7E-05	1.7E-04	3.6E-05	5.6E-05
CN	EAD	1.1E-06	5.8E-06	3.4E-06	4.5E-06	5.1E-06	2.6E-06	6.1E-06	8.5E-07
	AGE-2D	3.9E-06	4.7E-06	1.13E-05	4.2E-07	1.1E-05	3.9E-06	1.0E-05	1.2E-06
Exact solution		.006589	.049997	-.025326	-.040506	.040506	.025326	-.049997	-.006589

b) $x_2 = .50$; *mesh ratio* = 1.0, $\Delta x_1 = \Delta x_2 = .02$, $\Delta t = .0004$, $\epsilon = 10^{-4}$

	x_1 =	.02	.30	.58	.86	1.14	1.42	1.70	1.98
scheme									
IMP	EAD-LOD	1.6E-04	9.9E-04	4.9E-04	8.1E-04	7.9E-04	5.1E-04	9.8E-04	1.7E-04
	EAD-DR	2.2E-04	9.2E-04	4.1E-04	7.3E-04	6.8E-04	4.8E-04	8.3E-04	5.4E-04
	AGE-2D	1.9E-04	1.5E-03	6.8E-04	1.2E-03	1.1E-03	8.2E-04	1.5E-03	3.2E-04
CN	EAD	2.1E-05	1.3E-04	6.8E-05	1.0E-04	1.1E-04	6.4E-05	1.3E-04	2.0E-05
	AGE-2D	3.5E-05	2.8E-04	1.3E-04	2.3E-04	2.1E-04	1.5E-05	2.7E-04	5.3E-05
Exact solution		.104933	.796256	-.403340	-.645099	.645099	.403340	-.796256	-.104933

c) $x_2 = .98$; *mesh ratio* = 1.0, $\Delta x_1 = \Delta x_2 = .02$, $\Delta t = .0004$, $\epsilon = 10^{-4}$

	x_1 =	.02	.30	.58	.86	1.14	1.42	1.70	1.98
scheme									
IMP	EAD-LOD	1.4E-05	1.0E-04	4.8E-05	8.5E-05	8.0E-05	5.5E-05	1.0E-04	1.7E-05
	EAD-DR	1.4E-05	5.9E-05	2.0E-05	4.9E-05	3.9E-05	3.5E-05	5.0E-04	3.3E-05
	AGE-2D	8.9E-06	3.7E-04	2.6E-05	3.6E-04	1.6E-04	3.0E-04	2.7E-04	1.1E-04
CN	EAD	4.1E-06	3.0E-05	1.5E-05	2.4E-05	2.4E-05	1.5E-05	3.0E-05	4.2E-06
	AGE-2D	1.4E-06	3.8E-05	8.2E-06	3.5E-05	2.1E-05	2.7E-05	3.1E-05	9.3E-06
Exact solution		.006589	.049997	-.025326	-.040506	.040506	.025326	-.049997	-.006589

Table 6.3: The absolute errors of the solutions to the model 2D conduction problem, as obtained by the AGE-2D method and the EAD method, at $t = 0.0036$

Method	IMP			CN	
	EAD-LOD	EAD-DR	AGE-2D	EAD-PR	AGE-2D
Average of all absolute errors	2.61E-04	2.29E-04	6.98E-04	3.77E-05	1.21E-04
Number of iterations	4/2	4/1	4	3/2	4
No. (in mxn) of + & * operations	30+; 24*	25+; 20*	60+; 76*	25+; 20*	60+; 76*
Total (in mxn) of +/* operations	56	53	137	51	141
Relative cost w.r.t. AGE-2D	41 %	39 %	100 %	36 %	100 %
No. of points of synchronization	20	17	33	17	33

Table 6.4: Average of absolute errors, number of iterations, and the computational work involved in the experiments of table 6.3.

a) $x_1 = .1$; *mesh ratio* $= .1$, $\Delta x_1 = \Delta x_2 = 0.1$, $\Delta t = 0.001$, $\epsilon = 10^{-4}$

	$x_2 =$	.1	.2	.3	.4	.5	.6	.7	.8	.9
scheme										
IMP	EAD-LOD	1.3E-06	3.2E-06	3.9E-06	5.3E-06	5.6E-06	6.5E-06	5.5E-06	5.6E-06	2.2E-06
	EAD-DR	2.0E-06	3.9E-06	5.6E-06	7.1E-06	8.1E-06	8.7E-06	8.4E-06	7.1E-06	4.4E-06
	AGE-2D	2.0E-05	3.9E-05	5.7E-05	7.3E-05	8.4E-05	9.0E-05	8.9E-05	7.6E-05	4.8E-05
CN	EAD	1.2E-06	2.4E-06	3.5E-06	4.4E-06	5.1E-06	5.4E-06	5.2E-06	4.4E-06	2.7E-06
	AGE-2D	1.7E-05	3.4E-05	5.0E-05	6.3E-05	7.2E-05	7.7E-05	7.4E-05	6.2E-05	3.8E-05
Exact solution		.029018	0.067946	0.126695	0.205177	0.303308	0.421006	0.558194	0.714801	0.890760

b) $x_1 = .50$; *mesh ratio* $= .1$, $\Delta x_1 = \Delta x_2 = 0.1$, $\Delta t = 0.001$, $\epsilon = 10^{-4}$

	$x_2 =$	.1	.2	.3	.4	.5	.6	.7	.8	.9
scheme										
IMP	EAD-LOD	4.6E-06	9.8E-06	1.4E-05	1.8E-05	2.1E-05	2.3E-05	2.2E-05	1.9E-05	1.1E-05
	EAD-DR	8.1E-06	1.6E-05	2.3E-05	2.9E-05	3.4E-05	3.6E-05	3.6E-05	3.0E-05	1.9E-05
	AGE-2D	8.4E-05	1.66E-04	2.4E-04	3.1E-04	3.6E-04	3.9E-04	3.4E-04	3.3E-04	2.15E-04
CN	EAD	5.1E-06	1.0E-05	1.5E-05	1.8E-05	2.1E-05	2.3E-05	2.2E-05	1.9E-05	1.2E-05
	AGE-2D	7.2E-05	1.4E-04	2.1E-04	2.6E-04	3.0E-04	3.2E-05	3.2E-04	2.7E-04	1.6E-04
Exact solution		0.303308	0.376183	0.468197	0.578931	0.707976	0.854943	1.019463	1.201191	1.399809

c) $x_1 = .9$; *mesh ratio* $= .1$, $\Delta x_1 = \Delta x_2 = 0.1$, $\Delta t = 0.001$, $\epsilon = 10^{-4}$

	$x_2 =$	.1	.2	.3	.4	.5	.6	.7	.8	.9
scheme										
IMP	EAD-LOD	3.2E-06	5.9E-05	8.6E-06	1.1E-05	1.3E-05	1.4E-05	1.4E-05	1.3E-05	8.3E-06
	EAD-DR	4.4E-06	8.6E-06	1.3E-05	1.6E-05	1.9E-05	2.1E-05	2.2E-05	2.0E-05	1.4E-05
	AGE-2D	4.8E-05	9.6E-05	1.4E-04	1.8E-04	2.1E-04	2.4E-04	2.5E-04	2.3E-04	1.6E-04
CN	EAD	2.7E-06	5.4E-06	7.9E-06	1.0E-05	1.2E-05	1.3E-05	1.3E-05	1.2E-05	8.4E-06
	AGE-2D	3.8E-05	7.4E-05	1.1E-04	1.4E-04	1.6E-04	1.8E-04	1.8E-04	1.6E-04	1.1E-04
Exact solution		0.890760	0.990813	1.109460	1.246013	1.399809	1.570209	1.756611	1.958450	2.175209

Table 6.5: The absolute errors of the solutions to the model 2D heat problem, WITH A HEAT SOURCE as obtained by the AGE-2D method and the EAD method, at $t = 0.1$

Method	IMP			CN	
	EAD-LOD	EAD-DR	AGE-2D	EAD-PR	AGE-2D
Average of all absolute errors	1.01E-05	1.57E-05	2.13E-04	9.84E-06	1.74E-04
Number of iterations	2/2	2/1	2	2/2	3
No. (in mxn) of + & * operations	20+; 16*	15+; 12*	30+; 38*	20+; 16*	45+; 57*
Total (in mxn) of +/* operations	38	35	69	42	107
Relative cost w.r.t. AGE-2D	55 %	51 %	100 %	39 %	100 %
No. of points of synchronisation	14	11	17	14	22

Table 6.6: Average of absolute errors, number of iterations, and the computational work involved in the experiments of table 6.5.

a) $x_1 = .1$; *mesh ratio* $= .1$, $\Delta x_1 = \Delta x_2 = 0.1$, $\Delta t = 0.001$, $\epsilon = 10^{-8}$

	$x_2 =$	.1	.2	.3	.4	.5	.6	.7	.8	.9
scheme										
IMP	EAD-LOD	1.2E-06	2.4E-06	3.5E-06	4.4E-06	5.1E-06	5.4E-06	5.2E-06	4.4E-06	2.7E-06
	EAD-DR	2.0E-06	3.8E-06	5.6E-06	7.0E-06	8.1E-06	8.6E-06	8.4E-06	7.0E-06	4.4E-06
	AGE-2D	2.0E-06	3.9E-06	5.6E-06	7.0E-06	8.1E-06	8.6E-06	8.4E-06	7.1E-06	4.4E-06
CN	EAD	1.2E-06	2.4E-06	3.5E-06	4.4E-06	5.1E-06	5.4E-06	5.2E-06	4.4E-06	2.7E-06
	AGE-2D	1.2E-06	2.4E-06	3.5E-06	4.4E-06	5.1E-06	5.4E-06	5.2E-06	4.4E-06	2.7E-06
Exact solution		.029018	0.067946	0.126695	0.205177	0.303308	0.421006	0.558194	0.714801	0.890760

b) $x_1 = .50$; *mesh ratio* $= .1$, $\Delta x_1 = \Delta x_2 = 0.1$, $\Delta t = 0.001$, $\epsilon = 10^{-8}$

	$x_2 =$	.1	.2	.3	.4	.5	.6	.7	.8	.9
scheme										
IMP	EAD-LOD	5.1E-06	1.0E-05	1.4E-05	1.8E-05	2.1E-05	2.3E-05	2.2E-05	1.9E-05	1.2E-05
	EAD-DR	8.1E-06	1.6E-05	2.3E-05	2.9E-05	3.4E-05	3.7E-05	3.6E-05	3.0E-05	1.9E-05
	AGE-2D	8.1E-06	1.6E-05	2.3E-05	2.9E-05	3.4E-05	3.7E-05	3.6E-05	3.0E-05	1.9E-05
CN	EAD	5.1E-06	1.0E-05	1.5E-05	1.8E-05	2.1E-05	2.3E-05	2.2E-05	1.9E-05	1.2E-05
	AGE-2D	5.1E-06	1.0E-05	1.5E-05	1.8E-05	2.1E-05	2.3E-05	2.2E-05	1.9E-05	1.2E-05
Exact solution		0.303308	0.376183	0.468197	0.578931	0.707976	0.854943	1.019463	1.201191	1.399809

c) $x_1 = .9$; *mesh ratio* $= .1$, $\Delta x_1 = \Delta x_2 = 0.1$, $\Delta t = 0.001$, $\epsilon = 10^{-8}$

	$x_2 =$	.1	.2	.3	.4	.5	.6	.7	.8	.9
scheme										
IMP	EAD-LOD	2.7E-06	5.4E-06	7.9E-06	1.0E-05	1.2E-05	1.3E-05	1.3E-05	1.2E-05	8.3E-06
	EAD-DR	4.4E-06	8.6E-06	1.3E-05	1.6E-05	1.9E-05	2.1E-05	2.1E-05	1.9E-05	1.3E-05
	AGE-2D	4.4E-06	8.6E-06	1.3E-05	1.6E-05	1.9E-05	2.1E-05	2.1E-05	1.9E-05	1.3E-05
CN	EAD	2.7E-06	5.4E-06	7.9E-06	1.0E-05	1.2E-05	1.3E-05	1.3E-05	1.2E-05	8.4E-06
	AGE-2D	2.7E-06	5.4E-06	7.9E-06	1.0E-05	1.2E-05	1.3E-05	1.3E-05	1.2E-05	8.4E-06
Exact solution		0.890760	0.990813	1.109460	1.246013	1.399809	1.570209	1.756611	1.958450	2.175209

Table 6.7: The absolute errors of the solutions to the model 2D conduction problem, WITH HEAT SOURCE as obtained by the AGE-2D method and the EAD method, at $t = 0.1$

Method	IMP			CN	
	EAD-LOD	EAD-DR	AGE-2D	EAD-PR	AGE-2D
Average of all absolute errors	9.8E-06	1.57E-05	1.98E-05	9.8E-06	1.24E-05
Number of iterations	4/4	4/2	9	3/3	10
No. (in mxn) of + & * operations	40+; 32*	30+; 24*	135+; 171*	30+; 24*	150+; 190*
Total (in mxn) of +/* operations	74	62	307	60	345
Relative cost w.r.t. AGE-2D	24 %	20 %	100 %	17 %	100 %
No. of points of synchronization	26	20	73	20	81

Table 6.8: Average of absolute errors, number of iterations, and the computational work involved in the experiments of table 6.7.

$x_1 = .5$; *mesh ratio* = 1.0, $\Delta x_1 = \Delta x_2 = 0.1$, $\Delta t = 0.01$

a) $\epsilon = 10^{-4}$

	$x_2 =$	.1	.2	.3	.4	.5	.6	.7	.8	.9
scheme										
IMP	EAD-LOD	4.7E-06	1.0E-05	1.2E-05	1.5E-05	1.7E-05	1.8E-05	1.7E-05	1.4E-05	6.4E-07
	EAD-DR	2.5E-05	4.7E-05	7.1E-05	8.7E-05	1.0E-04	1.0E-04	1.0E-04	8.9E-05	5.2E-05
	AGE-2D	2.8E-05	1.1E-04	8.9E-05	1.6E-04	1.1E-04	1.5E-04	1.0E-04	6.5E-05	7.5E-05
CN	EAD	6.1E-06	1.5E-05	1.5E-05	2.1E-05	1.7E-05	1.4E-05	1.7E-05	1.1E-05	1.0E-05
	AGE-2D	1.6E-05	3.4E-05	4.3E-05	5.7E-05	5.9E-05	6.2E-05	5.4E-05	4.2E-05	2.6E-05
Exact solution		0.289030	0.347770	0.425933	0.523238	0.639410	0.774190	0.927330	1.098597	1.287781

b) $\epsilon = 10^{-8}$

	$x_2 =$	.1	.2	.3	.4	.5	.6	.7	.8	.9
scheme										
IMP	EAD-LOD	4.2E-06	8.3E-06	1.2E-05	1.5E-05	1.7E-05	1.8E-05	1.7E-05	1.4E-05	8.7E-06
	EAD-DR	3.1E-05	6.0E-05	8.6E-05	1.1E-04	1.2E-04	1.3E-04	1.2E-04	1.0E-04	6.4E-05
	AGE-2D	3.0E-05	5.9E-05	8.5E-05	1.1E-04	1.2E-04	1.3E-04	1.2E-04	1.0E-05	6.3E-05
CN	EAD-PR	4.3E-06	8.4E-06	1.2E-05	1.5E-05	1.7E-05	1.8E-05	1.7E-05	1.1E-05	9.0E-06
	AGE-2D	4.3E-06	8.3E-06	1.2E-05	1.5E-05	1.7E-05	1.8E-05	1.7E-05	1.4E-05	8.9E-06
Exact solution		0.289030	0.347770	0.425933	0.523238	0.639410	0.774190	0.927330	1.098597	1.287781

Table 6.9: The absolute errors of the solutions to the model 2D conduction problem, WITH A HEAT SOURCE as obtained by the AGE-2D method and the EAD method, at $t = 0.5$

a) $\epsilon = 10^{-4}$

Method	IMP			CN	
	EAD-LOD	EAD-DR	AGE-2D	EAD-PR	AGE-2D
Average of all absolute errors	1.09E-05	4.71E-05	7.28E-05	8.76E-06	3.2E-05
Number of iterations	4/4	3/1	4	2/2	4
No. (in mxn) of + & * operations	40+; 32*	20+; 16*	60+; 76*	20+; 16*	60+; 76*
Total (in mxn) of +/* operations	74	44	137	60	141
Relative cost w.r.t. AGE-2D	54 %	32 %	100 %	42 %	100 %
No. of points of synchronization	26	14	33	14	33

b) $\epsilon = 10^{-8}$

Method	IMP			CN	
	EAD-LOD	EAD-DR	AGE-2D	EAD-PR	AGE-2D
Average of all absolute errors	7.52E-06	5.5E-05	6.8E-05	7.74E-06	9.6E-06
Number of iterations	8/8	8/6	22	3/3	11
No. (in mxn) of + & * operations	80+; 64*	70+; 56*	330+; 418*	30+; 24*	165+; 209*
Total (in mxn) of +/* operations	146	134	749	114	379
Computational cost w.r.t. AGE-2D	19 %	18 %	100 %	30 %	100 %
No. of points of synchronization	50	44	67	20	89

Table 6.10: Average of absolute errors, number of iterations, and the computational work involved in the experiments of table 6.9.

The tables also include some results which are obtained when the EAD method employs an LOD scheme instead of an ADI scheme. This is the subject of section 6.2.3, where the appropriate comments concerning these results shall be given.

The results show throughout that the EAD method produces more accurate results than the AGE-2D method when the tolerance is not very small (e.g $\epsilon = 10^{-4}$). Of course when the tolerance is too small (e.g $\epsilon = 10^{-8}$) the solutions by both methods converge as expected, to the respective exact finite difference solution and are almost identical. Also it can be seen that the EAD method requires at most just over 50 % of the computations required by the AGE-2D method (see table 6.6). However in most experiments it even requires much less than that, with savings up to 83 % achieved in some cases, see table 6.8.

The other improvement on the AGE-2D is in the overall parallelism. To appreciate that we note that for the execution of one single AGE-2D iteration, we require four sub-iterations of the solution to be done in sequence (see equations 4.98). We also note that the method requires the evaluation of the vector representing the right hand side (i.e. $\mathbf{v_i}$, $i = 1,\ldots 4$) in each of the equations of (4.98) before evaluating explicitly the sub-iteration solution vectors $\mathbf{u}^{p+\frac{i}{4}} = (G_i + sI)^{-1}\mathbf{v_i}$. This means that there are two sets of computations to be done in sequence or two synchronization points at each sub-iteration level. This brings to *eight* the total number of synchronization points per each AGE-2D iteration.

We note here that the pairs of computation sets at the second, third, and fourth sub-iteration levels can be combined easily into one set at each level. This can be done by replacing the implicit (second, third, and fourth equations of (4.98) respectively by the following explicit equations:

$$\begin{aligned}
\mathbf{u}_r^{p+1/2} &= (G_2 + sI)^{-1}G_2\mathbf{u}_r^p + s(G_2 + sI)^{-1}\mathbf{u}_r^{p+1/4} \\
\mathbf{u}_r^{p+3/4} &= (G_3 + sI)^{-1}G_3\mathbf{u}_r^p + s(G_3 + sI)^{-1}\mathbf{u}_r^{p+1/2} \\
\mathbf{u}_r^{p+1} &= (G_4 + sI)^{-1}G_4\mathbf{u}_r^p + s(G_4 + sI)^{-1}\mathbf{u}_r^{p+3/4}
\end{aligned} \tag{6.6}$$

Each of the above equations can be evaluated by one set of computations. This brings down the total number of synchronization points to *five* per each full AGE-2D

iteration but will increase the number of computations required for each iteration.

Hence the total number of synchronization points involved at every time step when applying the AGE-2D method is $(8 * NIT + 1)$ or at best $(5 * NIT + 1)$, NIT being the total number of iterations required. One set is added in each case to account for the set of computations required for evaluating the R.H.S. of (4.93).

On the other hand, the number of synchronization points involved in applying the EAD method, depend only on the total number of AGE-1D iterations (NIT) at the two ADI levels, and on the number of synchronization points involved in each AGE-1D iteration.

The Algorithm for AGE-1D as given by equations (6.5) involves *three* computation sets to be done in sequence: one for evaluating the vector $\mathbf{z_1}$, the second for evaluating the vector $\mathbf{z_2}$ representing the RHS of the second equation in (6.5) and thirdly for evaluating $\mathbf{u}^{p+1} = (G_2 + sI)^{-1}\mathbf{z_2}$. This brings the total number of synchronization points to $(3 * NIT + 2)$. Two more synchronization points are added to account for the two computation sets required for evaluating the R.H.Ss of equations (6.3) and (6.4).

We note here also that the second equation of (6.5) can be evaluated in one set of computations using the following explicit replacement:

$$\mathbf{u}^{p+1} = [(G_2 + sI)^{-1}\mathbf{d}_1 - (G_2 + sI)^{-1}(G_1 - sI)(G_1 + sI)^{-1}\mathbf{z}_1] \tag{6.7}$$

This will reduce the total number of synchronization points to $(2 * NIT + 2)$, but it increases the number of multiplications required each iteration.

Tables 6.2, 6.4, 6.6, 6.8 and 6.10 show throughout that the EAD methods involve a much smaller number of synchronization points than the corresponding AGE-2D method. This reduction is on average much more than *half*, and nearly *three quarters* of the total number of synchronization points. This *broadly* implies that the overall parallelism of the EAD method is better than that of the AGE-2D method.

Finally we comment that the tables show that the computational cost of the EAD-

PR method is (except for table 6.10 a) very close and sometimes considerably less than that of the EAD-DR method. Also for the latter method the values of the solution at the explicit time level t should also be retained for use at the second ADI level, which means that it requires an extra storage of one vector more than the EAD-PR solution.This always justifies, the use of the more accurate EAD-PR method always for the two dimensional problems. We also note that the results obtained for the AGE-2D method in the above comparison are extracted from [20].

6.2.2 A hyperbolic problem

The EAD method can also be applied to a hyperbolic problem. This is the linear two dimensional advection equation given by:

$$\frac{\partial \tilde{u}}{\partial t} = -\frac{\partial \tilde{u}}{\partial x_1} - \frac{\partial \tilde{u}}{\partial x_2} \tag{6.8}$$

defined in $\Re \times t \geq 0$, where $\Re$ is a rectangular region defined by:

$$\Re = (x_1, x_2);\ 0 \leq x_1 \leq L \ and \ 0 \leq x_2 \leq M\ ,$$

subject to the initial conditions:

$$\tilde{u}(x_1, x_2, 0) = e^{x_1} + e^{x_2}\ ;\ (x_1, x_2, t) \in \Re \times 0$$

and where $\tilde{u}(x_1, x_2, t)$ is specified at the boundary $\partial\Re$ of $\Re$ as:

$$\tilde{u}(x_1, x_2, t) = e^{(x_1 - t)} + e^{(x_1 - t)};\ in\ \partial\Re \times t \geq 0$$

To obtain the numerical solution of the above problem we cover the domain with the grid mesh described in section 4.5. A Crank Nicholson type difference approximation to the above equation is given as:

$$[1 + \frac{1}{2}r\delta_{x_1} + \frac{1}{2}r\delta_{x_2}]U_{ij}^{n+1} = [1 - \frac{1}{2}r\delta_{x_1} - \frac{1}{2}r\delta_{x_2}]U_{ij}^{n} \tag{6.9}$$

where r is now $\frac{k}{2h}$.

When equation (6.9) is applied to the totality of meshpoints inside $\Re$, a normalized system (i.e. divided by $\frac{1}{2}r$) having exactly the same structure as (4.93) is obtained, but with $a_1 = 1$, $a_2 = -1$, and $b = \frac{2}{r}$. This system is then solved using the AGE-2D method. On the other hand the EAD-PR method is applied to solve the above problem, where we use the ADI-PR scheme which is now given as:

$$\begin{aligned} (1 + \frac{1}{2}r\delta_{x_1})U_{i,j}^{n*} &= (1 - \frac{1}{2}r\delta_{x_2})U_{i,j}^{n} \\ (1 + \frac{1}{2}r\delta_{x_2})U_{i,j}^{n+1} &= (1 - \frac{1}{2}r\delta_{x_1})U_{i,j}^{n*} \qquad \textbf{(PR)} \end{aligned} \tag{6.10}$$

When applied to the totality of the meshpoints, the pair of equations (6.10) give rise to two systems having the same structure as (4.25) and (4.26) which are to be solved *in sequence.* If these systems were normalized (i.e divided by $\frac{1}{2}r$) we then get the equivalent systems given by equations (6.3) and (6.4), but with $a_1 = 1$, $a_2 = -1$ and $b = \frac{2}{r}$. These systems are then solved using the AGE-1D iterative method as explained above for the parabolic problem. The cost of calculating the R.H.Ss of equations (6.3) and (6.4) and that of equation (4.93) is the same as that given above for the parabolic problem where the corresponding PR and CN methods were used. The costs per each AGE-2D iteration and AGE-1D iteration are also the same as in the parabolic problem.

Tables (6.11) and (6.12) give the results obtained by the AGE-2D method and the EAD method together with the computational work involved. It can be clearly seen that the EAD method again proves to be far less demanding in computational work (needs only 25 % of the computational requirements of the AGE-2D method) and involves a smaller number of sequential sets of operations than the AGE-2D method.

RESULTS AFTER 30 TIME STEPS. The *EAD* method

$\Delta x_1 = \Delta x_2 = h = 0.1$, *time step* $k = 0.05$, *mesh ratio* $(\frac{k}{h}) = .5$; $\epsilon = 10^{-6}$ *accel. param.* $s \in [3.8, 4.4]$

	$x_2 =$	.1	.2	.3	.4	.5	.6	.7	.8	.9
$x_2 = 0.1$	Exact	.49319	.51913	.54779	.57947	.61448	.65317	.69593	.74318	.79541
	Numer.	.48242	.52143	.54015	.58252	.60924	.65627	.69255	.74530	.79419
	percentage error	2.18 %	.44 %	1.4 %	.53 %	.85 %	.47 %	.48 %	.28 %	.15 %
$x_2 = 0.5$	Exact	.61448	.64041	.66907	.70075	.73576	.77445	.81721	.86446	.91669
	Numer.	.60924	.64198	.66431	.70314	.73108	.77684	.81241	.86594	.91197
	percentage error	.85 %	.24 %	.71 %	.34 %	.64 %	.31 %	.59 %	.17 %	.51 %
$x_2 = 0.9$	Exact	.79541	.82134	.85001	.88168	.91669	.95538	.99814	1.04540	1.09762
	Numer.	.79419	.82233	.84709	.88319	.91197	.95683	.99175	1.04637	1.08972
	percentage error	.15 %	.12 %	.34 %	.17 %	.51 %	.15 %	.64 %	.09 %	.72 %
Average of all absolute errors				0.5 %	Number of iterations					4/4
No. (in mxn) of + & * operations				40+; 32*	Total (in mxn) of +/* operations					78
Computational cost w.r.t. AGE-2D				25 %	No. of points of synchronization					26

RESULTS AFTER 30 TIME STEPS. The *AGE-2D* method

$\Delta x_1 = \Delta x_2 = h = 0.1$, *time step* $k = 0.05$, *mesh ratio* $(\frac{k}{h}) = .5$; $\epsilon = 10^{-6}$ *accel. param.* $s \in [4.5, 8.0]$

$x_2 = 0.1$	Exact	.49319	.51913	.54779	.57947	.61448	.65317	.69593	.74318	.79541
	Numer.	.49011	.48479	.54534	.56556	.59820	.65688	.69104	.74228	.79387
	percentage error	.62 %	6.61 %	.49 %	2.4 %	2.65 %	.57 %	.70 %	.12 %	.19 %
$x_2 = 0.5$	Exact	.61448	.64041	.66907	.70075	.73576	.77445	.81721	.86446	.91669
	Numer.	.62054	.65015	.64974	.71164	.72752	.76206	.82168	.86193	.90955
	percentage error	.99 %	1.52 %	2.89 %	1.55 %	1.12 %	1.6 %	.55 %	.29 %	.78 %
$x_2 = 0.9$	Exact	.79541	.82134	.85001	.88168	.91669	.95538	.99814	1.04540	1.09762
	Numer.	.79441	.82371	.84716	.88220	.91258	.95869	.99499	1.05419	1.09559
	percentage error	.12 %	.29 %	.33 %	.06 %	.45 %	.35 %	1.32 %	.18 %	.72 %
Average of all absolute errors				1.0 %	Number of iterations					9
No. (in mxn) of + & * operations				135+; 171*	Total (in mxn) of +/* operations					311
Computational cost w.r.t. AGE-2D				100 %	No. of points of synchronisation					73

Table 6.11: The absolute percentage errors of the solutions to the model 2D advection problem, as obtained by the AGE-2D method and the EAD method, at $t = 1.5$.

Method	CN	
	EAD	AGE-2D
Average of all absolute errors	0.5 %	1.0 %
Number of iterations	4/4	9
No. (in mxn) of + & * operations	40+; 32*	135+; 171*
Total (in mxn) of +/* operations	78	311
Relative cost w.r.t. AGE-2D	25 %	100 %
No. of points of synchronization	26	73

Table 6.12: Average of absolute errors, number of iterations, and the computational work involved in the experiments of table 6.11.

6.2.3 The EAD method with an LOD scheme component

It has been stated that the EAD method may be composed of the AGE-1D method and other alternating direction schemes rather than the ADI method. In this subsection we consider replacing the ADI component of the EAD method for the heat conduction problem given above with the following LOD scheme:

$$\begin{aligned}(1 - r\delta_{x_1}^2)U_{i,j}^{n*} &= U_{i,j}^{n} + kq(x_1, x_2, t + \frac{1}{2}) \\ (1 - r\delta_{x_2}^2)U_{i,j}^{n+1} &= U_{i,j}^{n*} \end{aligned} \tag{6.11}$$

The results obtained by this *EAD-LOD* method appear in *tables* 6.1...6.4, and in *tables* 6.5... 6.10 . It can be seen that compared to the EAD-DR scheme, the EAD-LOD scheme requires generally more computational work, although it needs one vector less of storage, since the values of $U_{i,j}^{n}$ do not need to be retained for the second equation of (6.11) while they need to be saved for the second equation of (6.2).

6.3 The EAD *fully iterative method*

In this section we represent the EAD *fully iterative method* (see figure 6.1-right) as it may be applied to time dependent problems (parabolic and hyperbolic) and as it may be applied to elliptic problems. For time dependent problems of two dimensions or more the EAD *fully iterative method* described in this section uses the respective conventional *fully implicit* or *Crank Nicholson* schemes rather than using the different ADI perturbations of these schemes. It then solves the resulting system of difference equations by using an *inner* AGE-1D iteration procedure within an *outer ADI* iteration procedure (fig 6.1-right). The same principle is applied when solving boundary value problems. The method is illustrated in the following subsections.

6.3.1 A two dimensional hyperbolic problem

We consider the same two dimensional advection problem described in the previous section. If we apply a Crank Nicholson type scheme to the afore mentioned problem we again end up needing to solve a system of difference equations which is given by (4.93) where $a_1 = 1,\ a_2 = -1$, and $b = \frac{2}{r}$.

We then split the coefficient matrix A such that:

$$A = H_1 + V_1 \tag{6.12}$$

where H_1 and V_1 has the same form as H and V of equations (4.25) and (4.26) but with:

$$a_1 = 1 \ and \ a_2 = -1 \quad and \quad b = \frac{1}{r}\ .$$

and then solve (4.93) using an *ADI iterative* algorithm in a manner usually applied in solving systems arising from *boundary value problems*, i.e., we solve:

$$\begin{aligned} (H_1 + \rho I)\mathbf{u}^* &= \mathbf{b} - (V_1 - \rho I)\mathbf{u}^p \\ (V_1 + \rho I)\mathbf{u}^{**} &= \mathbf{b} - (H_1 - \rho I)\mathbf{u}^* \end{aligned} \tag{6.13}$$

or more conveniently:

$$\begin{aligned} (H_1 + \rho I)\mathbf{u}^* &= \mathbf{b} - (V_1 - \rho I)\mathbf{u}^p \\ (V_1 + \rho I)\mathbf{u}^{**} &= (V_1 - \rho I)\mathbf{u}^* + 2\rho\mathbf{u}^* \end{aligned} \tag{6.14}$$

where ρ is an iteration parameter. Each of the systems in (6.14) is then solved using the AGE-1D method as given by (6.5) in which consecutively G_1 and G_2 are determined by the appropriate splittings:

$$H_1 + \rho I = G_1 + G_2 \tag{6.15}$$

and

$$V_1 + \rho I = G_1 + G_2. \tag{6.16}$$

RESULTS AFTER 30 TIME STEPS. The *EAD fully iterative* method
Convergence occured after 4 *ADI iterations each involving* 1/1 *AGE iterations.*
$\Delta x_1 = \Delta x_2 = h = 0.1$, *time step* $k = 0.05$, *mesh ratio* $(\frac{k}{h}) = .5$; $\epsilon = 10^{-6}$ *accel. param.* $s \in [3.8, 4.4]$

	$x_2 =$	.1	.2	.3	.4	.5	.6	.7	.8	.9
$x_2 = 0.1$	Exact	.49319	.51913	.54779	.57947	.61448	.65317	.69593	.74318	.79541
	Numer.	.48244	.52147	.54017	.58256	.60924	.65631	.69253	.74534	.79418
	abs. error	.01076	.00234	.00762	.00310	.00524	.00314	.00340	.00215	.00123
	percentage error	2.18 %	.45 %	1.4 %	.53 %	.85 %	.48 %	.48 %	.29 %	.15 %
$x_2 = 0.5$	Exact	.61448	.64041	.66907	.70075	.73576	.77445	.81721	.86446	.91669
	Numer.	.60924	.64197	.66430	.70313	.73108	.77682	.81242	.86592	.91197
	abs. error	.00524	.00155	.00477	.00238	.00467	.00238	.00479	.00146	.00472
	percentage error	.85 %	.24 %	.71 %	.34 %	.63 %	.31 %	.59 %	.17 %	.51 %
$x_2 = 0.9$	Exact	.79541	.82134	.85001	.88168	.91669	.95538	.99814	1.04540	1.09762
	Numer.	.79418	.82230	.84709	.88317	.91197	.95681	.99176	1.04636	1.08973
	abs. error	.00123	.00096	.00292	.00149	.00472	.00143	.00638	.00096	.00789
	percentage error	.15 %	.12 %	.34 %	.17 %	.51 %	.15 %	.64 %	.09 %	.72 %

Table 6.13: The absolute and percentage errors at $t = 1.5$ of the solutions to the model 2D advection problem, as obtained by the *EAD fully iterative* method, when a CN type difference scheme is applied.

where in (6.15) G_1 and G_2 are as given in equation (4.100), while in (6.16) they have a structure similar to that of G_3 and G_4 respectively. In both cases $a_1 = 1$, $a_2 = -1$ and $c = \frac{b+\rho}{2} = (\frac{1}{2r} + \frac{\rho}{2})$. We thus have an *outer* ADI iteration procedure and an *inner* AGE-1D iteration procedure in the EAD *fully iterative* method. However it is recognized that as each new iteration of the ADI algorithm (6.14) is only an *enhanced* approximation to the required solution we can do with *non exact* values of u^* and u^{**} in (6.14) and thus do *only one* inner AGE iteration. Table 6.13 shows the results after 30 time steps when the method is applied.

The *EAD fully iterative* method	
Average of all absolute errors	0.5 %
Total (in mxn) of + & * operations	60+ & 41*
Total (in mxn) of +/* operations	101
Relative cost w.r.t. AGE-2D	32 %
No. of points of synchronizations	26

Table 6.14: Average of absolute percentage errors, number of iterations, and the computational work involved in the experiments of table 6.13.

The accuracies of the EAD method and the EAD *fully iterative* method are similar. We note that calculating the R.H.S of the first equation in (6.14) requires mn multiplications and $3mn$ additions, and that of the second equation requires mn multiplications and mn additions. This has to be done for each iteration. Table 6.14 gives the total computational cost of this EAD method for this problem. It can be clearly seen that for such a problem (also expected for all 2D time dependent problems) the EAD fully iterative method is not better than the EAD method although it is still much more economic than the AGE-2D method. The advantage of the EAD fully iterative method is however that it can be easily extended to solve the 3D problems which employ the unconditionally stable CN scheme while the EAD method cannot apply a 3D version of the PR scheme due to the stability restrictions as is shown in the next section.

6.3.2 A three dimensional Parabolic Equation

In this subsection the EAD *fully iterative* method is applied to the three dimensional heat-conduction equation which is given as:

$$\frac{\partial \tilde{u}}{\partial t} = \frac{\partial^2 \tilde{u}}{\partial x_1^2} + \frac{\partial^2 \tilde{u}}{\partial x_2^2} + \frac{\partial^2 \tilde{u}}{\partial x_3^2} + q(x_1, x_2, x_3, t) \tag{6.17}$$

defined over the domain given by $\Re_1 = [0 \leq x_1, x_2, x_3 \leq 1]$, $t \geq 0$, with the initial conditions :

$$\tilde{u}(x_1, x_2, x_3, 0) = g_1(x_1, x_2, x_3) \tag{6.18}$$

where $(x_1, x_2, x_3) \in \Re_1$, and the boundary conditions:

$$\tilde{u}(x_1, x_2, x_3, t) = f_1(x_1, x_2, x_3, t) \tag{6.19}$$

where $(x_1, x_2, x_3, t) \in \partial\Re_1 \times [t > 0]$.

We cover $\Re_1$ with a uniform mesh of gridpoints with spacings h_1, h_2, and h_3 in the directions parallel to the axes x_1, x_2, and x_3 respectively, whereby for simplicity we

take $h_1 = h_2 = h_3 = h$. The meshpoints indices in the x_1, x_2, and x_3 directions are i, j, and k respectively, where $0 \leq i, j, k \leq m+1$. Thus $h = 1/(m+1)$. The index for the time direction is n. The mesh ratio is given as $r = k/h^2$, k being the time increment. A weighted finite difference approximation to (6.17) is given by (with $0 \leq \theta \leq 1$) :

$$[1 - \theta r\delta_{x_1}^2 - \theta r\delta_{x_2}^2 - \theta r\delta_{x_3}^2]U_{ijk}^{n+1} =$$
$$[1 + (1-\theta)r\delta_{x_1}^2 + (1-\theta)r\delta_{x_2}^2 + (1-\theta)r\delta_{x_3}^2]U_{ijk}^{n} + kq(x_1, x_2, x_3, t) \quad (6.20)$$

where θ equals $\frac{1}{2}$ and 1 for the Crank-Nicholson and the Implicit schemes respectively. For solving the above heat equation, it is established that such schemes are unconditionally stable. If equation (6.20) is divided by θr and applied to the totality of meshpoints $P_{i,j,k}$ (ordered in the natural sequence of $P_{1,1,1}, P_{2,1,1}, \ldots P_{m,1,1}, \ldots, P_{i,2,1}, \ldots, P_{i,m,1}, \ldots, P_{i,j,2}, \ldots, P_{i,j,3}, \ldots, P_{i,j,m}$) at each time step, it leads to a normalized system of finite difference equations of order m^3. Without loss of generality, we here choose m to be odd. This system is given as:

$$A\mathbf{u}^{n+1} = \mathbf{f} \quad (6.21)$$

where f is a known vector of order m^3 consisting of the boundary values, the source term values at each point, and the solution values at the time level $\mathbf{u}^n$. The vector $\mathbf{u}^{n+1}$ is the solution vector which is to be calculated for the time level $n+1$. The coefficient matrix A is *nonsingular* and has the form:

$$A = \begin{pmatrix} A_2 & -I_2 & & & \\ -I_2 & \ddots & \ddots & 0 & \\ & \ddots & \ddots & \ddots & \\ & 0 & \ddots & \ddots & -I_2 \\ & & & -I_2 & A_2 \end{pmatrix}_{m^3 \times m^3}$$

where I_2 is an identity matrix of order m^2, and A_2 is a block matrix given as:

$$A_2 = \begin{pmatrix} A_1 & -I_1 & & & \\ -I_1 & \ddots & \ddots & 0 & \\ & \ddots & \ddots & \ddots & \\ & 0 & \ddots & \ddots & -I_1 \\ & & & -I_1 & A_1 \end{pmatrix}_{m^2 \times m^2}$$

where I_1 is an identity matrix of order m, and A_1 is given as:

$$A_1 = \begin{pmatrix} c & -1 & & & \\ -1 & \ddots & \ddots & 0 & \\ & \ddots & \ddots & \ddots & \\ & 0 & \ddots & \ddots & -1 \\ & & & -1 & c \end{pmatrix}_{m \times m}$$

with $c = (6 + \frac{1}{r\theta})$. The AGE iterative method which was formulated by [22] for solving (6.21) consists of splitting A into six matrices G_1, G_2, G_3, G_4, G_5 and G_6 such that:

$$A = G_1 + G_2 + G_3 + G_4 + G_5 + G_6 \tag{6.22}$$

where we have:

$$G_1 + G_2 = H = diag(H_2) \tag{6.23}$$

with $H_2 = diag(H_1)$, H_1 being of order m, and given as $H_1 = diag(-1, c/3, -1)$. and

$$G_3 + G_4 = V = diag(V_2) \tag{6.24}$$

where $V_2 = diag(-I_1, V_1, -I_1)$ is of order m^2. The matrix V_1 is of order m and is given as $V_1 = diag(c/3)$.

Also G_5 and G_6 are given such that:

$$G_5 + G_6 = W = diag(-I_2, \frac{c}{3}I_2, -I_2) \tag{6.25}$$

G_1 and G_2 are now given respectively as:

$$G_1 = diag(\tilde{G}_1) \quad and \quad G_2 = diag(\tilde{G}_2)$$

where

$$\tilde{G}_1 = diag(T_1) \quad and \quad \tilde{G}_2 = diag(T_2)$$

T_1 and T_2 are given respectively as:

$$T_1 = \begin{pmatrix} c/6 & & & & \\ & \begin{bmatrix} c/6 & -1 \\ -1 & c/6 \end{bmatrix} & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & \begin{bmatrix} c/6 & -1 \\ -1 & c/6 \end{bmatrix} \end{pmatrix} \tag{6.26}$$

and

$$T_2 = \begin{pmatrix} \begin{bmatrix} c/6 & -1 \\ -1 & c/6 \end{bmatrix} & & & & \\ & \ddots & & & \\ & & \ddots & & \\ & & & \begin{bmatrix} c/6 & -1 \\ -1 & c/6 \end{bmatrix} & \\ & & & & c/6 \end{pmatrix} \tag{6.27}$$

The matrices G_3 and G_4 are given as:

$$G_3 = diag(\tilde{G}_3) \quad and \quad G_4 = diag(\tilde{G}_4)$$

where $\tilde{G}_3$ and $\tilde{G}_4$ are given respectively as:

$$\tilde{G}_3 = \begin{pmatrix} T_2 & & & \\ & \ddots & 0 & \\ & & \ddots & \\ & & & T_2 \\ & 0 & & \frac{c}{6} I_1 \end{pmatrix}_{m^2 \times m^2} \tag{6.28}$$

and

$$\tilde{G}_4 = \begin{pmatrix} \frac{c}{6}I_1 & & & \\ & T_2 & & 0 \\ & & \ddots & \\ & & & \ddots \\ & 0 & & T_2 \end{pmatrix}_{m^2 \times m^2} \tag{6.29}$$

where

$$T_2 = \begin{bmatrix} \frac{c}{6}I_1 & -I_1 \\ -I_1 & \frac{c}{6}I_1 \end{bmatrix}_{2m \times 2m}$$

Simlarily G_5 and G_6 are given respectively as:

$$G_5 = \begin{pmatrix} T_3 & & & \\ & \ddots & & 0 \\ & & \ddots & \\ & & & T_3 \\ & 0 & & \frac{c}{6}I_2 \end{pmatrix}_{m^3 \times m^3} \tag{6.30}$$

and

$$G_6 = \begin{pmatrix} \frac{c}{6}I_2 & & & \\ & T_3 & & 0 \\ & & \ddots & \\ & & & \ddots \\ & 0 & & T_3 \end{pmatrix}_{m^3 \times m^3} \tag{6.31}$$

where

$$T_3 = \begin{bmatrix} \frac{c}{6}I_2 & -I_2 \\ -I_2 & \frac{c}{6}I_2 \end{bmatrix}_{2m^2 \times 2m^2}$$

with I_2 as defined above. It can be seen that the inversion of the matices, G_1, G_2, G_3, G_4, G_5 and G_6 is simply a matter of inverting their respective constituent 2×2 matrices, or 2×2 block matrices. They are therefore *easily invertable* matrices.

The AGE method for this three dimensional problem, henceforth referred to as AGE-3D is a six level iterative formula given as:

$$\begin{aligned}
(G_1 + sI)\mathbf{u}_r^{p+1/6} &= (sI - G_1 - 2G_2 - 2G_3 - 2G_4 - 2G_5 - 2G_6)\mathbf{u}_r^p + 2\mathbf{f} \\
(G_2 + sI)\mathbf{u}_r^{p+2/6} &= G_2\mathbf{u}_r^p + s\mathbf{u}_r^{p+1/6} \\
(G_3 + sI)\mathbf{u}_r^{p+3/6} &= G_3\mathbf{u}_r^p + s\mathbf{u}_r^{p+2/6} \\
(G_4 + sI)\mathbf{u}_r^{p+4/6} &= G_4\mathbf{u}_r^p + s\mathbf{u}_r^{p+3/6} \\
(G_5 + sI)\mathbf{u}_r^{p+5/6} &= G_5\mathbf{u}_r^p + s\mathbf{u}_r^{p+4/6} \\
(G_6 + sI)\mathbf{u}_r^{p+1} &= G_6\mathbf{u}_r^p + s\mathbf{u}_r^{p+5/6}
\end{aligned} \tag{6.32}$$

where the suffix 'r' under the solution vector indicates a rowwise ordering of the components, and s is the acceleration parameter. The first equation of (6.32) can also be written as:

$$(G_1 + sI)\mathbf{u}_r^{p+1/6} = (G_1 + sI - 2A)\mathbf{u}_r^p + 2\mathbf{f} \tag{6.33}$$

It is to be noted here that if when solving for the third and fourth sub-iterations, and the fifth and sixth sub-iterations, the components of the solution vector were ordered rowwise along the x_2 axis and along the x_3 axis respectively, then G_3, G_4 and G_5, G_6 will have the forms as G_1 and G_2. Also with such a change in the ordering of the unknowns, the matrices V and W (defined by equations (6.24) and (6.25) respectively) will have the same form as H defined by equation (6.23).

Thus for the AGE-3D method, the solution is obtained by iterating equation (6.32) in six stages until convergence is obtained.

We now present the *EAD iterative* method for the above three dimensional problem. This consists again of applying the finite difference approximation (6.20) as described above to obtain the same system given by equation(6.21). i.e. we use an *unconditionally stable* finite difference scheme. We then split the coefficient matrix A such that :

$$A = H + V + W$$

and proceed to solve (6.21) by applying an *outer* three levels ADI *iterative* technique given by the following equations:

$$\begin{aligned}(H+\rho I)\mathbf{u}^{p+2/6} &= (\rho - V - W)\mathbf{u}^{p} + \mathbf{b} \\ (V+\rho I)\mathbf{u}^{p+4/6} &= (\rho - H - W)\mathbf{u}^{p+2/6} + \mathbf{b} \\ (W+\rho I)\mathbf{u}^{p+1} &= (\rho - H - V)\mathbf{u}^{p+4/6} + \mathbf{b}\end{aligned} \tag{6.34}$$

where H,V and W are defined in (6.23), (6.24), and (6.25) respectively, and ρ is the ADI iteration parameter. Each of the equations of (6.34) represents a tridiagonal, or block tridiagonal system which is then solved by the AGE-1D method, in what is referred to as the *inner* iterative process in the *EAD fully iterative* method.

To do this we redefine G_1, G_2, G_3, G_4, G_5, and G_6 given above as follows:

$$G_i = G_i + \frac{\rho}{2} I \qquad i = 1, \ldots 6 \tag{6.35}$$

We then do the following splittings:

$$\begin{aligned}(H+\rho I) &= G_1 + G_2 \\ (V+\rho I) &= G_3 + G_4 \\ (W+\rho I) &= G_5 + G_6\end{aligned}$$

where G_i are the newly defined matrices. The AGE-1D algorithm which is applied at each of the three ADI levels of equation (6.34) to solve for $\mathbf{u}^{p+2/6}$, $\mathbf{u}^{p+4/6}$, and $\mathbf{u}^{p+1}$ is that given by equation (6.5) i.e :

$$\begin{aligned}(G_1 + sI)\mathbf{u}^{p+1/2} &= \mathbf{d}_1 - (G_2 - sI)\mathbf{u}^{p} = \mathbf{z} \\ (G_2 + sI)\mathbf{u}^{p+1} &= [\mathbf{d}_1 - (G_1 - sI)(G_1 + sI)^{-1}\mathbf{z}]\end{aligned} \tag{6.36}$$

where $\mathbf{u}^{p+1}$ corresponds to the required iterations of the solution $\mathbf{u}^{p+2/6}$, $\mathbf{u}^{p+4/6}$, and $\mathbf{u}^{p+1}$ at the first, second and third ADI levels respectively. Also $\mathbf{u}^{p+1/2}$ in (6.36) refers also respectively to some intermediate values, i.e $\mathbf{u}^{p+1/6}$, $\mathbf{u}^{p+3/6}$, and $\mathbf{u}^{p+5/6}$.

Again with the same reasoning as put forward in the case of the 2D hyperbolic case earlier, (i.e, because each new iteration of equation (6.34) is only an *enhanced*

approximation to the required solution, we can therefore use *non exact* values of $\mathbf{u}^{p+2/6}$, $\mathbf{u}^{p+4/6}$ and $\mathbf{u}^{p+1}$ in equation (6.34) and do only *one inner* AGE-1D iteration). The solution is obtained for the above problem by the *EAD fully iterative* when the *outer* ADI iteration procedure converges. The *EAD fully iterative* method requires at each ADI level the calculation a RHS of one of the equations of (6.34). This requires m^3 multiplications and $5m^3$ additions. This is added to the cost of $4m^3$ multiplcations and $5m^3$ additions required by each iteration of the AGE-1D algorithm given by (6.5). This makes a total of $5m^3$ multiplications and $10m^3$ additions for each ADI level or $15m^3$ multiplications and $30m^3$ additions for each full EAD iteration. This is compared with the cost of $24m^3$ multiplications and $23m^3$ additions for a full AGE-3D iteration given by equations (6.32). We also make the following count on how many synchronization points the method involves to examine broadly the overall parallelism of each method. We note that for a single AGE-3D iteration, we require six sub-iterations of the solution to be done in sequence (see equations (6.32). The evaluation of the vector representing the right hand side (i.e. $\mathbf{v}_i$, $i = 1, \ldots 6$) in each of the equations of (4.98) has to be performed before evaluating explicitly the sub-iteration solution vectors $\mathbf{u}^{p+\frac{i}{6}} = (G_i + sI)^{-1}\mathbf{v}_i$. This implies that there are two sets of computations to be performed in sequence at each sub-iteration level. This brings the total number of synchronization points to *twelve* per each full AGE-3D iteration. We also note here that the pairs of computation set at the second, third, fourth, fifth, and sixth sub-iteration levels can be combined easily into one set at each level. This can be done by replacing the implicit (second, third, fourth, fifth, and sixth equations of (6.32) respectively by the following explicit equations:

$$
\begin{aligned}
\mathbf{u}_r^{p+2/6} &= (G_2 + sI)^{-1}G_2\mathbf{u}_r^p + s(G_2 + sI)^{-1}\mathbf{u}_r^{p+1/6} \\
\mathbf{u}_r^{p+3/6} &= (G_3 + sI)^{-1}G_3\mathbf{u}_r^p + s(G_3 + sI)^{-1}\mathbf{u}_r^{p+2/6} \\
\mathbf{u}_r^{p+4/6} &= (G_4 + sI)^{-1}G_4\mathbf{u}_r^p + s(G_4 + sI)^{-1}\mathbf{u}_r^{p+3/6} \\
\mathbf{u}_r^{p+5/6} &= (G_5 + sI)^{-1}G_5\mathbf{u}_r^p + s(G_5 + sI)^{-1}\mathbf{u}_r^{p+4/6} \\
\mathbf{u}_r^{p+1} &= (G_6 + sI)^{-1}G_6\mathbf{u}_r^p + s(G_6 + sI)^{-1}\mathbf{u}_r^{p+5/6}
\end{aligned}
\tag{6.37}
$$

Each of the above equations can be evaluated by one set of computations. This will increase the number of computations involved in one full AGE-3D iteration, but will bring down the number of synchronization points to *seven* sets per each iteration. We note that the first equation of (6.32) is much more complicated than the second, third and the fourth, fifth, and sixth and is thus more difficult to evaluate it explicitly. Hence the total number of synchronization points involved at every time step when applying the AGE-3D method is $(12 * NIT + 1)$ or at best $(7 * NIT + 1)$, NIT being the total number of iterations required. One point is added in each case to account for the set of computations required for evaluating the R.H.S. of (6.21).

On the other hand, when applying the *EAD fully iterative* method, we require one set of operations to evaluate the R.H.S at each ADI level, and (see page 141) three sets of operations to execute one AGE-1D iteration in that level, if the algorithm given by (6.36) is used. This makes a sum of 4 sets of computations at each ADI level, or a total of 12 sets for the execution of a full iteration of the EAD method. It is also noted before (see page 141) that every one AGE-1D iteration at each ADI level may be executed in only two sets of computations. This will again increase the number of computations to be done for each AGE-1D sub-iteration but will diminish the number of *sets* of computations which are to be done in sequence. The total number of synchronization points for the *EAD fully iterative* method is $(12 * NIT + 1)$, or at best $(9 * NIT + 1)$, NIT being the total number of iterations required. Again one set is added in each case to account for the set of computations required for evaluating the R.H.S. of (6.21). In the following experiment we do not attempt to reduce the number of synchronization points. We compare between the two methods by the following experiments equation where the source function q of equation (6.17) is given as:

$$q(x_1, x_2, x_3, t) = (3\pi^2 - 1)\sin(\pi x_1)\sin(\pi x_2)\sin(\pi x_3)\exp(-t) \tag{6.38}$$

and the functions defining the initial and boundary conditions are:

$$g_1(x_1, x_2, x_3, 0) = \sin(\pi x_1)\sin(\pi x_2)\sin(\pi x_3) \tag{6.39}$$

$$f_1(x_1, x_2, x_3, t) = \sin(\pi x_1)\sin(\pi x_2)\sin(\pi x_3)\exp(-t) \tag{6.40}$$

as derived from the following exact solution to equation (6.17):

$$\tilde{u}(x_1, x_2, x_3, t) = \sin(\pi x_1)\sin(\pi x_2)\sin(\pi x_3)\exp(-t) \tag{6.41}$$

The results which are obtained by applying the *EAD fully iterative*, are compared with those obtained by applying the AGE-3D method as given in [22] where we use a grid having 39 internal meshpoints along each of the directions x_1, x_2, and x_3, thus making the space step $h = 0.025$. Also a time step $k = 3.125 \times 10^{-5}$ is chosen, thus giving a mesh ratio of 0.05 . These results are shown in tables 6.15 and 6.16. Table 6.15 shows that the *EAD fully iterative* method converges much faster than the AGE-3D method (about four times faster). This fast convergence produces an even better accuracy of the solution (although both methods use identical finite difference schemes) when the tolerence applied is not very small (e.g. $\varepsilon = 10^{-4}$). This is because a method which converges faster cuts down, on average, larger chunks of error in the solution vector every iteration, and is likely to make a higher reduction of the error in the solution vector before it converges. The small number of iterations for the *EAD fully iterative* also results in much less computational work to be done, and a smaller number of sequential sets of operations to be executed. Table 6.16 shows that the *EAD fully iterative* requires less than a quarter of the computations [1] needed by the AGE-3D method. It also shows that the number of sets of computations, to be pereformed in sequence is cut down to only a quarter by the *EAD fully iterative* method as compared with the AGE-3D method. This implies a significant superiority in the overall parallelism of the method.

---

[1]This does not include the cost of calculating the R.H.S of (6.21) which is the same for both methods, and is trivial compared to the sums in this table.

a) $x_2 = 0.025$, $x_3 = 0.025$;

mesh ratio = 0.05, $\Delta x_1 = \Delta x_2 = \Delta x_3 = 0.025$, $\Delta t = 3.125E-05$, $t = 3.125E-05$, $\epsilon = 10^{-4}$						
scheme	x_1 =	.025	.50	0.975	No. of iterations	Average of errors
IMP	EAD	2.89E-08	3.14E-06	7.86E-07	2	1.3E-08
	AGE-3D	1.81E-06	3.89E-05	6.99E-07	9	2.4E-07
CN	EAD	2.98E-08	5.61E-07	5.14E-07	2	3.68E-08
	AGE-3D	7.1E-08	1.41E-05	5.6E-08	9	4.72E-07
Exact solution		0.000482996	0.006156022	0.000482996	...	...

b) $x_2 = 0.975$, $x_3 = 0.025$;

mesh ratio = 0.05, $\Delta x_1 = \Delta x_2 = \Delta x_3 = 0.025$, $\Delta t = 3.125E-05$, $t = 6.250E-05$, $\epsilon = 10^{-4}$						
scheme	x_1 =	.025	.50	0.975	No. of iterations	Average of errors
IMP	EAD	4.32E-08	7.75E-06	1.62E-06	2	2.71E-06
	AGE-3D	7.0E-06	2.93E-04	7.02E-05	8	1.23E-04
CN	EAD	4.35E-09	2.22E-06	1.04E-06	2	1.09E-06
	AGE-3D	1.67E-06	2.16E-04	7.46E-05	9	7.5E-05
Exact solution		0.000483011	0.006156214	0.000483011	...	...

c) $x_2 = 0.975$, $x_3 = 0.5$;

mesh ratio = 0.05, $\Delta x_1 = \Delta x_2 = \Delta x_3 = 0.025$, $\Delta t = 3.125E-05$, $t = 1.250E-04$, $\epsilon = 10^{-4}$						
scheme	x_1 =	.025	.50	0.975	No. of iterations	Average of errors
IMP	EAD	5.14E-06	2.54E-04	4.31E-05	2	1.0E-04
	AGE-3D	9.58E-04	2.59E-02	8.57E-04	8	9.2E-03
CN	EAD	1.4E-06	8.32E-05	2.56E-05	2	3.67E-03
	AGE-3D	2.81E-04	2.16E-02	5.16E-04	8	7.4E-03
Exact solution		0.006156599	0.078468904	0.006156599	...	...

d) $x_2 = 0.5$, $x_3 = 0.5$;

mesh ratio = 0.05, $\Delta x_1 = \Delta x_2 = \Delta x_3 = 0.025$, $\Delta t = 3.125E-05$, $t = 1.5625E-04$, $\epsilon = 10^{-4}$						
scheme	x_1 =	.025	.50	0.975	No. of iterations	Average of errors
IMP	EAD	6.6E-05	3.85E-03	6.56E-04	2	1.52E-03
	AGE-3D	9.83E-03	1.88E-01	1.55E-02	8	7.1E-02
CN	EAD	7.82E-06	1.19E-03	3.79E-04	2	5.25E-04
	AGE-3D	3.95E-03	8.77E-02	3.89E-03	8	3.1E-02
Exact solution		0.078471356	1.000156262	0.078471356	...	...

Table 6.15: The absolute errors of the solutions by the AGE-3D and the *EAD fully iterative* methods to the model 3D heat conduction problem.

Method	IMP		CN	
	EAD	AGE-3D	EAD	AGE-3D
Number of iterations	2	8	2	8
No. (in m^3) of + & * operations	30+; 60*	184+; 192*	30+; 60*	184+; 192*
Total (in m^3) of +/* operations	90	376	42	376
Computational cost w.r.t. AGE-3D	24 %	100 %	24 %	100 %
No. of synchronisation points	25	97	25	97

Table 6.16: The computational work involved in the experiments of table 6.15.

6.4 The EAD *fully Iterative* method for Elliptic problems

In this section, the *EAD fully iterative* method strategy is applied in the solution of a *boundary value* problem.

Consider the simple two dimensional Laplace equation given by:

$$\frac{\partial^2 \tilde{u}}{\partial x_1^2} + \frac{\partial^2 \tilde{u}}{\partial x_2^2} = 0 \tag{6.42}$$

defined over the domain $\Re \cup \partial\Re$. where $\Re = [0 < x_1, x_2 < L]$, with the *Dirichlet* boundary conditions :

$$\tilde{u}(x_1, x_2) = f_1(x_1, x_2) \quad (x_1, x_2) \in \partial\Re \tag{6.43}$$

We cover $\Re$ with a uniform mesh of gridpoints with spacings, h_1, and h_2, in the directions parallel to the axes x_1, and x_2 respectively, whereby for simplicity we take $h_1 = h_2 = h$. We choose h such that we have an odd number m of gridpoints lying inside $\Re$ along any line of the mesh along the x_1 or x_2 directions. We thus have $h = L/(m+1)$.

A central finite difference replacement of the second order derivatives to $\mathcal{O}(h^2)$ accuracy in equation (6.42) gives :

$$\frac{1}{h^2}[\delta_{x_1}^2 + \delta_{x_2}^2]U_{ij} = 0 \tag{6.44}$$

which leads to the following five point formula:

$$-U_{i+1,j} - U_{i-1,j} - U_{i,j+1} - U_{i,j-1} + 4U_{ij} = 0 \tag{6.45}$$

If we apply equation (6.45) to the totality of mesh points in $\Re$ we obtain a symmetric system of difference equations given as:

$$A\mathbf{u} = \mathbf{b} \tag{6.46}$$

where A is a *nonsingular* square matrix of order m^2 given by:

$$A = \begin{pmatrix} A_1 & -I_1 & & \\ -I_1 & \ddots & \ddots & 0 \\ & \ddots & \ddots & \ddots \\ & 0 & \ddots & \ddots & -I_1 \\ & & & -I_1 & A_1 \end{pmatrix}_{m^2 \times m^2}$$

where I_1 is an identity matrix of order m, and A_1 is a matrix given as:

$$A_1 = \begin{pmatrix} 4 & -1 & & \\ -1 & \ddots & \ddots & 0 \\ & \ddots & \ddots & \ddots \\ & 0 & \ddots & \ddots & -1 \\ & & & -1 & 4 \end{pmatrix}_{m \times m}$$

and **b** is the vector associated with the values at the boundaries.

If the Peaceman-Rachford method is applied to solve the system (6.46) iteratively we need to split A into :

$$A = H + V \tag{6.47}$$

and solve the following two systems in sequence:

$$\begin{aligned} (H + \rho I)\mathbf{u}^{p+1/2} &= (\rho I - V)\mathbf{u}^p + \mathbf{b} \\ (V + \rho I)\mathbf{u}^{p+1} &= (\rho I - H)\mathbf{u}^{p+1/2} + \mathbf{b} \end{aligned} \tag{6.48}$$

where $H = diag(H_1)$, with $H_1 = (-1, 2, -1)$ and $V = diag(-I_1, 2I_1, -I_1)$, and ρ a chosen acceleration parameter.

Equations (6.48) are equivalent to the following, less computationaly demanding, systems:

$$\begin{aligned} (H + \rho I)\mathbf{u}^{p+1/2} &= (\rho I - V)\mathbf{u}^p + \mathbf{b} = \mathbf{q} \\ (V + \rho I)\mathbf{u}^{p+1} &= (V - \rho I)\mathbf{u}^p + 2\rho\mathbf{u}^{p+1/2} \end{aligned} \tag{6.49}$$

It is customary to solve each of the two systems in (6.49) using a *direct* method i.e Guassian elimination.

The *EAD fully iterative* technique for solving (6.46) employs the Peaceman-Rachford method producing equations (6.49), but rather than using a direct tridiagonal solver, it implements the AGE-1D algorithm given by (6.5) to solve each system in (6.48). There are three ways of implementing the AGE-1D within each ADI iteration here.

Strategy I: One is to iterate the AGE-1D algorithm until it converges to the values $\mathbf{u}^{p+1/2}$ and $\mathbf{u}^{p+1}$ in (6.49), and this approach is referred to as strategy I.

Strategy II: Another way is to apply only one sweep of AGE-1D within each ADI iteration, which is the strategy followed for the *initial boundary value* problems in subsections 6.3.1 and 6.3.2. This is referred to as strategy II, and is based on the reasoning that the values $\mathbf{u}^{p+1/2}$ and $\mathbf{u}^{p+1}$ in each ADI iteration of (6.48) are only enhanced values of the solution and need not to be computed *exactly* each time.

Strategy III We note however that unlike the time dependent problems where the starting vector, representing the values at the previous time step, is close to the required solution, the starting vector here may be very far off the required solution. We therefore use *more enhanced non-exact* values of $\mathbf{u}^{p+1/2}$ and $\mathbf{u}^{p+1}$, for the first ADI iteration by having three inner AGE-1D iterations. In subsequent ADI iterations only one sweep of the AGE-1D algorithm is used. This is strategy III.

For strategy I convergence is expected for a low number of ADI iterations. However the computational cost for each ADI iteration of strategy I is high compared to that of strategies II and III.

In strategy III, it is expected that by having *three* sweeps of the AGE-1D algorithm in the first ADI iteration, a better starting vector will be provided for the subsequent ADI iterations and will thus result in a significant improvement in the convergence rate compared to strategy II, and thus save the computational cost further. Strategy III is thus recommended, and is henceforth meant whenever reference is made to the *EAD fully iterative method* for *elliptic* problems.

Tables 6.17 ... 6.20 give the numerical solutions to the Laplace equations by the *EAD fully iterative* method and a comparison of the computational costs of strategies I, II, and III. It is clear that strategy III is most desirable, requiring about 43 % and 72 % of the computational cost of strategy I in the first and second experiments respectively.

The counts for the number of additions/subtractions and multiplications is done as follows. The cost of calculating the *right hand side* of the first equation in (6.49) is easily worked out as m^2 multiplications and $3m^2 - 2m$ additions. The operations count for calculating the *rhs* of the second equation in (6.49) is m^2 additions and m^2 multiplications. This makes the total for both right hand sides $2m^2$ multiplications and $4m^2 - 2m$ ($\simeq 4m^2$) additions or $6m^2$ $(+/\times)$ operations. To this we add the cost of solving each block tridiagonal system at each of the levels in every ADI iteration. This depends on the number of sweeps of the AGE-1D algorithm of (6.5) applied at each ADI level in each iteration. Each sweep costs $5m^2$ additions and $5m^2$ multiplications or $10m^2$ $(+/\times)$ operations.

For strategy II, we have only one sweep of the AGE-1D algorithm in each of the two ADI levels. This together with the costs of calculating the right hand sides in (6.49) make the total cost for strategy II equal to $(20m^2 + 6m^2)$NIT, or $(26m^2$NIT) operations, where NIT is the number of ADI iterations needed for the solution to converge.

For strategy III, the cost is $(60m^2+20m^2(NIT-1)+6m^2NIT)$ or $26m^2NIT+40m^2$ operations.
For strategy I the number of sweeps of the AGE-1D algorithm, required in each ADI iteration starts high and drops to 1 at the final ADI iteration. The average number of sweeps per each ADI level is denoted by β and is given in brackets for the respective experiments in tables 6.18 and 6.20. The total cost for strategy I is thus $(20m^2\beta NIT + 6m^2NIT)$.

Experiment

Consider the above Laplace equation where the Dirichlet conditions at the boundaries are given as follows:

$$\tilde{u}(x_1, 0) = \tilde{u}(x_1, L) = f(x_1) = x_1(x_1 - L)$$

$$\tilde{u}(0, x_2) = \tilde{u}(L, x_2) = g(x_2) = x_2(L - x_2) \tag{6.50}$$

The exact solution of such a problem is unique and is given by:

$$\tilde{u}(x_1, x_2) = x_1(x_1 - L) + x_2(L - x_2) \tag{6.51}$$

By discretizing the problem as described above, we arrive at the system (6.46) where the vector **b** is given as:

$$\begin{aligned} b = [&g(x_2(1)) + f(x_1(1)), f(x_1(2)), \ldots, f(x_1(m-1)), g(x_2(1)) + f(x_1(m)); \\ &g(x_2(2)), 0, \ldots, 0, g(x_2(2)); \ldots, \ g(x_2(m-1)), 0, \ldots, 0, g(x_2(m-1)); \\ &f(x_1(1)) + g(x_2(m)), f(x_1(2)), \ldots, f(x_1(m-1)), f(x_1(m)) + g(x_2(m))]^T \end{aligned}$$

We follow the same steps above for (6.49) which is rewritten here as:

$$\begin{aligned} (H + \rho I)\mathbf{u}^{p+1/2} &= \underbrace{(\rho I - V)\mathbf{u}^p}_{\alpha} + \mathbf{b} \equiv \mathbf{q} \\ (V + \rho I)\mathbf{u}^{p+1} &= (V - \rho I)\mathbf{u}^p + 2\rho\mathbf{u}^{p+1/2} \end{aligned} \tag{6.52}$$

The term denoted by vector α above may be stored after being calculated to be used in the second equation of (6.49) to save work. Alternatively, we note from the first equation of (6.49) that :

$$(V - \rho I)\mathbf{u}^p = \mathbf{b} - \mathbf{q} \tag{6.53}$$

Therefore, we may replace the second equation in (6.52) by :

$$(V + \rho I)\mathbf{u}^{p+1} = \mathbf{b} - \mathbf{q} + 2\rho\mathbf{u}^{p+1/2} \tag{6.54}$$

This costs an extra m^2 additions/subtractions, but saves the need for the extra storage of the vector α, and reduces by one the number of synchronization points in the execution of each ADI iteration.

$\Delta x_1 = \Delta x_2 = .027,\ \varepsilon = 10^{-6}$

$x_1 =$	.027	.136	.245	.354	.463	.572
$x_2 = .027$						
Anal.	.0000000	.0912397	.1586777	.2023140	.2221488	.2181818
Numer.	-.0000012	.0912400	.1586780	.2023139	.2221488	.2181815
error	1.2E-06	3.1E-07	3.6E-07	1.7E-07	4.9E-07	2.7E-07
$x_2 = .245$						
Anal.	-.1586777	-.0674380	.0000000	.0436364	.0634711	.0595041
Numer.	-.1586772	-.0674367	-.0000009	.0436360	.0634700	.0595038
error	4.9E-07	1.3E-06	9.0E-07	3.2E-07	1.0E-06	3.4E-07
$x_2 = .464$						
Anal.	-.2221488	-.1309091	-.0634711	-.0198347	.0000000	-.0039669
Numer.	-.2221480	-.1309059	-.0634672	-.0198317	.0000016	-.0039666
error	7.8E-07	3.1E-06	3.9E-06	3.0E-06	1.6E-06	3.6E-07

Table 6.17: The absolute errors of the solutions to the Laplace equation, as obtained by the EAD fully iterative method, strategy III for $h = 6/210$.

The computational cost for different strategies of the EAD fully iterative method.			
	Strategy I	Strategy II	Strategy III
ADI/ AGE parameters	$\rho = 0.4$ & $s = 1.0$	$\rho = 0.3$ & $s = 1.0$	$\rho = 0.3$ & $s = 1.0$
No. of ADI/ AGE iterations	28 (4.8)	69 (1/1)	1 (3/3) & 45(1/1)
Average of absolute errors	6.3E-07	$3.8E-06$	1.26E-06
Total number of (+/x) in m^2	2856	1794	1236

Table 6.18: The computational cost of the three possible strategies of the EAD fully iterative method for the experiment of table 6.17

$\Delta x_1 = \Delta x_2 = .0375,\ \varepsilon = 10^{-6}$

$x_1 =$	.037	.15	.26	.37	.49
$x_1 = .037$					
Anal.	.0000000	.0914062	.1575000	.1982812	.2137500
Numer.	.0000003	.0914072	.1575013	.1982825	.2137507
error	2.7E-07	9.6E-07	1.3E-06	1.2E-06	7.1E-07
$x_1 = .262$					
Anal.	-.1575000	-.0660938	.0000000	.0407812	.0562500
Numer.	-.1574985	-.0660886	.0000070	.0407877	.0562538
error	1.5E-06	5.1E-06	7.0E-06	6.4E-06	3.8E-06
$x_1 = .487$					
Anal.	-.2137500	-.1223438	-.0562500	-.0154688	.0000000
Numer.	-.2137491	-.1223406	-.0562458	-.0154648	.0000023
error	8.9E-07	3.1E-06	4.2E-06	3.9E-06	2.3E-06

Table 6.19: The absolute errors of the solutions to the Laplace equation, as obtained by the EAD iterative method, strategy III for $h = 6/160$.

The computational cost for different strategies of the EAD fully iterative method.			
	Strategy I	Strategy II	Strategy III
ADI/ AGE parameters	$\rho = 0.7$ and $s = 1.0$	$\rho = 0.9$ & $s = 1.3$	$\rho = 0.7$ & $s = 1.4$
No. of ADI/ AGE iterations	24(3/3)	46 (1/1)	1(3/3) & 42(1/1)
Average of absolute errors	$3.2E-06$	$3.6E-06$	3.3E-06
Total number of (+/x) in m^2	1604	1196	1158

Table 6.20: The computational cost of the three possible strategies of the EAD fully iterative method, for $h = 6/160$.

Two experiments for solving the above problem with different mesh sizes are carried out. For both experiments we choose $L = 0.6$. We also choose $m = 20$ for the first experiment, tables 6.17 and 6.18, and $m = 15$ for the second experiment, tables 6.19 and 6.20, thus giving the mesh sizes $h = 0.6/21$ and $h = 0.6/16$ respectively.

6.5 Convergence analysis of the EAD fully iterative method

The convergence of the EAD fully iterative follows in an obvious manner if the convergence of its outer and inner iteration procedures is assured.

For the inner AGE-1D iteration the convergence is assured by the analyses given in subsections 4.4.2 and 5.3.2.

For the case where the EAD fully iterative method is applied to two dimensional problems, where the constituent matrix H_1 and V_1 of the coefficient matrix A are symmetric positive definites (e.g case of the elliptic problem of section 6.4) the convergence of the outer ADI iterative procedure is well documented in the literature (e.g see [49]p:212-213 and [28]p:196).

In the following two subsections the convergence of the outer ADI iterative procedure is given for the problems of subsections 6.3.1 and 6.3.2 respectively.

6.5.1 The two dimensional advection problem

As for the application of the EAD fully iterative method for the two dimensional advection problem of subsection 6.3.1 it is noted that the outer ADI iterative procedure is given by (6.13). This is rewritten here as :

$$\begin{aligned} (H_1 + \rho I)\mathbf{u}^{p+1/2} &= (\rho I - V_1)\mathbf{u}^{p} + \mathbf{b} \\ (V_1 + \rho I)\mathbf{u}^{p+1} &= (\rho I - H_1)\mathbf{u}^{p+1/2} + \mathbf{b} \end{aligned} \tag{6.55}$$

with ρ being the acceleration parameter.

The matrices H_1 and V_1, each of order m^2, are again given here respectively as:

$$H_1 = diag(T_1) \quad where \quad T_1 = diag(a_2, b, a_1)$$

$$and \qquad V_1 = diag(a_4 I, bI, a_3 I);$$

For the advection problem we have the following relations:

$$a_1 = -a_2 = a_x \ \ and \ \ a_3 = -a_4 = a_y$$

($a_x = a_y = 1$ for the normalized system of subsection 6.3.1)

and $b = 1/r$, where $r = k/2h$. k and h being respectively the time increment and space step used in the discretization of the problem.

Now we establish the proof for the convergence of the ADI procedure of (6.55). The iteration matrix of (6.13) is given by:

$$T_\rho = (V_1 + \rho I)^{-1}(H_1 - \rho I)(H_1 + \rho I)^{-1}(V_1 - \rho I) \tag{6.56}$$

is the ADI iteration matrix. The ADI iterative procedure will converge if :

$$S(T_\rho) \leq 1 \tag{6.57}$$

By the special structures of H_1 and V_1, it can be shown that they are *commutative* i.e $H_1 V_1 = V_1 H_1$. Therefore they share the same *eigenvector matrix* X such that

$$H_1 = X\Lambda_1 X^{-1} \quad and \quad V_1 = X\Lambda_2 X^{-1}. \tag{6.58}$$

where $\Lambda_1 = diag(\lambda_j)$ and $\Lambda_2 = diag(\eta_j)$ are diagonal matrices with elements λ_j and η_j representing respectively the eigenvalues of H_1 and V_1. It is noted also from the structures of H_1 and V_1 that they have only m distinct eigenvalues. These are given by:

$$\lambda_j = b + iv_j, \quad \eta_j = b + iw_j \quad with \ \ i^2 = -1,$$

where

$$v_j = 2a_x cos(\frac{j\pi}{(n+1)}), \quad and \quad w_j = 2a_y cos(\frac{j\pi}{(n+1)}) \quad j = 1, \ldots m$$

By (6.56) and (6.58), the eigenvalues ξ_j of T_ρ can be given as:

$$\xi_j = \frac{(\lambda_j - \rho)(\eta_j - \rho)}{(\lambda_j + \rho)(\eta_j + \rho)}$$

Hence the spectral radius $S(T_\rho)$ is given by:

$$\mathbf{S}(T_\rho) = \max_j \mid \xi_j \mid = \max_j \frac{\sqrt{(b-\rho)^2 + v_j^2}}{\sqrt{(b+\rho)^2 + v_j^2}} \frac{\sqrt{(b-\rho)^2 + w_j^2}}{\sqrt{(b+\rho)^2 + w_j^2}} \tag{6.59}$$

By defining

$$v = \max_j v_j = 2a_x \cos\left(\frac{\pi}{m+1}\right) \quad ; \quad w = \max_j w_j = 2a_y \cos\left(\frac{\pi}{m+1}\right) \tag{6.60}$$

$$\Longrightarrow S(T_\rho) = \sqrt{\frac{((b-\rho)^2 + v^2)}{((b+\rho)^2 + v^2)} \frac{((b-\rho)^2 + w^2)}{((b+\rho)^2 + w^2)}} \leq 1 \quad for\ every\ \rho > 0 \tag{6.61}$$

Hence the method is convergent.

6.5.2 The three dimensional heat conduction problem

We now consider the convergence analysis for the EAD fully iterative method given by equations (6.34) for the three dimensional parabolic problem. The iteration matrix T_ρ of the method is given as:

$$T_\rho = (W + \rho I)^{-1}(\rho I - V - H)(V + \rho I)^{-1}(\rho I - H - W)(H + \rho I)^{-1}(\rho I - V - W)$$

By virtue of the mutual commutativity of H,V, and W, the eigenvalues of T_ρ may be given as:

$$\lambda_\ell(T_\rho) = (\frac{\lambda_\ell(\rho I - V - H)}{\lambda_\ell(W + \rho I)})(\frac{\lambda_\ell(\rho I - H - W)}{\lambda_\ell(V + \rho I)})(\frac{\lambda_\ell(\rho I - V - W)}{\lambda_\ell(H + \rho I)}) \tag{6.62}$$

If we define:

$$H_1 = H - \frac{c}{3}I$$
$$V_1 = V - \frac{c}{3}I$$
$$W_1 = W - \frac{c}{3}I$$

Then equation (6.62) becomes:

$$\lambda_\ell(T_\rho) = \left(\frac{\lambda_\ell[(\rho - \frac{2c}{3})I - V_1 - H_1]}{\lambda_\ell[W_1 + (\rho + \frac{c}{3})I]}\right) \times \left(\frac{\lambda_\ell[(\rho - \frac{2c}{3})I - H_1 - W_1]}{\lambda_\ell[V_1 + (\rho + \frac{c}{3})I]}\right) \times \left(\frac{\lambda_\ell[(\rho - \frac{2c}{3})I - V_1 - W_1]}{\lambda_\ell[H_1 + (\rho + \frac{c}{3})I]}\right) \quad (6.63)$$

Then by the formula of (2.13) the eigenvalues of T_ρ can be written as:

$$\lambda_{i,j,k}(T_\rho) = \left(\frac{\rho - \frac{2c}{3} + 2\cos(\frac{i\pi}{m_1+1}) + 2\cos(\frac{j\pi}{m_2+1})}{2\cos(\frac{k\pi}{m_3+1}) + \frac{c}{3} + \rho}\right) \times \left(\frac{\rho - \frac{2c}{3} + 2\cos(\frac{i\pi}{m_1+1}) + 2\cos(\frac{k\pi}{m_3+1})}{2\cos(\frac{j\pi}{m_2+1}) + \frac{c}{3} + \rho}\right) \times \left(\frac{\rho - \frac{2c}{3} + 2\cos(\frac{k\pi}{m_3+1}) + 2\cos(\frac{j\pi}{m_2+1})}{2\cos(\frac{i\pi}{m_1+1}) + \frac{c}{3} + \rho}\right) \quad (6.64)$$
$$i = 1, \ldots, m_1 \; ; \quad j = 1, \ldots m_2 \; ; \quad k = 1, \ldots m_3$$

from which we can write:

$$S(T_\rho) = \max_{i,j,k}\{\lambda_{i,j,k}(T_\rho)\} \leq \max_{i,j,k}\left\{\left|\frac{\rho - \frac{2c}{3} + 2\cos(\frac{i\pi}{m_1+1}) + 2\cos(\frac{j\pi}{m_2+1})}{2\cos(\frac{k\pi}{m_3+1}) + \frac{c}{3} + \rho}\right|\right\} \times \max_{i,j,k}\left\{\left|\frac{\rho - \frac{2c}{3} + 2\cos(\frac{i\pi}{m_1+1}) + 2\cos(\frac{k\pi}{m_3+1})}{2\cos(\frac{j\pi}{m_2+1}) + \frac{c}{3} + \rho}\right|\right\} \times \max_{i,j,k}\left\{\left|\frac{\rho - \frac{2c}{3} + 2\cos(\frac{k\pi}{m_3+1}) + 2\cos(\frac{j\pi}{m_2+1})}{2\cos(\frac{i\pi}{m_1+1}) + \frac{c}{3} + \rho}\right|\right\} \quad (6.65)$$
$$i = 1, \ldots, m_1 \; ; \quad j = 1, \ldots m_2 \; ; \quad k = 1, \ldots m_3$$

We now consider the first term on the right hand side of inequality (6.65). For this term to have a modulus less than unity it must have:

$$-1 \leq \left(\frac{\rho - \frac{2c}{3} + 2\cos(\frac{i\pi}{m_1+1}) + 2\cos(\frac{j\pi}{m_2+1})}{2\cos(\frac{k\pi}{m_3+1}) + \frac{c}{3} + \rho)}\right) \leq 1 \quad (6.66)$$

To satisfy the right inequality, we should have:

$$\rho - \frac{2c}{3} + 2\cos(\frac{i\pi}{m_1+1}) + 2\cos(\frac{j\pi}{m_2+1}) \leq 2\cos(\frac{k\pi}{m_3+1}) + \frac{c}{3} + \rho \qquad (6.67)$$

or

$$-c \quad \leq 2\cos(\frac{k\pi}{m_3+1}) - 2\cos(\frac{i\pi}{m_1+1}) - 2\cos(\frac{j\pi}{m_2+1}) \qquad (6.68)$$

which is always satisfied because $c > 6$.

To satisfy the left inequality of (6.66) we should have:

$$-2\cos(\frac{k\pi}{m_3+1}) - \frac{c}{3} - \rho \leq \rho - \frac{2c}{3} + 2\cos(\frac{i\pi}{m_1+1}) + 2\cos(\frac{j\pi}{m_2+1}) \qquad (6.69)$$

$$+\frac{c}{3} - 2\rho \leq +2\cos(\frac{k\pi}{m_3+1}) + 2\cos(\frac{i\pi}{m_1+1}) + 2\cos(\frac{j\pi}{m_2+1}) \qquad (6.70)$$

The right side of (6.70) is greater than -6, therefore (6.70) is satisfied if:

$$+\frac{c}{3} - 2\rho \leq -6 \qquad (6.71)$$

Note that $c = 6 + \frac{1}{r\theta}$. Therefore the left inequality of (6.66) is satisfied if:

$$\rho > 4 + \frac{1}{6r\theta} \qquad (6.72)$$

A similar condition can be derived to assure that the second and the third terms on the right hand side of inequality (6.65) are less than unity.

This means that the convergence of the iterative procedure given by (6.34) is assured for values of the acceleration parameter ρ which satisfy (6.72).

6.6 Consistency of the three level ADI iterative procedure

We consider here the proof of the following lemma.

Lemma 6.1 *The three level ADI iterative procedure defined by equation (6.34) for solving (6.21) is* consistent.

Proof :

Upon eliminating $\mathbf{u}^{p+2/6}$ and $\mathbf{u}^{p+4/6}$ the three level ADI iterative procedure may be written in one step as:

$$\mathbf{u}^{p+1} = T\mathbf{u}^{p} + k \tag{6.73}$$

where T_ρ is the iteration matrix given as:

$$T_\rho = (W+\rho I)^{-1}(\rho I-V-H)(V+\rho I)^{-1}(\rho I-W-V)(H+\rho I)^{-1}(\rho I-W-V) \tag{6.74}$$

and k is given as:

$$\begin{aligned}
k &= (W+\rho I)^{-1}\{(\rho I-H-V)(V+\rho I)^{-1}[(\rho I-H-W)(H+\rho I)^{-1}+I]+I\}\mathbf{b} \\
&= (W+\rho I)^{-1}\{(\rho I-H-V)(V+\rho I)^{-1}[-I-(W+\rho I)(H+\rho I)^{-1} \\
&\quad +3\rho(H+\rho I)^{-1}+I]+I\}\mathbf{b} \\
&= (W+\rho I)^{-1}\{(\rho I-H-V)[-(V+\rho I)^{-1}(W+\rho I)(H+\rho I)^{-1} \\
&\quad +3\rho(V+\rho I)^{-1}(H+\rho I)^{-1}]+I\}\mathbf{b} \\
&= (W+\rho I)^{-1}\{(3\rho I-(H+\rho I)-(V+\rho I))[-(V+\rho I)^{-1}(W+\rho I)(H+\rho I)^{-1} \\
&\quad +3\rho(V+\rho I)^{-1}(H+\rho I)^{-1}]+I\}\mathbf{b} \\
&= (W+\rho I)^{-1}\{[-3\rho(V+\rho I)^{-1}(W+\rho I)(H+\rho I)^{-1}+(V+\rho I)^{-1}(W+\rho I) \\
&\quad +(W+\rho I)(H+\rho I)^{-1}+9\rho^2(V+\rho I)^{-1}(H+\rho I)^{-1}-3\rho(V+\rho I)^{-1} \\
&\quad -3\rho(H+\rho I)^{-1}]+I\}\mathbf{b} \\
&= \{-3\rho(V+\rho I)^{-1}(H+\rho I)^{-1}+(V+\rho I)^{-1}+(H+\rho I)^{-1}+(W+\rho I)^{-1} \\
&\quad -3\rho(W+\rho I)^{-1}(V+\rho I)^{-1}+9\rho^2(V+\rho I)^{-1}(H+\rho I)^{-1}(W+\rho I)^{-1} \\
&\quad -3\rho(W+\rho I)^{-1}(H+\rho I)^{-1}\}\mathbf{b}
\end{aligned} \tag{6.75}$$

To prove consistency for the above iterative procedure we need (by theorem 3.1) to prove that $(I-T_\rho)A^{-1}\mathbf{b} = \mathbf{k}$. From (6.74) and using the relation $(A = H+V+W)$

we can write:

$$
\begin{aligned}
I - T_\rho &= I - (W+\rho I)^{-1}((W+\rho I) - A)(V+\rho I)^{-1}((V+\rho I) - A) \\
&\quad (H+\rho I)^{-1}((H+\rho I) - A) \\
&= I - [I - (W+\rho I)^{-1}A][I - (V+\rho I)^{-1}A][I - (H+\rho I)^{-1}A] \\
&= I - [(I - (W+\rho I)^{-1}A)][I - (V+\rho I)^{-1}A - (H+\rho I)^{-1}A \\
&\quad +(V+\rho I)^{-1}(H+\rho I)^{-1}A^2] \\
&= I - [I - (H+\rho I)^{-1}A - (V+\rho I)^{-1}A - (W+\rho I)^{-1}A \\
&\quad +(W+\rho I)^{-1}(H+\rho I)^{-1}A^2 + (W+\rho I)^{-1}(V+\rho I)^{-1}A^2 \\
&\quad +(H+\rho I)^{-1}(V+\rho I)^{-1}A^2 \\
&\quad -(W+\rho I)^{-1}(V+\rho I)^{-1}(H+\rho I)^{-1}A^3] \qquad (6.76)
\end{aligned}
$$

Further expansions of the last term of (6.76) and eliminations lead to :

$$
\begin{aligned}
I - T_\rho &= (H+\rho I)^{-1}A + (V+\rho I)^{-1}A + (W+\rho I)^{-1}A \\
&\quad -(W+\rho I)^{-1}(H+\rho I)^{-1}A^2 \\
&\quad -(W+\rho I)^{-1}(V+\rho I)^{-1}A^2 - (H+\rho I)^{-1}(V+\rho I)^{-1}A^2 \\
&\quad +(W+\rho I)^{-1}(V+\rho I)^{-1}(H+\rho I)^{-1}A^2[(H+\rho I) \\
&\quad +(V+\rho I) + (W+\rho I) - 3\rho I)] \\
&= (H+\rho I)^{-1}A + (V+\rho I)^{-1}A + (W+\rho I)^{-1}A - (W+\rho I)^{-1}(H+\rho I)^{-1}A^2 \\
&\quad -(W+\rho I)^{-1}(V+\rho I)^{-1}A^2 - (H+\rho I)^{-1}(V+\rho I)^{-1}A^2 \\
&\quad +(W+\rho I)^{-1}(H+\rho I)^{-1}A^2 + (W+\rho I)^{-1}(V+\rho I)^{-1}A^2 \\
&\quad +(H+\rho I)^{-1}(V+\rho I)^{-1}A^2 - 3\rho(H+\rho I)^{-1}(V+\rho I)^{-1}(W+\rho I)^{-1}A^2 \\
&= (H+\rho I)^{-1}A + (V+\rho I)^{-1}A + (W+\rho I)^{-1}A \\
&\quad -3\rho(H+\rho I)^{-1}(V+\rho I)^{-1}(W+\rho I)^{-1}A^2 \\
&= [(H+\rho I)^{-1} + (V+\rho I)^{-1} + (W+\rho I)^{-1} \\
&\quad +9\rho^2(H+\rho I)^{-1}(V+\rho I)^{-1}(W+\rho I)^{-1} - 3\rho(W+\rho I)^{-1}(H+\rho I)^{-1} \\
&\quad -3\rho(W+\rho I)^{-1}(V+\rho I)^{-1} \\
&\quad -3\rho(H+\rho I)^{-1}(V+\rho I)^{-1}]A \qquad (6.77)
\end{aligned}
$$

Therefore from (6.77) and (6.75) we can see that:

$$(I - T_\rho)A^{-1}\mathbf{b} = \mathbf{k} \tag{6.78}$$

Hence this ADI method is consistent.

To prove also that this iterative procedure is *reciprocally consistent* we need by theorem 3.2 to show that $(I - T_\rho)$ is nonsingular.

This however does not follow in a staightforward manner from equations (6.76) and (6.77).

Chapter 7

Further applications of the AGE-1D and EAD methods for coupled systems

7.1 Physical Background

The equations governing fluid motion through a control volume V fixed in space and time may be derived from Newton's second law of motion which require that the rate of change of the linear momentum be equal to the sum of the forces applied. This can be expressed by the following equation:

$$\int_V \varrho \frac{d\mathbf{v}}{dt} dV = \sum F \tag{7.1}$$

i.e " mass $\times$ acceleration = force", where ϱ is the density of the fluid, and $\mathbf{v}$ is the velocity vector having components u,v, and w in the cartesian coordinates $(\vec{i}x, \vec{j}y, \vec{k}z)$. $\sum F$ sums the contribution from all forces acting on the surface of the control volume or throughout the volume. The most common of these forces are gravity, viscosity, pressure gradient and force due to the rotation of the earth (i.e the coriolis force).

Different types of the equations may be derived from (7.1) depending on the nature of the flow. Of particular importance are the:

1. The Euler equations: These apply strictly to inviscid flows. This is a category which covers a wide range of flows where the viscosity forces are negligible compared to other acting forces (e.g the barotropic flow in the 'free atmosphere' away from the effect of the surface of the earth).

2. The Navier-Stokes equations: These account also for the contribution of the viscosity or stress forces to the rate of change of the linear momentum. These equations also have a wide range of applications in modelling different flows (e.g the flow around a streamlined body immersed in fluid, and the atmospheric flow in the Planetary Boundary Layer (i.e the layer of atmosphere close to the earth's surface.)

These equations can be found in ([24] chapter 11) among various other specialised references and are not given here.

Different types of equations can be obtained through various simplifications of the Euler and Navier-Stokes equations to model certain flows based on appropriate assumptions that can be made about the flow (such as about the scale of contribution of some forces compared to others, and assumptions about the compressibilty of the fluid) as well as on whether the model is one, two, or three dimensional. In the following sections we consider some of these simplified models. We apply the AGE-1D method to solve the momentum equations of a simple idealized 1D model of the Planetary Boundary Layer, in the steady state. We also apply the EAD method for solving a linearized version of the Shallow Water Equations which describes an incompressible, inviscid flow with a free surface.

7.2 The AGE-1D method for an idealized planetary boundary layer model

Introduction

In this section the momentum equations of a simple one dimensional model for an

idealized planetary boundary layer (henceforth referred to as PBL) in its *steady* state is considered.

The PBL is the region close to the ground (up to 1 or 2 km) where the atmosphere is strongly influenced by the presence of the surface. In contrast to the smoother flow above it in the "free atmosphere", the PBL is usually turbulent, and thus the "turbulent diffusion" terms in its momentum equations are significant. The equations for such a model can be given as:

$$\begin{aligned} \frac{\partial \bar{u}}{\partial t} &= -fv_g + f\bar{v} + k_m \frac{\partial^2 \bar{u}}{\partial z^2} \\ \frac{\partial \bar{v}}{\partial t} &= fu_g - f\bar{u} + k_m \frac{\partial^2 \bar{v}}{\partial z^2} \end{aligned} \tag{7.2}$$

where $\bar{u}$ and $\bar{v}$ represent the components of the wind *mean* velocity in the west-east and south-north direction respectively. Similarly u_g and v_g are components of the wind velocity in the upper *free atmosphere* layer. Also k_m is a turbulent diffusion coefficient, and $f = 2\Omega \sin \Phi$ is called the coriolis parameter which is constant at any fixed latitude Φ of the earth, $\Omega = 7.27 \times 10^{-5} rds^{-1}$ is the angular momentum of the earth.

The problem and the AGE-1D method

In the steady state form the above problem is given by the following equations:

$$\begin{aligned} f\bar{v} - fv_g + k_m \frac{d^2 \bar{u}}{dz^2} &= 0 \\ f\bar{u} - fu_g + k_m \frac{d^2 \bar{v}}{dz^2} &= 0 \end{aligned} \tag{7.3}$$

defined over the domain $\Re = [0, L]$.

The model is subject to the following conditions at its lower and upper boundaries:

$$\left.\begin{array}{l} \bar{u} = \bar{v} = 0 \\ \left.\begin{array}{l} \bar{v} = v_g = 0 \\ \bar{u} = u_g \end{array}\right\} \end{array}\right. \begin{array}{c} at\ z = 0 \\ at\ z = L \end{array} \left.\begin{array}{l} \\ \end{array}\right\} \begin{array}{c} (No\ wind\ at\ earth\ surface) \\ wind\ direction\ is\ east - west \\ in\ the\ free\ atmosphere \end{array} \tag{7.4}$$

and has an analytical solution given by:

$$\begin{aligned} \bar{u} &= u_g(1 - \exp \frac{-z}{\delta} \cos \frac{z}{\delta}) - v_g(\exp \frac{-z}{\delta} \sin \frac{z}{\delta}) \\ \bar{v} &= v_g(1 - \exp \frac{-z}{\delta} \cos \frac{z}{\delta}) + u_g(\exp \frac{-z}{\delta} \sin \frac{z}{\delta}) \end{aligned}$$

where $\delta = \sqrt{\frac{2k_m}{f}}$. Equations (7.3) may be rewritten as:

$$A_1 \frac{d^2 w}{dz^2} + B_1 w + C_1 = 0 \tag{7.5}$$

with $w = \begin{bmatrix} \bar{u} \\ \bar{v} \end{bmatrix}$; $A_1 = \begin{bmatrix} k_m & 0 \\ 0 & k_m \end{bmatrix}$; $C_1 = \begin{bmatrix} -fv_g \\ fu_g \end{bmatrix}$; $B_1 = \begin{bmatrix} 0 & f \\ -f & 0 \end{bmatrix}$.

We now discretize equation (7.5) along the z-axis with equally spaced gridpoints p_i $(0 \leq i \leq m+1)$ and gridspacing $h = L/(m+1)$, and apply a central difference approximation to the second order derivative to obtain the following replacement of (7.5):

$$\frac{1}{h^2}A_1 w_{i+1} + (-\frac{2}{h^2}A_1 + B_1)w_i + \frac{1}{h^2}A_1 w_{i-1} + C_1 = 0 \qquad (7.6)$$
$$for \quad 1 \leq i \leq m$$

Multiplying by $\frac{-h^2}{k_m}$ this equation becomes:

$$-w_{i+1} + Ew_i + w_{i-1} = S \qquad (7.7)$$

where $E = \begin{bmatrix} 2a & 2b \\ -2b & 2a \end{bmatrix}$ and $S = \begin{bmatrix} su \\ sv \end{bmatrix}$

with $a = 1$; $b = -\frac{fh^2}{2k_m}$; $su = fv_g h^2/k_m$; and $sv = -fu_g h^2/k_m$.

If (7.7) is applied to all the gridpoints p_i inside $\Re$ we get a bivariate unsymmetric system of difference equations of the form (5.48) where

$$\mathbf{w} = (w_1, w_2, \ldots w_m)^T.$$

To solve this system by the AGE method we split A in the same way as given by (5.50) and (5.51) and apply the AGE algorithm given by the first two equations of equations (5.52).

It can be seen that $(G_1 + sI)$ and $(G_2 + sI)$ can be easily inverted since they are block diagonal and consist only of 2×2 matrices (i.e $G + sI$) and/or 2×2 block matrices (i.e C+sI). We can thus define:

$$(C+sI)^{-1} = \frac{1}{\Delta}\hat{C} = \frac{1}{\Delta}\begin{bmatrix} G+sI & I \\ I & G+sI \end{bmatrix} = \begin{bmatrix} (G+sI)\Delta^{-1} & \Delta^{-1} \\ \Delta^{-1} & (G+sI)\Delta^{-1} \end{bmatrix} \qquad (7.8)$$

where Δ is a nonsingular matrix given as:

$$\Delta = (G+sI)^2 - I^2 = \begin{bmatrix} a+s & b \\ -b & a+s \end{bmatrix}\begin{bmatrix} a+s & b \\ -b & a+s \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} adelc & bdelc \\ -bdelc & adelc \end{bmatrix} \qquad (7.9)$$

with $adelc = (a+s)^2 - b^2 - 1$ and $bdelc = 2(a+s)b$.

We can now write $(G_1 + sI)^{-1}$ and $(G_2 + sI)^{-1}$ (and also define $\hat{G}_1$ and $\hat{G}_2$) as:

$$(G_1 + sI)^{-1} = \frac{1}{\Delta}\hat{G}_1 = \frac{1}{\Delta}\begin{bmatrix} \hat{C} & & & \\ & \hat{C} & & 0 \\ & & \hat{C} & \\ & & & \hat{C} \\ 0 & & \hat{C} & \\ & & & \hat{C} \end{bmatrix} \tag{7.10}$$

and

$$(G_2+sI)^{-1} = \frac{1}{\Delta}\hat{G}_2 = \frac{1}{\Delta}\begin{bmatrix} \Delta(G+sI)^{-1} & & & & \\ & \hat{C} & & 0 & \\ & & \hat{C} & & \\ & & & \hat{C} & \\ & 0 & & \hat{C} & \\ & & & & \Delta(G+sI)^{-1} \end{bmatrix} \tag{7.11}$$

Thus the second equation of (5.52) may be rewritten as:

$$\mathbf{w}^{p+1} = \mathbf{b}' - \hat{G}_2[\underbrace{\Delta^{-2}(G_1 - sI)\hat{G}_1}]\mathbf{y} \tag{7.12}$$

where $\mathbf{b}' = (G_2 + sI)^{-1}\mathbf{b}$.

If we define $\tilde{C} = \Delta^{-2}\hat{C}(C - sI)$ then the underbraced may be replaced by H and equation (7.12) can be written as:

$$\mathbf{w}^{p+1} = \mathbf{b}' - \hat{G}_2 H\mathbf{y} \tag{7.13}$$

where $H = diag(\tilde{C})$.

The use of equation (7.13) saves 2m multiplications per iteration compared to when using the second equation of (5.52) because equation (7.13) preserves the unity coefficients in $\hat{C}$, while in the latter equation these elements are not preserved. The component equations of the above matrix equations are not given here but can be found in the listing of the 'program pbl' given in Appendix 2. We give here the numerical results obtained by applying the AGE method to the above problem.

L=210 ; $h = 10$ $u_g = 10$ $v_g = 0.0$										
$\epsilon = 1.0E-03$ Accelaration Parameter $s = .28$										
RESULTS AFTER 30 iterations										
Anal U	.227	.679	1.129	1.575	2.015	2.449	2.874	3.291	3.698	4.094
Numer U	.227	.679	1.128	1.574	2.014	2.448	2.874	3.290	3.698	4.094
error	.000	.000	.001	.001	.001	.001	.001	.001	.000	.000
Anal V	.222	.635	1.010	1.349	1.653	1.924	2.164	2.374	2.557	2.715
Numer V	.222	.635	1.010	1.349	1.653	1.924	2.164	2.374	2.557	2.715
error	.001	.000	.000	.001	.001	.001	.001	.001	.000	.000

Table 7.1: The absolute errors in the velocity profiles of a 1D model for the planetary boundary layer in the steady state

7.3 The EAD method for the linearized Shallow Water Equations

In this section we solve using the EAD method, the linearized form of the equations describing the flow of an incompressible inviscid fluid in a rectangular basin. These equations are called the Shallow Water Equations. They are of the Euler type of equations and apply to flows where the horizontal components of velocity exhibit wavelike solutions with wavelengths which are much greater than the depth of the fluid, hence the name *Shallow Water* equations.

These equations in their linearized form are given in [48] as :

$$\begin{aligned} \frac{\partial \tilde{u}}{\partial t} &= -g\frac{\partial \tilde{h}}{\partial x} \\ \frac{\partial \tilde{v}}{\partial t} &= -g\frac{\partial \tilde{h}}{\partial y} \\ \frac{\partial \tilde{h}}{\partial t} &= -h_0(\frac{\partial \tilde{u}}{\partial x} - \frac{\partial \tilde{v}}{\partial y}) \end{aligned} \tag{7.14}$$

where $\tilde{u}$ and $\tilde{v}$ are the depth-averaged velocity components in the $x - direction$ (East-West) and the $y - direction$ (South-North) respectively, $\tilde{h}$ is the depth below

the moving water (fluid) surface, h_0 is the depth when the water is at rest, and g is the acceleration of gravity.

The set of equations (7.14) are defined over the square region $\Re = [0 \leq x, y \leq L]$ which fits (in space) exactly one period of the analytical solution given by:

$$\tilde{u} = \frac{1}{4}\sin[(-\sqrt{2gh_0}t + x + y)\frac{2\pi}{L}] \; ; \tag{7.15}$$

$$\tilde{v} = \frac{1}{4}\sin[(-\sqrt{2gh_0}t + x + y)\frac{2\pi}{L}] \; ; \tag{7.16}$$

$$\tilde{h} - h_0 = \frac{\sqrt{2h_0/g}}{4}\sin[(-\sqrt{2gh_0}t + x + y)\frac{2\pi}{L}] \; . \tag{7.17}$$

with the initial and boundary (periodic) conditions taken from the analytical solution.

If we define a new variable $\tilde{\Phi} = \sqrt{\frac{g}{h_0}}\tilde{h}$ then equations (7.14) may be rewritten as:

$$\begin{aligned}
\frac{\partial \tilde{u}}{\partial t} &= -\Phi_0 \frac{\partial \tilde{\Phi}}{\partial x} \\
\frac{\partial \tilde{v}}{\partial t} &= -\Phi_0 \frac{\partial \tilde{\Phi}}{\partial y} \\
\frac{\partial \tilde{\Phi}}{\partial t} &= -\Phi_0 (\frac{\partial \tilde{u}}{\partial x} - \frac{\partial \tilde{v}}{\partial y})
\end{aligned} \tag{7.18}$$

or in vector form as:

$$\frac{\partial \tilde{\mathbf{w}}}{\partial t} = -A\frac{\partial \tilde{\mathbf{w}}}{\partial x} - B\frac{\partial \tilde{\mathbf{w}}}{\partial y} \tag{7.19}$$

where

$$\tilde{\mathbf{w}} = \begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{\Phi} \end{bmatrix} \; ; \; A = \begin{bmatrix} 0 & 0 & \Phi_0 \\ 0 & 0 & 0 \\ \Phi_0 & 0 & 0 \end{bmatrix} \; ; \; and; \; B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & \Phi_0 \\ 0 & \Phi_0 & 0 \end{bmatrix}$$

The analytical solution is rewritten as:

$$\begin{aligned}
\tilde{u} &= \frac{1}{4}\sin[(-\sqrt{2}\Phi_0 t + x + y)\frac{2\pi}{L}] \; ; \\
\tilde{v} &= \frac{1}{4}\sin[(-\sqrt{2}\Phi_0 t + x + y)\frac{2\pi}{L}] \; ; \\
\tilde{\Phi} - \Phi_0 &= \frac{\sqrt{2}}{4}\sin[(-\sqrt{2}\Phi_0 t + x + y)\frac{2\pi}{L}] \; .
\end{aligned}$$

A finite difference approximation to equation (7.19) is given as:

$$(I - P'\delta_x)(1 - Q'\delta_y)\mathbf{W}_{ij}^{n+1} = (I + P'\delta_x)(1 + Q'\delta_y)\mathbf{W}_{ij}^{n} \tag{7.20}$$

where

$$P' = \frac{-k}{4ds}A = \begin{bmatrix} 0 & 0 & e_2 \\ 0 & 0 & 0 \\ e_2 & 0 & 0 \end{bmatrix} \; ; \; and; \; Q' = \frac{-k}{4ds}B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & e_2 \\ 0 & e_2 & 0 \end{bmatrix}$$

with $e_2 = \frac{-k}{4ds}\phi_0$. Equation (7.20) is clearly a perturbation of the Crank- Nicolson type scheme. Gustafsson in ([27]) has derived an ADI scheme for equation (7.19) by splitting (7.20) in the usual Peaceman-Rachford manner to obtain the following two equations:

$$(I - P'\delta_x)\mathbf{W}_{ij}^{n+1*} = (1 + Q'\delta_y)\mathbf{W}_{ij}^{n} \tag{7.21}$$

$$(I - Q'\delta_y)\mathbf{W}_{ij}^{n+1} = (1 + P'\delta_x)\mathbf{W}_{ij}^{n+1*} \tag{7.22}$$

where $\mathbf{W}_{ij}^{n+1*}$ is an intermediate solution. We now difference equation (7.19) using an ADI scheme by [23] which is similar to Gustafsson's scheme, but requires less computations. It is also unconditionally stable and of second order accuracy in time and space as shown in ([23]). This is given as:

$$(I - P'\delta_x)\mathbf{W}_{ij}^{n+1*} = \mathbf{z}_{ij}^{n} \tag{7.23}$$

$$(I - Q'\delta_y)\mathbf{W}_{ij}^{n+1} = 2\mathbf{W}_{ij}^{n+1*} - \mathbf{z}_{ij}^{n} \tag{7.24}$$

where $\mathbf{z}_{ij}^{n} = (I + Q'\delta_y)\mathbf{W}_{ij}^{n} = \begin{bmatrix} z_{1_{ij}} \\ z_{2_{ij}} \\ z_{3_{ij}} \end{bmatrix} = \begin{bmatrix} U_{ij}^{n} \\ V_{ij}^{n} + e_2\delta_y\Phi_{ij}^{n} \\ \Phi_{ij}^{n} + e_2\delta_y V_{ij}^{n} \end{bmatrix}$.

By eliminating $\mathbf{W}_{ij}^{n+1*}$ from the equations (7.23) and (7.24) we obtain equation (7.20).

If we apply each of equations (7.23) and (7.24) to the totality of mesh points with a rowwise ordering we get the following two systems:

$$A_1'\mathbf{x_r} = \mathbf{b_1} \tag{7.25}$$

$$A_2'\mathbf{y_r} = \mathbf{b_2} \tag{7.26}$$

where the unknown vectors $\mathbf{x} = (\mathbf{w}_{1,1}^{n+1*} \ldots \mathbf{w}_{i,1}^{n+1*} \ldots \mathbf{w}_{i,j}^{n+1*} \ldots \mathbf{w}_{m_1,m_2}^{n+1*})^T$ and $\mathbf{y} = (\mathbf{w}_{1,1}^{n+1} \ldots \mathbf{w}_{i,1}^{n+1} \ldots \mathbf{w}_{i,j}^{n+1} \ldots \mathbf{w}_{m_1 m_2}^{n+1})^T$ are the intermediate solution vector and the solution vector at the time level n+1 respectively. The **r** index indicates a rowwise ordering and $\mathbf{b_1}$ and $\mathbf{b_2}$ are known vectors.

The matrix A_1' is a block diagonal square matrix of order $3m_1m_2$ and is given as:

$$A_1' = diag(T_1') \quad , \tag{7.27}$$

where T_1' is the following circulant block-tridiagonal matrix:

$$T_1' = \begin{bmatrix} I & -P' & & & & P' \\ P' & \ddots & \ddots & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & -P' \\ -P' & & & & P' & I \end{bmatrix}_{3m_1 \times 3m_1} \tag{7.28}$$

A_2' is a circulant block-tridiagonal matrix given as:

$$A_2' = \begin{bmatrix} I & -T_2' & & & & T_2' \\ T_2' & \ddots & \ddots & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & -T_2' \\ -T_2' & & & & T_2' & I \end{bmatrix}_{3m_1m_2 \times 3m_1m_2} \tag{7.29}$$

with

$$T_2' = diag(Q') \tag{7.30}$$

The EAD method proceeds to solving systems (7.23) and (7.24) by applying the AGE method. However, because of the form of the matrices P and Q in equations (7.23) and (7.24), only two variables U and Φ, (in equation 7.23) and V and Φ (in equation 7.24) are coupled together on the left hand sides of these equations.

This means that we can decompose each of (7.23) and (7.24) into two simpler difference equations, one *vector* and one *scalar*, given here respectively for (7.23) and (7.24) as:

$$(I - P\delta_x)\mathbf{U}_{i,j}^{n+1*} = \mathbf{U}_{ij}^n + Q\delta_y\mathbf{V}_{ij}^n = \mathbf{zu}_{ij}^n \equiv \begin{bmatrix} z_{1_{ij}} \\ \\ z_{3_{ij}} \end{bmatrix} \tag{7.31}$$

$$V_{ij}^{n+1*} = V_{ij}^n + e_2\delta_y\Phi_{ij}^n \equiv z_{2_{ij}}^n \tag{7.32}$$

and

$$(I - P\delta_y)\mathbf{V}_{i,j}^{n+1} = 2\mathbf{V}_{ij}^{n+1*} - \mathbf{zv}_{ij}^n \equiv \begin{bmatrix} z_{2_{ij}} \\ \\ z_{3_{ij}} \end{bmatrix} \tag{7.33}$$

$$U_{ij}^{n+1} = 2U_{ij}^{n+1*} - z_{1_{ij}}^n \tag{7.34}$$

where **U**, **V**, **zu**, **zv**, P and Q are defined as: $\mathbf{U}_{ij} = \begin{bmatrix} U_{ij} \\ \\ \Phi_{ij} \end{bmatrix}$; $\mathbf{V}_{ij} = \begin{bmatrix} V_{ij} \\ \\ \Phi_{ij} \end{bmatrix}$;

$$\mathbf{zu}_{ij} = \begin{bmatrix} z_{1_{ij}} \\ \\ z_{3_{ij}} \end{bmatrix};\ \mathbf{zv}_{ij} = \begin{bmatrix} z_{2_{ij}} \\ \\ z_{3_{ij}} \end{bmatrix};\ P = \begin{bmatrix} 0 & e_2 \\ \\ e_2 & 0 \end{bmatrix} and\ Q = \begin{bmatrix} 0 & 0 \\ \\ e_2 & 0 \end{bmatrix}$$

Equations (7.32) and (7.34) are explicit in nature, while equations (7.31) and (7.33) lead respectively to the following two systems:

$$A_1\mathbf{x_r} = \mathbf{d_1} \tag{7.35}$$

$$A_2\mathbf{y_r} = \mathbf{d_2} \tag{7.36}$$

where **x** and **y** are now given as: $\mathbf{x} = (\mathbf{u}_{1,1}^{n+1*} \ldots \mathbf{u}_{i,1}^{n+1*} \ldots \mathbf{u}_{i,j}^{n+1*} \ldots \mathbf{u}_{m_1m_2}^{n+1*})^T$ and $\mathbf{y} = (\mathbf{v}_1^{n+1} \ldots \mathbf{v}_{i,1}^{n+1} \ldots \mathbf{v}_{i,j}^{n+1} \ldots \mathbf{v}_{m_1m_2}^{n+1})^T$ and **u** and **v** represent the computed values of **U** and **V**. The Known vectors $\mathbf{d_1}$ and $\mathbf{d_2}$ are given as: $\mathbf{d_1} = (\mathbf{zu}_{1,1} \ldots \mathbf{zu}_{i,1} \ldots \mathbf{zu}_{i,j} \ldots \mathbf{zu}_{m_1m_2})^T$ and $\mathbf{d_2} = (\mathbf{zv}_{1,1} \ldots \mathbf{zv}_{i,1} \ldots \mathbf{zv}_{i,j} \ldots \mathbf{zv}_{m_1m_2})^T$. The **r** index indicates a rowwise ordering.

The matrices A_1 and A_2 have the same structure as A_1' and A_2' given by equations (7.27 ... 7.30) but by replacing each of P' and Q' by P.

i.e we now have $A_1 = diag(T_1)$,

where T_1 is the following circulant block-tridiagonal matrix:

$$T_1 = \begin{bmatrix} I & -P & & & & P \\ P & \ddots & \ddots & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & -P \\ -P & & & & P & I \end{bmatrix}$$

A_2 is a circulant block-tridiagonal matrix given as:

$$A_2 = \begin{bmatrix} I & -T_2 & & & & T_2 \\ T_2 & \ddots & \ddots & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & -T_2 \\ -T_2 & & & & T_2 & I \end{bmatrix}_{2m_1m_2 \times 2m_1m_2}$$

where $T_2 = diag(P)$ and I is an identity matrix. Equation (7.32) leads to an explicit system of difference equations which may be solved concurrently with the new system (7.35). After that we can solve concurrently (7.36) and the explicit system of difference equations which arises from (7.34).

The two systems (7.35) and (7.36) are now simpler than (7.25) and (7.26) and are, according to the EAD method, solved using the AGE-1D iterative procedure given as:

$$(G_1 + sI)\mathbf{x}^{p+\frac{1}{2}} = \mathbf{d_1} - (G_2 - sI)\mathbf{x}^p \equiv g_1 \tag{7.37}$$

$$\mathbf{x}^{p+1} = (G_2 + sI)^{-1}[\mathbf{d_1} - \underbrace{(G_1 - sI)(G_1 + sI)^{-1}}\, g_1] \tag{7.38}$$

$$(G_3 + sI)\mathbf{y}^{p+\frac{1}{2}} = \mathbf{d_2} - (G_4 - sI)\mathbf{y}^p \equiv g_2 \tag{7.39}$$

$$\mathbf{y}^{p+1} = (G_4 + sI)^{-1}[\mathbf{d_2} - \underbrace{(G_3 - sI)(G_3 + sI)^{-1}}\, g_2] \tag{7.40}$$

where s is the acceleration parameter of the AGE method. Here we have split the matrices A_1 and A_2 such that:

$$A_1 = G_1 + G_2 \tag{7.41}$$

$$A_2 \;=\; G_3 + G_4 \tag{7.42}$$

where $G_1 = diag(\tilde{G}_1)$ and $G_2 = diag(\tilde{G}_2)$, with $\tilde{G}_1$ and $\tilde{G}_2$ being given as:

$$\tilde{G}_1 = \begin{pmatrix} \frac{1}{2}I & & & P \\ & C & & \\ & & \ddots\ 0 & \\ & & \ddots & \\ & 0 & & C \\ -P & & & \frac{1}{2}I \end{pmatrix}_{2m_1 \times 2m_1} \tag{7.43}$$

and

$$\tilde{G}_2 = \begin{pmatrix} C & & & \\ & \ddots & 0 & \\ & & \ddots & \\ & & \ddots & \\ & 0 & & \ddots \\ & & & C \end{pmatrix}_{2m_1 \times 2m_1} \tag{7.44}$$

where $C = \begin{bmatrix} \frac{1}{2}I & -P \\ P & \frac{1}{2}I \end{bmatrix}$.

G_3, and G_4 are given as:

$$G_3 = \begin{pmatrix} \hat{R} & & & \hat{P} \\ & G & & \\ & & \ddots\ 0 & \\ & & \ddots & \\ & 0 & & G \\ -\hat{P} & & & \hat{R} \end{pmatrix}_{2m_1m_2 \times 2m_1m_2} \tag{7.45}$$

and

$$G_4 = \begin{pmatrix} G & & & \\ & \ddots & 0 & \\ & & \ddots & \\ & & & \ddots \\ & 0 & & G \end{pmatrix}_{2m_1m_2 \times 2m_1m_2} \tag{7.46}$$

where $\hat{R} = diag(\frac{1}{2}I)$, and $\hat{P} = diag(P)$ is of order $2m_2$. G is given as:

$$G = \begin{bmatrix} \hat{R} & -\hat{P} \\ \hat{P} & \hat{R} \end{bmatrix}$$

The matrices $(G_i + sI)$ (i=1 ... 4) can now be easily inverted, since we have:

$$(C + sI)^{-1} = \frac{1}{\Delta_1} \begin{bmatrix} (\frac{1}{2} + s)I & P \\ -P & (\frac{1}{2} + s)I \end{bmatrix} = \begin{bmatrix} (\frac{1}{2} + s)\Delta_1^{-1} & P\Delta_1^{-1} \\ -P\Delta_1^{-1} & (\frac{1}{2} + s)\Delta_1^{-1} \end{bmatrix} \tag{7.47}$$

where Δ_1 is a matrix given as:

$$\Delta_1 = (\frac{1}{2}I + sI)^2 + P^2 = \begin{bmatrix} (0.5+s) & 0 \\ 0 & (0.5+s) \end{bmatrix} \begin{bmatrix} (0.5+s) & 0 \\ 0 & (0.5+s) \end{bmatrix} + \begin{bmatrix} 0 & e_2 \\ e_2 & 0 \end{bmatrix} \begin{bmatrix} 0 & e_2 \\ e_2 & 0 \end{bmatrix} = \begin{bmatrix} a_1 & 0 \\ 0 & a_1 \end{bmatrix}$$

with $a_1 = (0.5 + s)^2 + e^2 = 0.25 + s^2 + s + e_2^2$.

Therefore

$$(C + sI)^{-1} = \begin{pmatrix} \begin{bmatrix} e_3 & 0 \\ 0 & e_3 \end{bmatrix} & \begin{bmatrix} 0 & -e_4 \\ -e_4 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & e_4 \\ e_4 & 0 \end{bmatrix} & \begin{bmatrix} e_3 & 0 \\ 0 & e_3 \end{bmatrix} \end{pmatrix}$$

where $e_3 = \frac{(0.5+s)}{a_1}$ and $e_4 = \frac{-e_2}{a_1}$.

We can now write $(\tilde{G}_1 + sI)^{-1}$ and $(\tilde{G}_2 + sI)^{-1}$ as:

$$(\tilde{G}_1+sI)^{-1} = \begin{bmatrix} (\frac{1}{2}+s)\Delta_1^{-1} & & & -P\Delta_1^{-1} \\ & (C+sI)^{-1} & 0 & \\ & & \ddots & \\ & 0 & (C+sI)^{-1} & \\ P\Delta_1^{-1} & & & (\frac{1}{2}+s)\Delta_1^{-1} \end{bmatrix} \quad (7.48)$$

and

$$(\tilde{G}_2+sI)^{-1} = \begin{bmatrix} (C+sI)^{-1} & & & \\ & (C+sI)^{-1} & 0 & \\ & & \ddots & \\ & & (C+sI)^{-1} & \\ 0 & & & (C+sI)^{-1} \end{bmatrix} \quad (7.49)$$

We now have:

$$(G_1 + sI)^{-1} = diag[(\tilde{G}_1 + sI)^{-1}],\ and\ (G_2 + sI)^{-1} = diag[(\tilde{G}_2 + sI)^{-1}]$$

Similarly we have:

$$(G+sI)^{-1} = \begin{bmatrix} (0.5+s)\Delta_2^{-1} & \hat{P}\Delta_2^{-1} \\ -\hat{P}\Delta_2^{-1} & (0.5+s)\Delta_2^{-1} \end{bmatrix} \quad (7.50)$$

where Δ_2 is a matrix given as:

$$\Delta_2 = (\hat{R}+sI)^2 + \hat{P}^2 = diag(\Delta_1) \quad (7.51)$$

We can now write $(G_3 + sI)^{-1}$ and $(G_4 + sI)^{-1}$ as:

$$(G_3+sI)^{-1} = \begin{bmatrix} (0.5+s)\Delta_2^{-1} & & & -\hat{P}\Delta_2^{-1} \\ & (G+sI)^{-1} & 0 & \\ & & \ddots & \\ & 0 & (G+sI)^{-1} & \\ \hat{P}\Delta_2^{-1} & & & (0.5+s)\Delta_2^{-1} \end{bmatrix} \quad (7.52)$$

and

$$(G_4+sI)^{-1} = \begin{bmatrix} (G+sI)^{-1} & & & \\ & (G+sI)^{-1} & 0 & \\ & & \ddots & \\ & 0 & & (G+sI)^{-1} \end{bmatrix} \quad (7.53)$$

The matrices $(G_i - sI), i = 1, \ldots 4$ are given respectively by equations (7.43 ... 7.46) but by replacing the value $\frac{1}{2}$ along the diagonals by e_1, where $e_1 = 0.5 - s$. Thus instead of C in (7.43) and (7.44) and G in (7.45) and (7.46) we have:

$$C - sI = \begin{bmatrix} e_1 I & -P \\ P & e_1 I \end{bmatrix} \; ; \; G - sI = \begin{bmatrix} e_1 I & -\hat{P} \\ \hat{P} & e_1 I \end{bmatrix} \; ;$$

Let us also define:

$$\hat{D} \;=\; (C - sI)(C + sI)^{-1}$$

$$= \left(\begin{bmatrix} e_1 & 0 \\ 0 & e_1 \end{bmatrix} \; \begin{bmatrix} 0 & -e_2 \\ -e_2 & 0 \end{bmatrix} \right) \times \left(\begin{bmatrix} e_3 & 0 \\ 0 & e_3 \end{bmatrix} \; \begin{bmatrix} 0 & -e_4 \\ -e_4 & 0 \end{bmatrix} \right)$$
$$\begin{bmatrix} 0 & e_2 \\ e_2 & 0 \end{bmatrix} \quad \begin{bmatrix} e_1 & 0 \\ 0 & e_1 \end{bmatrix} \qquad \begin{bmatrix} 0 & e_4 \\ e_4 & 0 \end{bmatrix} \quad \begin{bmatrix} e_3 & 0 \\ 0 & e_3 \end{bmatrix}$$

$$\Rightarrow \hat{D} \;=\; \begin{bmatrix} E_1 & -E_2 \\ E_2 & E_1 \end{bmatrix}$$

where $E_1 = \begin{bmatrix} e_5 & 0 \\ 0 & e_5 \end{bmatrix}$; $E_2 = \begin{bmatrix} 0 & e_6 \\ e_6 & 0 \end{bmatrix}$;

and

$e_5 = e_1 e_3 - e_2 e_4$; $e_6 = e_1 e_4 + e_2 e_3$.

The underbraced products in (7.38) and (7.40) may be replaced respectively by

the matrices D_1 and D_2 which are given as: $D_1 = diag(\tilde{D}_1)$. $\tilde{D}_1$ is given by:

$$\tilde{D}_1 = \begin{bmatrix} E_1 & & & & E_2 \\ & \hat{D} & & 0 & \\ & & \hat{D} & & \\ & & & \hat{D} & \\ & 0 & & \hat{D} & \\ -E_2 & & & & E_1 \end{bmatrix}$$

and

$$D_2 = \begin{bmatrix} \hat{E}_1 & & & & \hat{E}_2 \\ & \tilde{D}_2 & & 0 & \\ & & \tilde{D}_2 & & \\ & & & \tilde{D}_2 & \\ & 0 & & \tilde{D}_2 & \\ -\hat{E}_2 & & & & \hat{E}_1 \end{bmatrix}$$

where $\hat{E}_1 = diag(E_1)$ and $\hat{E}_2 = diag(E_2)$ are diagonal square matrices of order $2m_1$. The matrix $\tilde{D}_2$ is given as:

$$\tilde{D}_2 = \begin{bmatrix} \hat{E}_1 & -\hat{E}_2 \\ \hat{E}_2 & \hat{E}_1 \end{bmatrix}$$

Now the component level algorithms for (7.37 ...7.40) follow in a straight forward manner.

The following table shows the results from an experiment where in equations (7.18) we choose $\Phi_0 = 28$, which correspond to a depth of the basin $h_0 = 78.4m$ in (7.14) where the $g = 10ms^{-2}$.

The gridpoints spacing is $h = 60000m$ and the basin is a square of dimension $L = 9 \times h = 54,0000m$. The time step is taken to be $\Delta t = 60$ seconds.

The AGE-1D algorithm in the EAD method converges in 2 iterations.

Convergence at every time step occurred after 2/2 AGE-1D iterations					
$\Delta x_1 = \Delta x_2 = 60000.0m,\ \Delta t = 60sec,\ \varepsilon = 10^{-6}$					
$x_2 = 1.8E+05$					
$x_1 =$	6.0 E+04	1.8E+05	3.0E+05	4.2E+05	5.4E+05
Anal. u	.14695	-.17366	-.20726	.10168	.24257
Numer. u	.11781	-.17769	-.20412	.10672	.24278
error u	2.9E-02	4.0E-03	3.1E-03	5.0E-03	2.1E-04
Anal. v	.14695	-.17366	-.20726	.10168	.24257
Numer. v	.14404	-.17769	-.20420	.10677	.24128
error v	2.9E-03	4.0E-03	3.1E-03	5.1E-03	1.3E-03
Anal. ϕ	28.492	28.039	27.991	28.428	28.627
Numer. ϕ	28.467	28.033	27.996	28.435	28.628
error ϕ	2.5E-02	5.7E-03	4.4E-03	7.2E-03	4.3E-04
$x_2 = 4.8E+05$					
$x_1 =$	6.0 E+04	1.8E+05	3.0E+05	4.2E+05	5.4E+05
Anal. u	-.06891	.22470	.14695	-.17366	-.20726
Numer. u	-.03185	.22718	.14243	-.17761	-.20564
error u	3.7E-02	2.5E-03	4.5E-03	3.9E-03	1.6E-03
Anal. v	-.06891	.22470	.14695	-.17366	-.20726
Numer. v	-.06446	.22696	.14240	-.17750	-.20409
error v	4.4E-03	2.3E-03	4.5E-03	3.8E-03	3.2E-03
Anal. ϕ	28.187	28.602	27.492	28.039	28.991
Numer. ϕ	28.218	28.605	27.486	28.033	28.993
error ϕ	3.1E-02	3.4E-03	6.4E-03	5.5E-03	2.3E-03

Table 7.2: The three components of the solution to the shallow water equations as obtained by the EAD method after 10 time steps i.e at t=600s

Chapter 8

Conclusions and suggestions for Further work

8.1 Conclusions

In this thesis the AGE-1D method has been developed and shown to be applicable to a wide range of problems with acceptable results.

Of particular importance was the establishment of formula (5.43) for the optimum acceleration parameter of the method for linear systems arising from the use of central difference operators for the advection equation, and establishing the convergence of the method for the solution of block symmetric systems such as equation (5.48) in section 5.5. Also, the conditions for a possible acceleration of the method by using Chebyshev polynomials were determined. These are important contributions to the justification of the method and to the widening of its application. The application of the AGE-1D algorithm is still however restricted to tridiagonal and block tridiagonal systems, and suggestions for appropriate splittings which maintain the AGE concept of forming easily invertible matrices need to be considered for other systems (see section 8.2).

As for AGE-2D and AGE-3D iterative methods which require a relatively large

amount of computational work, it is worth here to mention that we did question the wisdom behind having four and six subiterations respectively. The formulation of these methods was in some analogy to the formulation of ADI methods in general and the ADI Douglas-Rachford finite difference schemes of (4.12) and (4.18) in particular.

For these ADI schemes it is necessary to have p number of equations (p is the number of space dimensions of the problem e.g 2 and 3 for (4.12) and (4.18) respectively) whereby in each equation the solution is advanced implicitly along one dimension to an intermediate value, and it is the combination of these equations which forms the respective ADI *finite difference* scheme which is usually a perturbation of the Crank-Nicholson or the implicit scheme.

In the formulation of an iterative method, however we only require an algorithm which has an easily invertible splitting matrix, and which is completely consistent and convergent. The algorithm does not have to have a number of subiterations which is equal to the number of the constituent matrices into which we split the coefficient matrix A.

This means that an iterative method can be made of only the first equation of (4.98) or (6.32) alone or from a combination of the first equation and any one of the remaining equations. We shall refer to these methods as the *Reduced AGE-2D* and *Reduced AGE-3D* algorithms or RAGE-2D and RAGE-3D. These methods can be shown to be consistent. They also cost less multiplication and addition operations for each full iteration.

Such an alternative course in the formulation of iterative methods for two and three dimensional problems appeared to be the remedy for the large amount of computational work involved in the AGE-2D and AGE-3D algorithms. This course however was tested by solving the normalized system (4.93). The right hand vector was set to the value which makes the solution vector have unity elements. The number of iterations and computational work required by the AGE-2D and all the other RAGE-2D methods were compared and the results are given in table 8.1 below.

The method		$r\theta$					cost per iteration
		0.6	1.1	1.6	2.1	2.6	
AGE-2D	*NIT*	11	13	15	16	17	$16mn\times$ & $15mn+$
1− > 4	s^*	2.5	2.2	1.9	1.6	1.6	
Total No. in (mn) of + & × operations		341	403	465	496	527	
RAGE-2D	*NIT*	23	35	44	51	58	$4mn\times$ &$6mn+$
1;1	s^*	8.6	7.7	7.1	6.8	6.8	
Percentage w.r.t. AGE-2D of Total + & × ops.		67%	87%	95%	103%	110%	
RAGE-2D	*NIT*	18	26	33	39	43	$8mn\times$ & $9mn+$
1;2	s^*	6.5	5.3	5.0	5.0	4.7	
Percentage w.r.t. AGE-2D of Total + & × ops.		90%	110%	121%	134%	139%	
RAGE-2D	*NIT*	17	25	32	37	42	$8mn\times$ & $9mn+$
1;3	s^*	5.8	5.3	5.0	4.7	4.7	
Percentage w.r.t. AGE-2D of Total + & × ops.		85%	105%	117%	127%	135%	
RAGE-2D	*NIT*	18	26	34	38	44	$8mn\times$ & $9mn+$
1;4	s^*	6.2	5.4	5.2	4.9	4.9	
Percentage w.r.t. AGE-2D of Total + & × ops.		90%	110%	124%	130%	142%	

Table 8.1: A comparison between the computational requirements of the AGE-2D and the RAGE-2D methods

The RAGE-2D methods denoted by 1;2, 1;3, and 1;4 have algorithms which consist of a combination of the first equation and the second, the third and the fourth equations of (4.98) respectively. The algorithm for the RAGE-2D method which is denoted by 1; consist only of the first equation in (4.98). Table 8.1 shows that on the whole the AGE-2D method costs less in terms of multiplication and addition operations then any of the RAGE-2D methods, increasingly so as the mesh ratio r increases for a fixed θ. The exception is when the mesh ratio is small where still the best of the RAGE-2D methods (i.e RAGE-2D 1;) does not offer very large savings.

The fact that the AGE-2D method requires less iterations than any of the RAGE-2D methods (see table 8.1) suggests that having a number of subiterations equal to the number of G_i matrices does perform the expected function of spreading the errors across the solution vector which in turn minimizes the required number of iterations to achieve convergence.

The EAD methods on the other hand have achieved large savings (45 % to 83 %) in the computational work as compared to the AGE-2D and AGE-3D methods. These savings are evident throughout tables 6.1... 6.14. The savings consistently increase as the tolerance ϵ of the criteria for convergence decreases. The EAD method also has consistently better accuracies in the results shown in the above mentioned tables, and also in the comparison with the AGE-3D methods as shown in table 6.15.

By comparing tables 6.12 and 6.14, it becomes clear that the EAD method is a little more efficient compared to the EAD *fully iterative* method in solving the two dimensional advection problem although the two methods have comparable accuracies.

The EAD fully iterative method however represents a novel approach in solving elliptic problems as illustrated in section 6.4. The main advantage is in making use of the fact that the intermediate values of the solution derived through an ADI iterative method need not be obtained with large accuracies. This gives a greater flexibilty in employing a parallel algorithm such as the AGE-1D in obtaining such intermediate values at a relatively low cost.

The convergence of the outer ADI iteration is established in section 6.5.1 and the

convergence of the inner AGE-1D algorithm is given earlier in chapters 4 and 5. This assures the convergence of the EAD fully iterative method for the elliptic problem of section 6.4 and the hyperbolic problem of subsection 6.2.2.

The conditional convergence of the outer ADI type iterative procedure has also been established in subsection 6.5.2. for the EAD *fully iterative* method for the three dimensional heat conduction problem. The convergence condition however requires that relatively high values of the acceleration parameter be used which may be undesirable because it can lead to high rounding errors being created and thus affect greatly the accuracy of the method. Even more worrying is the difficulty to prove the *complete consistency* of the outer ADI iterative procedure in section 6.6. This calls for further consideration of the method and its formulation before its application can be recommended.

In chapter 7, the application of the AGE-1D and the EAD methods were shown to extend easily to the solution of multivariate systems of equations while maintaining the simplicity of the inversion of the G_i matrices involved in these problems. The large number of iterations needed for convergence in table 7.2 of section 7.2 is due to the fact that the problem considered is of elliptic type. If the problem considered was that of solving the unsteady state equation (7.2) with the proper initial conditions rather than solving (7.3), the AGE-1D method would have definitely converged in a much smaller number of iterations.

8.2 Suggestions for further work

It can be noticed that the AGE-1D method is limited so far to tridiagonal and block-tridiagonal systems. This in turn limits the application of the EAD method of which the AGE-1D algorithm is a major component. Here we suggest the following AGE-1D type splitting for quindiagonal systems.

Consider the system:

$$A\mathbf{u} = \mathbf{b} \tag{8.1}$$

where

$$\dot{A} = \begin{pmatrix} d & b & c & & & & & & & \\ b & a & b & c & & & & & & \\ c & b & a & b & c & & & & & \\ & c & b & a & b & c & & & & \\ & & c & b & a & b & c & & & \\ & & & c & b & a & b & c & & \\ & & & & c & b & a & b & c & \\ & & & & & c & b & a & b & \\ & & & & & & c & b & d & \end{pmatrix}$$

In a manner consistent with the splitting strategy of AGE-1D we can split A into:

$$A = G_1 + G_2 + G_3$$

whereby G_1, G_2 and G_3 consist now mainly of 3x3 submatrices and are given as:

$$G_1 = \begin{pmatrix} d/3 & b/2 & c & & & & & & & \\ b/2 & a/3 & b/2 & & & & & & & \\ c & b/2 & a/3 & & & & & & & \\ & & & a/3 & b/2 & c & & & & \\ & & & b/2 & a/3 & b/2 & & & & \\ & & & c & b/2 & a/3 & & & & \\ & & & & & & a/3 & b/2 & c & \\ & & & & & & b/2 & a/3 & b/2 & \\ & & & & & & c & b/2 & a/3 & \\ & & & & & & & & & d/3 \end{pmatrix}$$

and

$$G_2 = \begin{pmatrix} d/3 & b/2 & & & & & & & \\ b/2 & a/3 & & & & & & & \\ & & a/3 & b/2 & c & & & & \\ & & b/2 & a/3 & b/2 & & & & \\ & & c & b/2 & a/3 & & & & \\ & & & & & a/3 & b/2 & c & \\ & & & & & b/2 & a/3 & b/2 & \\ & & & & & c & b/2 & a/3 & \\ & & & & & & & a/3 & b/2 \\ & & & & & & & b/2 & d/3 \end{pmatrix}$$

and

$$G_3 = \begin{pmatrix} d/3 & & & & & & & & & \\ & a/3 & b/2 & & & & & & & \\ & b/2 & a/3 & & & & & & & \\ & & & a/3 & b/2 & c & & & & \\ & & & b/2 & a/3 & b/2 & & & & \\ & & & c & b/2 & a/3 & & & & \\ & & & & & & a/3 & b/2 & c & \\ & & & & & & b/2 & a/3 & b/2 & \\ & & & & & & c & b/2 & a/3 & \\ & & & & & & & & & d/3 \end{pmatrix}$$

The 3×3 submatrix of G_1, G_2, and G_3 has the form of:

$$C = \begin{bmatrix} x & y & z \\ y & x & y \\ z & y & x \end{bmatrix}$$

and can still be inverted easily. Its inverse has the simple form of:

$$C = \begin{bmatrix} \frac{(x^2-y^2)}{\alpha\beta} & \frac{-y}{\beta} & \frac{y^2-xz}{\alpha\beta} \\ \frac{-y}{\beta} & \frac{(x^2-z^2)}{\alpha\beta} & \frac{-y}{\beta} \\ \frac{y^2-xz}{\alpha\beta} & \frac{-y}{\beta} & \frac{(x^2-y^2)}{\alpha\beta} \end{bmatrix}$$

where $\beta = x^2 + xz - 2y^2$ and $\alpha = x - z$

An AGE-1D algorithm for solving system (8.1) can have the form:

$$(G_1 + sI)\mathbf{u}^{\mathbf{p}+\frac{\mathbf{1}}{\mathbf{3}}} = \mathbf{b} - (\mathbf{G_2} + \mathbf{G_3} - \mathbf{sI})\mathbf{u}^{\mathbf{p}} \tag{8.2}$$

$$(G_2 + sI)\mathbf{u}^{\mathbf{p}+\frac{\mathbf{2}}{\mathbf{3}}} = \mathbf{b} - (\mathbf{G_1} + \mathbf{G_3} - \mathbf{sI})\mathbf{u}^{\mathbf{p}+\frac{\mathbf{1}}{\mathbf{3}}} \tag{8.3}$$

$$(G_3 + sI)\mathbf{u}^{\mathbf{p+1}} = \mathbf{b} - (\mathbf{G_1} + \mathbf{G_2} - \mathbf{sI})\mathbf{u}^{\mathbf{p}+\frac{\mathbf{2}}{\mathbf{3}}} \tag{8.4}$$

For the EAD *fully iterative* method, it can be pointed out that the convergence analyses given in section 6.5 falls short of determining what combination of values should be assigned for the acceleration parameters of the outer ADI and inner AGE-1D procedures respectively. This is because our analysis was based on establishing the convergence of the EAD *fully iterative* method through the analysis of the convergence of the outer ADI and inner AGE-1D iterative procedures separately.

There is however another route which can lead to a more quantitative assesment of the convergence of the method and may lead to a formula defining the relation between the acceleration parameter ρ of the outer ADI iterative procedure and the acceleration parameter s of the inner AGE-1D iteration for optimal convegence. Such a route is based on analysing the iteration matrix T of the EAD *fully iterative* method which for two and three dimensional problems respectively is given by:

$$\begin{aligned} T(\rho, s) &= (G_2 + sI)^{-1}(G_1 - sI)(G_1 + sI)^{-1}(G_2 - sI)(G_4 + sI)^{-1} \\ &\times (G_3 - sI)(G_3 + sI)^{-1}(G_4 - sI) \end{aligned} \tag{8.5}$$

$$\begin{aligned} T(\rho, s) &= (G_2 + sI)^{-1}(G_1 - sI)(G_1 + sI)^{-1}(G_2 - sI) \\ &\times (G_4 + sI)^{-1}(G_3 - sI)(G_3 + sI)^{-1}(G_4 - sI) \\ &\times (G_6 + sI)^{-1}(G_5 - sI)(G_5 + sI)^{-1}(G_6 - sI) \end{aligned} \tag{8.6}$$

The matrices G_1, G_2, G_3 and G_4 (also G_5 and G_6) in equations (8.5) and (8.6) do not commute. This makes it difficult to deduce directly the formula for the spectral radius of T. But the spectral norm of all the G_i (i=1...6) matrices can be easily obtained and this can lead easily to a relation defining the upper limit of the spectral radius of the iteration matrix T in terms of ρ and s.

Finally among other areas which need further research we suggest a comparative study between the EAD fully iterative method and methods such as the SOR method.

Bibliography

[1] G. BIRKHOFF, R. S. VARGA, AND D. M. YOUNG, *Alternating Direction Implicit methods, Advances in Computers*, vol. 3, Academic Press, New York, 1962.

[2] R. L. BURDEN, J. D. FAIRES, AND A. C. REYNOLDS, *Numerical Analysis*, P.W.S Publishers, Boston, Massachusettes, second ed., 1981.

[3] G. CARVER, *A spectral meteorological model on the ICL DAP*, Parallel Computing, 8 (1988), pp. 121–126.

[4] C. CANUTO, M. Y. HUSSAINI, A. QUARTTERONI, AND T. A. ZANG, *Spectral Methods in Fluid Dynamics*, Springler-Verlag, New York, 1988.

[5] C. R. CHESTER, *Techniques in Partial Differential Equations*, McGraw-Hill, Newyork, 1971.

[6] A. DEIF, *Advanced Matrix Theory for Scientists and Engineers*, Academic Press Inc., New York, 1981.

[7] J. DOUGLAS, *Alternating direction methods for three space variables*, Numerische Mathematik, 4 (1984), pp. 41 – 63.

[8] J. DOUGLAS AND H. H. RACHFORD, *On the numerical solution of heat conduction problems in two or three space variables*, Trans. Amer. Maths. Soc., 82 (1956), pp. 421 – 439.

[9] Y. D'YAKONOV, *On the application of disintegrating difference operators*, Z, Vycil. Mat. i. Mat. Fiz., 3 (1963), pp. 385–388.

[10] D. J. EVANS, *The use of preconditioning in iterative methods for solving linear equations with symmetric positive definite matrices*, J.I.M.A., 4 (1968), pp. 295–314.

[11] ——, *Comparison of the convergence rates of iterative methods for solving linear equations with preconditioning*, Greek Math Soc., Caratheodory Symp., (1973), pp. 106–135.

[12] ——, *Iterative Sparse Matrix Algorithms, in: Software for Numerical Mathematics*, Academic Press., 1974, pp. 49–83. edited by D. J. Evans.

[13] ——, *New Parallel Algorithms for Partial Differential Equations, in: Parallel Computing 83*, Elsevier Science Publishers, Amesterdam, 1984, pp. 3–56. edited by M. Feilmeier, J. Joubert and U. Schendel.

[14] ——, *Group Explicit Iterative methods for solving large linear systems*, Intern. J. Computer Math., 17 (1985), pp. 81–108.

[15] ——, *The solution of periodic parabolic equations by the Coupled Alternating Group Explicit CAGE iterative method*, Intern. J. Computer Math., 34 (1990), pp. 227–235.

[16] D. J. EVANS AND C. LI, *The Alternating Group Explicit (AGE) Iterative Method and its Parallel Implementation, in: Iterative Methods in Linear Algebra*, Elsevier Science Publishers, Amsterdam, 1992, pp. 243–250. edited by R. Beauwens and P. de Groen.

[17] D. J. EVANS, E. A. LIPITAKIS, AND N. M. MISSIRILIS, *On sparse and compact preconditioned Conjugate Gradient methods for partial differential equations*, Intern. J. Computer Math., 9 (1980), pp. 55–80.

[18] D. J. EVANS AND N. M. MISSIRILIS, *The preconditioned simultaneous displacement (PSD) method for elliptic difference equations*, M.C.S., 22 (1980), pp. 256–263.

[19] D. J. EVANS AND A. S. ROOMI, *The solution of parabolic differential equations by the age method with d'yakonov splitting*, Intern. J. Computer Math., 32 (1990), pp. 181–191.

[20] D. J. EVANS AND M. S. SAHIMI, *The Alternating Group Explicit (AGE) iterative method for solving parabolic equations I: 2-Dimensional Problems*, Intern. J. Computer Math., 24 (1988), pp. 311–341.

[21] ——, *The Alternating Group Explicit (AGE) iterative method for solving parabolic equations II: 3 Space Dimensional Problems*, Intern. J. Computer Math., 26 (1989), pp. 117–142.

[22] ——, *The Alternating Group Explicit (AGE) Iterative Method to Solve Parabolic and Hyperbolic PDEs, in: Annual Review of Numerical Fluid Mechanics, and Heat Transfer.*, vol. II, Hemispheric Pub. Co. USA., 1989, pp. 283–390. edited by C. L. Tien and T. C. Chawla.

[23] G. FAIRWEATHER AND I. M. NAVON, *A linear ADI method for the shallow water equations*, Journal of Computational Physics, 37 (1980), pp. 1–18.

[24] C. A. J. FLETCHER, *Computational Techniques for Fluid Dynamics*, Spring-Verlag, New York, 1988.

[25] C. F. GERALD AND P. O. WHEATLEY, *Applied Numerical Analysis*, Addison-Wesley, New York, fifth ed., 1994.

[26] D. GOTLEIB AND E. TURKEL, *Phase error and stability of second order methods for hyperbolic problems II*, Journal of Computational Physics, 15 (1974), pp. 251–265.

[27] B. GUSTAFSSON, *An alternating direction implicit method for solving the shallow water equations*, Journal of Computational Physics, 7 (1971), pp. 239–254.

[28] W. HACKBUSCH, *Iterative Solution of Large Sparse Systems of Equations*, Springer-Verlag, New York, 1994.

[29] L. A. HAGEMAN AND D. M. YOUNG, *Applied Iterative Methods*, Academic Press Inc., New York, 1981.

[30] G. HELLWIG, *Partial Differential Equations, An Introduction*, Blaisdell, New York, 1964.

[31] F. B. HILDEBRAND, *Introduction to Numerical Analysis*, McGraw-Hill, New Delhi, second ed., 1974.

[32] M. K. JAIN, *Numerical Solution of Partial Differential Equations*, Wiley Eastern Ltd., New Delhi, second ed., 1984.

[33] L. LAPIDUS AND G. F. PINDER, *Numerical solution of partial differential equations in science and engineering*, John Wiley and Sons, Newyork, 1982.

[34] A. R. MITCHELL AND D. F. GRIFFITHS, *The Finite Difference Method in Partial Differential Equations*, Jhon Wiley & Sons, INC., New York, 1980.

[35] J. NOYE, *Numerical Solution of Differential Equations*, North-Holland, Amsterdam, 1984.

[36] I. G. PAPAGEORGIOU, *Mesoscale Modelling of the Atmospheric Boundary Layer Including Pollution Dispersion of a Coastal Area*, PhD thesis, University of Reading, Reading, U.K., 1985.

[37] D. W. PEACEMAN, *Fundamentals of Numerical Reservoir Simulation*, Elsevier Scientific Publishing Company, Amesterdam, 1977.

[38] D. W. PEACEMAN AND H. H. RACHFORD, *The numerical solution of parabolic and elliptic partial differential equations*, J. Soc. Indust. Appl. Math., 3 (1955), pp. 28 – 41.

[39] R. D. RICHTMYER AND K. W. MORTON, *Difference Methods for Initial Value Problems*, Wiley, New York, 1967.

[40] R. SADOURNY, *The dynamics of finite difference models of the shallow water equations*, Journal of the Atmospheric Sciences, 32 (1975), pp. 680–689.

[41] M. SAHIMI, *Numerical Methods for solving Hyperbolic and Parabolic Partial Differential Equations*, PhD thesis, Loughborough University of Technology, Loughborough, U.K., 1986.

[42] N. SATOFUKA, *Group Explicit Methods for the Solution of Fluid Dynamic Equations, in: Computational Fluid Dynamics*, Elsevier Science Publishers, Amesterdam, 1988, pp. 117–134. edited by G. Davis and C. Fletcher.

[43] V. K. SAUL'YEV, *Integration of Equations of Parabolic Type by the method of Nets*, Pergamon Press, Newyork, 1964. Translated by G. J. Tee.

[44] G. W. STEWART, *Introduction to Matrix Computations*, Academic Press Inc., Newyork, 1973.

[45] G. STRANG, *Linear Algebra and Its Applications*, Academic Press Inc., Newyork, third ed., 1988.

[46] E. TURKEL, *Phase error and stability of second order methods for hyperbolic problems I*, Journal of Computational Physics, 15 (1974), pp. 227–250.

[47] E. H. TWIZELL, *Computational Methods for Partial Differential Equations*, John Wiley & Sons, New York, 1984.

[48] P. J. VAN DER HOWEN AND B. P. SOMMEIJER, *Reduction of dispersion in hyperbolic difference schemes by adapting the space discretization*, Tech. Rep. NM-R8519, Centre of Mathematics and Computer Science, Amsterdam, September 1985.

[49] R. S. VARGA, *Matrix Iterative Analysis, Prentice-Hall*, Academic Press, New York, 62. Series in Automatic Computation.

[50] D. M. YOUNG, *Iterative Solution of Large Linear Systems*, Academic Press Inc., New York, 1971. Computer Science and Applied Mathematics Series.

Appendix A

The truncation error, consistency and stability analysis of an LOD scheme

In this appendix we present the truncation error, consistency and stability analysis of the LOD scheme given in subsection 6.2.3.

The Composite formula of the scheme given by equation 6.11 is expanded as :

$$(1 - r\delta_{x_1}^2 - r\delta_{x_2}^2 + r^2\delta_{x_1}^2\delta_{x_2}^2)U_{i,j}^{n+1} = U_{i,j}^n \tag{A.1}$$

This gives the following formula:

$$\begin{aligned}(1+4r)U_{i,j}^{n+1} - r(U_{i+1,j}^{n+1} + U_{i-1,j}^{n+1} + U_{i,j+1}^{n+1} + U_{i,j-1}^{n+1}) \\ 4r^2U_{i,j}^{n+1} - 2r^2(U_{i+1,j}^{n+1} + U_{i-1,j}^{n+1} + U_{i,j+1}^{n+1} + U_{i,j-1}^{n+1}) \\ +r^2(U_{i+1,j+1}^{n+1} + U_{i-1,j+1}^{n+1} + U_{i+1,j-1}^{n+1} + U_{i-1,j-1}^{n+1}) - U_{i,j}^n = 0\end{aligned} \tag{A.2}$$

The local truncation error of the above formula is easily obtained by Taylor's expansion in terms of $U_{i,j}^{n+\frac{1}{2}}$. This is done easily using the REDUCE package. After expanding, and substituting for $r = k/h^2$, equation (A.2) can be written as:

$$\begin{aligned}(1+4r)U_{i,j}^{n+1} - r(U_{i+1,j}^{n+1} + U_{i-1,j}^{n+1} + U_{i,j+1}^{n+1} + U_{i,j-1}^{n+1}) \\ +4r^2U_{i,j}^{n+1} - 2r^2(U_{i+1,j}^{n+1} + U_{i-1,j}^{n+1} + U_{i,j+1}^{n+1} + U_{i,j-1}^{n+1})\end{aligned}$$

$$+r^2(U^{n+1}_{i+1,j+1}+U^{n+1}_{i-1,j+1}+U^{n+1}_{i+1,j-1}+U^{n+1}_{i-1,j-1})\ -U^n_{i,j}=$$
$$k(\frac{\partial\tilde{u}}{\partial t}-\frac{\partial^2\tilde{u}}{\partial x_1^2}-\frac{\partial^2\tilde{u}}{\partial x_2^2}$$
$$+\frac{\partial^4\tilde{u}}{\partial x_1^2\partial x_2^2}k-\frac{1}{2}\frac{\partial^3\tilde{u}}{\partial t\partial x_2^2}k-\frac{1}{2}\frac{\partial^3\tilde{u}}{\partial t\partial x_1^2}k$$
$$-\frac{1}{12}\frac{\partial^4\tilde{u}}{\partial x_1^4}h^2-\frac{1}{12}\frac{\partial^4\tilde{u}}{\partial x_2^4}h^2+\frac{1}{24}\frac{\partial^3\tilde{u}}{\partial t^3}k^2-\frac{1}{8}\frac{\partial^4\tilde{u}}{\partial t^2\partial x_1^2}k^2$$
$$-\frac{1}{8}\frac{\partial^4\tilde{u}}{\partial t^2\partial x_2^2}k^2+\frac{1}{2}\frac{\partial^5\tilde{u}}{\partial t\partial x_1^2\partial x_2^2}k^2+O(k^\alpha h^\beta)\ )=0 \qquad \text{(A.3)}$$

where $(\alpha+\beta)\geq 3$.

$$\Leftrightarrow(1+4r)U^{n+1}_{i,j}-r(U^{n+1}_{i+1,j}+U^{n+1}_{i-1,j}+U^{n+1}_{i,j+1}+U^{n+1}_{i,j-1})$$
$$4r^2U^{n+1}_{i,j}-2r^2(U^{n+1}_{i+1,j}+U^{n+1}_{i-1,j}+U^{n+1}_{i,j+1}+U^{n+1}_{i,j-1})$$
$$+r^2(U^{n+1}_{i+1,j+1}+U^{n+1}_{i-1,j+1}+U^{n+1}_{i+1,j-1}+U^{n+1}_{i-1,j-1})\ -U^n_{i,j}=$$
$$\frac{\partial\tilde{u}}{\partial t}-\frac{\partial^2\tilde{u}}{\partial x_1^2}-\frac{\partial^2\tilde{u}}{\partial x_2^2}+O(k,h^2) \qquad \text{(A.4)}$$

This shows that the above scheme has a local truncation error of order $O(k,h^2)$. Furthermore it can be seen from equation (A.3) that all the terms of the truncation error are products of k and/or h. Thus as k *and* $h\rightarrow 0$, the truncation error tends to zero unconditionally.

Hence the scheme is *unconditionally consistent* with the heat conduction equation 4.4.

We next consider the stability of the LOD scheme using the matrix method. The two equations of the scheme are given as:

$$(1-r\delta^2_{x_1})U^{n*}_{i,j}=U^n_{i,j} \qquad \text{(A.5)}$$

$$(1-r\delta^2_{x_2})U^{n+1}_{i,j}=U^{n*}_{i,j} \qquad \text{(A.6)}$$

which when applied to all points of the mesh produce two systems to be solved in sequence. These systems are given the following equations:

$$H\mathbf{u}^*=\mathbf{u}^n+\mathbf{b_1} \qquad \text{(A.7)}$$

$$V\mathbf{u}^{n+1}=\mathbf{u}^*+\mathbf{b_2} \qquad \text{(A.8)}$$

where $\mathbf{u}^n$, $\mathbf{u}^*$ *and* $\mathbf{u}^{n+1}$ are the solution vectors at time n, the intermediate solution vector, and the solution vector at time n+1. Vectors $\mathbf{b_1}$ *and* $\mathbf{b_2}$ are known vectors associated

with the boundary values at each stage. Also H and V have the same structure as in equations 4.25 and 4.26 in section 4.1, but with $a = -r$ and $b = 1 + 2r$.

By eliminating $\mathbf{u}^*$ from equation (A.7) using equation (A.8), we get the following equation:

$$\mathbf{u}^{n+1} = G\mathbf{u}^n + b \tag{A.9}$$

where b is a known vector, and G is the *amplification matrix* and is given as:

$$G = H^{-1}V^{-1} \tag{A.10}$$

Knowing that H and V are both SPD matrices and that they commute with each other we can write that:

$$\rho(G) = \max \lambda_{i,j}(G) = \max \lambda_{i,j}(H)\lambda_{i,j}(V) \tag{A.11}$$

Also all the eigenvalues of H and V are larger than unity and are given as:

$$\lambda_{i,j}(H) = \lambda_{i,j}(V) = (1+2r) + 2\sqrt{r^2}\cos\frac{(i\pi)}{m_1+1} \quad i = 1,\ldots m_1, \quad j = 1,\ldots m_2. \tag{A.12}$$

This means that all the eigenvalues of H^{-1} and V^{-1} are less than unity. Therefore

$$\rho(G) \leq 1 \tag{A.13}$$

and the scheme is thus *unconditionally stable.*

Appendix B

The listings of some programs

B.1 Program par_AGE-1D_odd

```
program par_AGE-1D_odd
c*************************************************************c
c          This is a program showing one way of how the AGE-1D          c
c      algorithm can executed in parallel to solve the system Au=b      c
c      where A=diag(a1,c11,a1) of order m (m  is odd). The system       c
c      arises from the approx. of the 1D heat problem, and the diagonal c
c      elements are given in terms of lamda=(the mesh ratio) and        c
c      teta=(a weighting parameter depending on the scheme used).       c
c*************************************************************c
c
        implicit real  (a-h,o-z),integer*2(i-n)
        integer*4 dim1,i,m,byte1,byte_size
        parameter (dim1=10000)
        external forksub
        integer m_fork,np,n
        dimension xu(0:dim1)
        dimension ua(0:dim1)
        dimension rhst(0:dim1)
```

```
      real ls(0:dim1)
      real ls2(0:dim1)
      real lamda
      integer time1,time2
      logical flagon
      COMMON/SHAREALL/m,rhst,xu,ls,ls2,it1,flagon,maxit,
     1  eps,byte_size,byte1,e2,a1,c12,a12,c13,c14,a14,c16
            print*,'Input: m  then  maxit'
            read(*,*)m,maxit
            print*,'Input: no. of procs. np'
            read(*,*)np
            print*,'Input: lamda & teta'
            read(*,*) lamda,teta
c
            eps=0.00000010
            a1=-1.d0
c           a2=(1.-teta)*lamda
            c11=2.d0+1.d0/(lamda*teta)
c           The following line is to obtain the 'optimum' acc. param. as
c     calculated from the formula.
             s=sqrt( (c11/2.0)**2 - a1**2)
c
            e1=c11/2.d0+s
            e2=c11/2.d0-s
            delta=e1**2-a1**2
            c12=(e1*e2-a1*a1)/delta
            a12=(e1*a1-a1*e2)/delta
            c13=e2/e1
            c14=e1/delta
            a14=-a1/delta
            c16=1/e1
C
```

```
      write(*,*)'m= ',m
C
C       Setting the initial guess vector to zero.
      do 9065 i=1,m
      xu(i)=0.0d0
 9065 continue
c
c       Choosing the RHS vector (i.e., b ) so that the solution
c        vector has all its elements=1 .
      rhst(1)=c11+a1
      do 64 i=2,m-1
      rhst(i)=c11+a1+a1
 64   continue
      rhst(m)=c11+a1
C
c     %%%  preparing for static load balancing of the    %%
c     %%%  computational tasks  among the processors.    %%%%%
C
      wj=real(m)/real(np)
      if (real(int(wj)).eq.wj) then
      byte1=int(wj)
      else
      byte1=int(wj)+1
      endif
      if(mod(byte1,2).ne.0) then
      byte_size=byte1+1
      else
      byte_size=byte1
      endif
 7100 it1=1
      n=m_set_procs(np)
c       Timing the execution of the next subroutine.
```

```
      call _clock_time(time1)
c  ** Forking the AGE-1D subroutine to be executed by np processors.  ***
      n=m_fork(forksub)
c   End of parallel execution.  Sequential execution proceeds next.
      call m_kill_procs
      call _clock_time(time2)
      if(it1.ge.maxit) write(*,*)'Max number of iterations is exceeded'
      d1=real(time2-time1)/100.0
      write(*,25) np,d1
   25 format(/,30x,' The time duration for the',/,20x,'
     1 AGE-1D  ALGORITHM executed on ',i2,
     1                    ' machines',/, 30x,' is',f10.3,' seconds')
      write(*,133)it1,(ls(i),i=1,m,m/5)
  133 format(2x,t30,'No. of iters. =',i4,/,'un= ',10(f10.7))
      stop
      end
      subroutine forksub
      implicit real  (a-h,o-z),integer*2(i-n)
      integer*4 dim1,i,m,byte1,byte_size,k1
      integer*4 start1,start2,finish1,finish2,begin,end
      integer m_get_numprocs,m_get_myid,n,m_sync
      parameter (dim1=10000)
      dimension xu(0:dim1)
      dimension ua(0:dim1)
      dimension rhst(0:dim1)
      real   ls(0:dim1)
      real   ls2(0:dim1)
      logical flagon
      COMMON/SHAREALL/m,rhst,xu,ls,ls2,it1,flagon,maxit,
     1 eps,byte_size,byte1,e2,a1,c12,a12,c13,c14,a14,c16
      n_procs=m_get_numprocs()
      n=m_get_myid()
```

```
      start1=n*byte_size+1
      finish1=(n+1)*byte_size
      if(finish1.gt.(m-2)) finish1=m-2
      start2=start1+1
      finish2=finish1+1
      begin=n*byte1+1
      end=(n+1)*byte1
      if(end.gt.m) end=m
  700 continue
      if(n.eq.(n_procs-1)) then
      flagon=.true.
      ls(1)=rhst(1)-e2*xu(1)
      endif
      do 111 i=start2,finish2,2
      k1=i+1
c
      ls(i)=rhst(i)-(e2*xu(i)+a1*xu(k1))
      ls(k1)=rhst(k1)-(e2*xu(k1)+a1*xu(i))
  111 continue
C sync pt. 1
      call m_sync()
c  Synchronize here, although parctically not necessary
      do 114 i=start1,finish1,2
      k1=i+1
      ls2(i)=rhst(i)-(c12*ls(i)+a12*ls(k1))
      ls2(k1)=rhst(k1)-(c12*ls(k1)+a12*ls(i))
  114 continue
      if(n.eq.(n_procs-1)) ls2(m)=rhst(m)-c13*ls(m)
C sync pt. 2
      call m_sync()
c
c--------------
```

```
c  FROM HERE ON, the vector ls is used again (instead of up1) to save memory.
      if(n.eq.(n_procs-1)) ls(1)=c16*ls2(1)
      do 120 i=start2,finish2,2
      k1=i+1
      ls(i)=c14*ls2(i)+a14*ls2(k1)
      ls(k1)=c14*ls2(k1)+a14*ls2(i)
  120 continue
      if(it1.ge.maxit) go to 499
C                          sync pt. 3
      call m_sync()
c  Begin testing
      do 3386 i=begin,end
      s1=abs(ls(i)-xu(i))
      if(s1.gt.eps) flagon=.false.
      if(.not.(flagon)) i=end
 3386 continue
c  End of testing
      if(flagon) go to 499
      do 3065 i=begin,end
      xu(i)=ls(i)
 3065 continue
      if(n.eq.(n_procs-1)) it1=it1+1
      call m_sync()
C sync pt. 4
      go to 700
 49   return
      end
```

B.2 Program pbl

```
$system
        program PBL
c
c   Steady  state equations of momentum for the  1-D model of the idealized
c                          PLANETARY BOUNDARY LAYER
c
c                          ***********************************
c
        implicit integer*2(i-n),real*8(a-h,o-z)
        integer dim1,m
        logical flagon
        character*10 fname,output
        parameter (dim1=102)
        dimension yu(0:dim1)
        dimension yv(0:dim1),ua(0:dim1),fzv(dim1),fzu(dim1),va(0:dim1)
        dimension xu(0:dim1),xv(0:dim1),z(0:dim1)
        dimension bpu(dim1),bpv(dim1),rsu(dim1),rsv(dim1)
        dimension up1(0:dim1),vp1(0:dim1),eru(0:dim1),erv(0:dim1)
        real km
        pi=4.0d0*datan(1.d0)
        phi=(pi/4.0)
        omega=2.0*pi/(24.0*3600.0)
        f=2.0*omega*dsin(phi)
        km=10.0
        write(*,545)
 545    format(//,1x,50('*'),/,
     1  1x,t25,' Enter the name of the Input File ',2x,/,t4,'Enter',/,
     1  ' "i1"  ; for varying the s',t35,' "i2" ; for varying eps,ug,and vg ')
        read(*,*)fname
c       fname='i3'
```

```
      open(unit=5,file=fname,form='formatted')
      if(fname.eq.'i1') then
      output='pbl_r1'
      elseif(fname.eq.'i2') then
      output='pbl_r2'
      else
      output='pbl_r'
      endif
      open(unit=6, file='output',form='formatted')
      read(5,*)nruns
      write(*,845)
  845 format(//,1x,80('*'))
      write(6,74)
   74 format(1x,90('*'),/1x,'**',t8,'Ad2(W)/dz**2+BW+C=0 ',/,1x,'**',t8,
     1 'Where  W=(u,v)     A=((Km,0),(0,Km))
     1 ;B=((0,f),(-f,0))     ;C=(-f*vg),(f*ug))'
     1 ,/,80('*'),//)
      write(6,444)
  444 format(1x,t30,'    ANALYTICAL SOLUTON IS :',/,1x,'
     1 ua(k)=ug*(1-exp(-cf*z(k))*cos(cf*z(k)))-vg*exp(-cf*z(k))*sin(cf*z(k))
     1 ',/,1x,'
     1 va(k)=vg*(1-exp(-cf*z(k))*cos(cf*z(k)))+ug*exp(-cf*z(k))*sin(cf*z(k))'
     1 ,/,1x,80('*'),/)
c
c
      do 4300 nn=1,nruns
      read(5,*)ug,vg,eps,s
      read(5,*)dz,finz
      h=dz
      m=int((finz/dz)) -1
      cf=sqrt((f/(2.0*km)))
c
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      write(6,294)finz,dz,ug,vg,eps
 294  format(1x,'finz =',f10.4,5x,'dz = ',f8.4,/,
     1 1x,t15,'ug =',f8.5,t30,'vg = ',f8.4,/,t24,'eps =',e7.2)
      write(6,297)s
 297  format(1x,t50,'Acc. Param. s =',f4.2)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  setting the z_axis
      zd=0.0
      do 3334 i=0,m+1
      z(i)=zd
      zd=zd+dz
 3334 continue
c
C                  Calculating the analytical solution
c
      do 32 k=0,m+1
      ua(k)=ug*(1-dexp(-cf*z(k))*cos(cf*z(k)))-vg*dexp(-cf*z(k))*sin(cf*z(k))
      va(k)=vg*(1-dexp(-cf*z(k))*cos(cf*z(k)))+ug*dexp(-cf*z(k))*sin(cf*z(k))
   32 continue
c
C                        Setting the initial guess
      do 3012 k=1,m
            xu(k)=0.0
            xv(k)=0.0
 3012 continue
c
c        Setting Boundary Conditions from the analytical sol.
c
      do 3002 k=0,m+1,m+1
      xu(k)=ug*(1.d0-dexp(-cf*z(k))*cos(cf*z(k)))
     1                              -vg*dexp(-cf*z(k))*sin(cf*z(k))
```

```
      xv(k)=vg*(1.d0-dexp(-cf*z(k))*cos(cf*z(k)))
     1                              +ug*dexp(-cf*z(k))*sin(cf*z(k))
 3002 continue
      xx=xu(m+1)
      yy=xv(m+1)
CCCCCccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccCCCCC
CCCCC                                                                  CCCCC
c
c   values a and b of G=1/2E=[(a,b)(-b,a)]
      a=1.d0
      b=-f*h**2/(2.d0*km)
      cu=f*vg*h**2/km
      cv=-f*ug*h**2/km
      ams=a-s
      aps=a+s
c                        The inverse of G+sI
      delg=aps**2+b**2
      agi=aps/delg
      bgi=-b/delg
c
c              On the DELTA of C=[((G+sI),-I),(-I,(G+sI))].
c
      adelc=aps**2-b**2-1.d0
      bdelc=2.d0*b*aps
      ddc=adelc**2+bdelc**2
C
C
c On the  Elements of [Delta(C)*Inv(G1+sI)] needed for up1 and vp1 at i=1 & i=m.
      sh1=agi*adelc-bgi*bdelc
      sh2=agi*bdelc+bgi*adelc
C
c                On the  Elements of the Inverse of DELTA(C).
```

```
c
      adelci=adelc/ddc
      bdelci=-bdelc/ddc
C           On the  Elements of the (square of the Inverse) of delta(C).
      e1=adelci**2-bdelci**2
      e2=2.d0*adelci*bdelci
C           On the elements of the product matrix of (G+sI)(G-sI).
      f1=a**2-s**2-b**2-1.d0
      f2=2.d0*a*b
C           -2s*INV[(DELTA)]**2
      h1=-2.d0*s*e1
      h2=-2.d0*s*e2
C
C     On the elements of the product matrix of :
C           [(Square of the Inverse) of delta(C)]*[(C+sI)(C-sI)].
C
      g1=e1*f1-e2*f2
      g2=e1*f2+e2*f1
C
c           On the  Elements of D=Inv(C+sI). needed for calculating b' .
c        *********    In addition to elements of Inv(Delta):
c
      aci=aps*adelci-b*bdelci
      bci=aps*bdelci+b*adelci
c
C                     Calculating the rhs. vector b.
      fzu(1)=-cu+xu(0)
      fzv(1)=-cv+xv(0)
      do 2001 i=2,m-1
      fzu(i)=-cu
      fzv(i)=-cv
```

```
 2001   continue
        fzu(m)=-cu+xu(m+1)
        fzv(m)=-cv+xv(m+1)

c
C                          Calculating the rhs. vector b' or (bp).
        bpu(1)=agi*fzu(1)+bgi*fzv(1)
        bpv(1)=-bgi*fzu(1)+agi*fzv(1)
        do 2002 i=2,m-2,2
        bpu(i)=aci*fzu(i)+bci*fzv(i)+adelci*fzu(i+1)+bdelci*fzv(i+1)
        bpv(i)=-bci*fzu(i)+aci*fzv(i)-bdelci*fzu(i+1)+adelci*fzv(i+1)
c
        bpu(i+1)=adelci*fzu(i)+bdelci*fzv(i)+aci*fzu(i+1)+bci*fzv(i+1)
        bpv(i+1)=-bdelci*fzu(i)+adelci*fzv(i)+aci*fzv(i+1)-bci*fzu(i+1)
 2002   continue
        bpu(m)=agi*fzu(m)+bgi*fzv(m)
        bpv(m)=-bgi*fzu(m)+agi*fzv(m)
C
C                                The AGE-1D Algorithm.
C                           y = b-(G2-sI)u(p)
C                           u(p+1) = Inv(G2+sI)[b - (G1-sI)*Inv(G+sI)y]
C                                  = b' - Inv(G2+sI)*(G1-sI)*Inv(G1+sI)*y
C
        it=1
 2000   continue
C
c  ********     calculating the vector y=b-(G_{2}-sI)w     ******
        yu(1)=fzu(1)-( ams*xu(1)+b*xv(1))
        yv(1)=fzv(1)-( ams*xv(1)-b*xu(1))
        do 2003 i=2,m-2,2
        yu(i)=fzu(i)-( ams*xu(i)+b*xv(i)-xu(i+1))
        yv(i)=fzv(i)-( ams*xv(i)-b*xu(i)-xv(i+1))
c
```

```
      yu(i+1)=fzu(i+1)-( ams*xu(i+1)+b*xv(i+1)-xu(i))
      yv(i+1)=fzv(i+1)-( ams*xv(i+1)-b*xu(i+1)-xv(i))
 2003 continue
      yu(m)=fzu(m)-( ams*xu(m)+b*xv(m))
      yv(m)=fzv(m)-( ams*xv(m)-b*xu(m))
C
c                Calculating  [Inv*(DELTA)]**2 * (G1-sI)*(G1+sI)*y
      do 2004 i=1,m-1,2
      rsu(i)=    g1*yu(i)+g2*yv(i)          +h1*yu(i+1)+h2*yv(i+1)
      rsv(i)=   -g2*yu(i)+g1*yv(i)          -h2*yu(i+1)+h1*yv(i+1)
c
      rsu(i+1)=    g1*yu(i+1)+g2*yv(i+1)          +h1*yu(i)+h2*yv(i)
      rsv(i+1)=   -g2*yu(i+1)+g1*yv(i+1)          -h2*yu(i)+h1*yv(i)
 2004 continue
C                      Calculating up1,vp1.
      up1(1)=bpu(1)-( sh1*rsu(1)+sh2*rsv(1) )
      vp1(1)=bpv(1)-( sh1*rsv(1)-sh2*rsu(1) )
      do 2005 i=2,m-2,2
      up1(i)=bpu(i)-( aps*rsu(i)+b*rsv(i)+rsu(i+1) )
      vp1(i)=bpv(i)-( aps*rsv(i)-b*rsu(i)+rsv(i+1) )
c
      up1(i+1)=bpu(i+1)-( aps*rsu(i+1)+b*rsv(i+1)+rsu(i) )
      vp1(i+1)=bpv(i+1)-( aps*rsv(i+1)-b*rsu(i+1)+rsv(i) )
 2005 continue
      up1(m)=bpu(m)-( sh1*rsu(m)+sh2*rsv(m) )
      vp1(m)=bpv(m)-( sh1*rsv(m)-sh2*rsu(m) )
C
      flagon=.true.
      do 3085 i=1,m
      if ((abs(up1(i)-xu(i)).gt.eps).or.(abs(vp1(i)-xv(i)).gt.eps)) then
      flagon=.false.
      endif
```

```
 3085    continue
         do 400 i=1,m
         xu(i)=up1(i)
         xv(i)=vp1(i)
 400     continue
c
         if(flagon) then
         do 408 i=1,m
         eru(i)=ua(i)-xu(i)
         erv(i)=va(i)-xv(i)
408      continue
         elseif(it.gt.200) then
         go to 4001
         else
         it=it+1
         go to 2000
         endif
         write(6,48)it
48       format(//,t20,' RESULTS AFTER ',i3,' iterations ')
         write(6,18)(ua(i),i=1,m,(m+1)/10)
         write(6,19)(xu(i),i=1,m,(m+1)/10)
         write(6,56)(eru(i),i=1,m,(m+1)/10)
         write(6,98)(va(i),i=1,m,(m+1)/10)
         write(6,89)(xv(i),i=1,m,(m+1)/10)
         write(6,56)(erv(i),i=1,m,(m+1)/10)
  18     format(/,1x,'Anal U = ',10('  ',f6.3))
  98     format(/,1x,'Anal V = ',10('  ',f6.3))
  19     format(1x,'Numer U =  ',10('  ',f6.3))
  89     format(1x,'Numer V =  ',10('  ',f6.3))
  56     format(1x,'error =    ',10('  ',f6.3))
         go to 4002
4001     write(6,*)'run is stopped : more than 200 inner iters. for it =',it
```

```
4002    continue
4300    continue
        end
```

B.3 Program EAD_fully_iterative

```
$system
$callwarn
$alignwarn
      program elliptic_EAD
C   This program solves the Laplace problem using
C  the EAD fully iterative method- Strategy III
C
      implicit real*8 (a-h,o-z),integer*2(i-n)
      integer dim1,dim,dim2,din
      parameter (dim1=100,dim2=100)
      integer i1,i2
      character*10 boundary
      real*8 l
      dimension ua(0:dim1,0:dim2),un(0:dim1,0:dim2)
      dimension d(0:dim1,0:dim2),rhs(1:dim1,1:dim2)
      dimension aer(0:dim1),xu(0:dim1,0:dim2)
      dimension x(0:dim1),y(0:dim2)
      logical finish
      COMMON/shareall/dim,din
C
      write(*,*)'enter dx,dy,finx,finy,s,ro'
      open(unit=5, file='vv_in',form='formatted')
      read(5,*)mnit
      read(5,*) dim,din,finx,finy,s,ro,eps
      open(unit=6, file='vv_strat3',form='formatted')
      dx=(finx)/(float(dim) + 1)
      dy=(finy)/(float(din) + 1)
      write(6,15)dx,dy,finx,finy
      write(6,*)'        dim= ',dim,'        din= ',din
      write(6,915)eps
```

```
915 format(/,t35,'Eps= ',1pe9.2)
      write(6,151)ro,s
 15   format(1x,2x,'dx=',f5.2,2x,'dy=',f5.2,2x,
   1  'finx=',f5.2,2x,'finy=',f5.2)
151   format(1x,t40,'The ADI parameter rho= ',f5.3,/,
   1           t40,'The AGE parameter s= ',f5.3)
      a1=-1.d0
      c1=4.d0
      c11=(1.d0/2.d0)*c1+ro
      e=2.d0-ro
      two_ro=2.d0*ro
c
C  setting the x_axis
      xd=0.0
      do 3334 i=0,dim+1
      x(i)=xd
      xd=xd+dx
3334  continue
C  setting the y_ayis
      yd=0.0
      do 34 j=0,din+1
      y(j)=yd
      yd=yd+dy
 34   continue
C Setting initial values for the numerical solution.
      do 33 i=1,dim
      do 33 j=1,din
      xu(i,j)=0.0d0
      un(i,j)=xu(i,j)
 33   continue
C
c          B O U N D A R Y      C O N D I T I O N S
```

```
C Setting  boundary values for the numerical solution.
c
        do 38 i=0,dim+1,dim+1
        do 38 j=1,din
        xu(i,j)=x(i)*(1.0-x(i))+y(j)*(y(j)-1.0)
   38   continue
c
        do 39 j=0,din+1,din+1
        do 39 i=1,dim
        xu(i,j)=x(i)*(1.0-x(i))+y(j)*(y(j)-1.0)
   39   continue
c
C Calculating the analytical solution
        do 32 i=1,dim
        do 32 j=1,din
        ua(i,j)=x(i)*(1.0-x(i))+y(j)*(y(j)-1.0)
   32   continue
c
C   The right hand of        Au=d        is now Calculated.
C
        d(1,1)=xu(0,1)+xu(1,0)
        do 12 i=2,dim-1
        d(i,1)=xu(i,0)
   12   continue
        d(dim,1)=xu(dim+1,1)+xu(dim,0)
        do 310 j=2,din-1
        d(1,j)=xu(0,j)
        do 2 i=2,dim-1
        d(i,j)=0.d0
    2   continue
        d(dim,j)=xu(dim+1,j)
  310   continue
```

```
      d(1,din)=xu(1,din+1)+xu(0,din)
      do 22 i=2,dim-1
      d(i,din)=xu(i,din+1)
   22 continue
      d(dim,din)=xu(dim,din+1)+xu(dim+1,din)
c
C setting
      it=0
      write(6,*)'The first ADI iterations had 3 inner AGE iters.'
   88 level_adi=1
      it=it+1
c   Calculating the R.H.S at each ADI level.
   98 call rhsadi(e,two_ro,level_adi,d,xu,rhs)
c   CALLING THE AGE ROUTINE TO EXCUTE ONE ITERATION.
      ii=0
 6666 call ageodd(xu,rhs,level_adi,a1,c11,s)
      ii=ii+1
c
      if(it.gt.1) ii=ii+2
  677 if(ii.le.2) go to 6666
      level_adi=level_adi+1
      if(level_adi.le.2) go to 98
c Test the convergence of ADI
      finish=.true.
      call test2(xu,un,eps,finish)
      do 954 i=1,dim
      do 954 j=1,din
      un(i,j)=xu(i,j)
  954 continue

c
      if(finish) go to 231
      if(it.ge.mnit) then
```

```
      write(6,*)'The ADI did NOT converge; max. No. of iterations exceeded'
      go to 231
      endif
      go to 88
231   write(6,*) 'The results after ',it,'  ADI iterations.'
      call abser(ua,un,x,y)
      stop
      end
c
c SUBROUTNES FOLLOW HERE
      subroutine rhsadi(e,two_ro,level_adi,d,xu,rhs)
      implicit real*8(a-h,o-z),integer*2(i-n)
      integer dim1,dim2,dim,din
      parameter (dim1=100,dim2=100)
      dimension d(0:dim1,0:dim2),rhs(1:dim1,1:dim2)
      dimension xu(0:dim1,0:dim2)
      COMMON/shareall/dim,din
c                    At the first ADI level
      if(level_adi.eq.2) go to 701
 700  do 223 i=1,dim
      rhs(i,1)=d(i,1)-(e*xu(i,1)-xu(i,2))
      do 224 j=2,din-1
      rhs(i,j)=d(i,j)-(-xu(i,j-1)+e*xu(i,j)-xu(i,j+1))
 224  continue
      rhs(i,din)=d(i,din)-(e*xu(i,din)-xu(i,din-1))
 223  continue
C
      go to 498
c  Calculating the RHS=(V-ro*I)xu+ 2*ro*uph  which is equivalent to:
c    RHS=d-(rhs=[d-(V-ro*I)xu]) + 2*ro*uph
c                    At the SECOND ADI level
 701  do 225 j=1,din
```

```
        do 225 i=1,dim
rhs(i,j)=d(i,j)-rhs(i,j)+two_ro*xu(i,j)
 225    continue
 498    return
        end
c
        subroutine ageodd(xu,rhs,level_adi,a1,c11,s)
        implicit real*8(a-h,o-z),integer*2(i-n)
        integer dim1,dim2,dim,din
        parameter (dim1=100,dim2=100)
        dimension d(0:dim1,0:dim2),rhs(1:dim1,1:dim2)
        dimension xu(0:dim1,0:dim2)
        real*8 ls(0:dim1,0:dim2)
        real*8 ls2(0:dim1,0:dim2)
        COMMON/shareall/dim,din
        e1=c11/2.d0+s
        e2=c11/2.d0-s
        delta=e1**2-a1**2
        c12=(e1*e2-a1*a1)/delta
        a12=(e1*a1-a1*e2)/delta
        c13=e2/e1
        c14=e1/delta
        a14=-a1/delta
        c16=1.d0/e1
c-----------------------------
c
        if(level_adi.eq.1) go to 700
        if(level_adi.eq.2) go to 701
c AGE for level-ADI    I
C
 700    continue
c  Calculating the right hand side of :
```

```
c (G1+sI)uph=rhsadi-(G2-sI)xu=ls
      do 112 j=1,din
      ls(1,j)=rhs(1,j)-e2*xu(1,j)
      do 111 i=2,dim-1,2
      ls(i,j)=rhs(i,j)-(e2*xu(i,j)+a1*xu(i+1,j))
      ls(i+1,j)=rhs(i+1,j)-(e2*xu(i+1,j)+a1*xu(i,j))
 111  continue
 112  continue
C
c  Calculating the right hand side of :
c (G2+sI)up1=rhsadi-(G1-sI)(G1+sI)^{-1}ls
      do 115 j=1,din
      do 114 i=1,dim-2,2
      ls2(i,j)=rhs(i,j)-(c12*ls(i,j)+a12*ls(i+1,j))
      ls2(i+1,j)=rhs(i+1,j)-(c12*ls(i+1,j)+a12*ls(i,j))
 114  continue
      ls2(dim,j)=rhs(dim,j)-c13*ls(dim,j)
 115  continue
c
c-----------------------------
c  FROM HERE ON, the vector ls is used again (instead of up1) to save memory.
      do 118 j=1,din
      xu(1,j)=c16*ls2(1,j)
      do 120 i=2,dim-1,2
      xu(i,j)=c14*ls2(i,j)+a14*ls2(i+1,j)
      xu(i+1,j)=c14*ls2(i+1,j)+a14*ls2(i,j)
 120  continue
 118  continue
      go to 499
c
c AGE for level-ADI II
 701  continue
```

```
c
      do 211 i=1,dim
      ls(i,1)=rhs(i,1)-e2*xu(i,1)
      do 212 j=2,din-1,2
      ls(i,j)=rhs(i,j)-(e2*xu(i,j)+a1*xu(i,j+1))
      ls(i,j+1)=rhs(i,j+1)-(e2*xu(i,j+1)+a1*xu(i,j))
 212  continue
 211  continue
C
      do 214 i=1,dim
      do 215 j=1,din-2,2
      ls2(i,j)=rhs(i,j)-(c12*ls(i,j)+a12*ls(i,j+1))
      ls2(i,j+1)=rhs(i,j+1)-(c12*ls(i,j+1)+a12*ls(i,j))
 215  continue
      ls2(i,din)=rhs(i,din)-c13*ls(i,din)
 214  continue
c
c-----------------------------
c FROM HERE ON, vector ls is used again (instead of up1) to save memory.
      do 220 i=1,dim
      xu(i,1)=c16*ls2(i,1)
      do 218 j=2,din-1,2
      xu(i,j)=c14*ls2(i,j)+a14*ls2(i,j+1)
      xu(i,j+1)=c14*ls2(i,j+1)+a14*ls2(i,j)
 218  continue
 220  continue
 499  return
      end
c
      subroutine test2(up1,uph,eps,flagon)
      implicit real*8(a-h,o-z),integer*2(i-n)
      integer dim1,dim2,dim,din
```

```
      parameter (dim1=100,dim2=100)
      dimension uph(0:dim1,0:dim2),up1(0:dim1,0:dim2)
      logical flagon
      COMMON/shareall/dim,din
      flagon=.true.
      emax=0
      do 3386 j=1,din
      do 3386 i=1,dim
      temp=abs(up1(i,j)-uph(i,j))
      if (temp.gt.emax) emax=temp
 3386 continue
      if (emax.gt.eps) flagon=.false.
      return
      end
c
      subroutine abser(ua,un,x,y)
      implicit real*8(a-h,o-z),integer*2(i-n)
      integer dim1,dim2,dim,din
      parameter (dim1=100,dim2=100)
      dimension un(0:dim1,0:dim2),ua(0:dim1,0:dim2)
      dimension aer(0:dim1),x(0:dim1),y(0:dim1)
      COMMON/shareall/dim,din
      errorsum=0.d0
      count=0.d0
      write(6,249)(x(i),i=1,dim,(dim/5))
      do 101 j=1,din,(din/5)
      do 9 i=1,dim
      count=count+1.0d0
      aer(i)=abs(ua(i,j)-un(i,j))
      errorsum=errorsum+aer(i)
 9    continue
      write(6,186)y(j)
```

```
      write(6,18)(ua(i,j),i=1,dim,(dim/5))
      write(6,19)(un(i,j),i=1,dim,(dim/5))
      write(6,56)(aer(i),i=1,dim,(dim/5))
 56   format(2x,6x,'  ','er =  ',10(1pe9.2))
101   continue
186   format(/,'x_{2}=',f5.4)
 18   format(/,7x,'Anal.  U = ',10(f9.7),/)
  19   format(7x,'Numer. U =  ',10(f9.7),/)
  249  format(7x,'   x_{1}=   ',10(f9.7),/)
c
 614    allerroraverage=errorsum/count
        write(6,965)allerroraverage
 965    format(/,t35,'Average of all absolute errors =',1pe11.2)
        return
        end
```

B.4 Program Shallow_Water_EAD

```
$system
        program water
c*************************************************************C
c A program to solve the linearized shallow water    C
c equations in 2D using the     C
c EAD method    C
c*************************************************************C
c
        implicit real*8(a-h,o-z),integer*2(i-n)
        real*8 lamda,l
        logical convergent
        integer dim1,dim2
        parameter (dim1=11,dim2=11)
        dimension un(0:dim1,0:dim2)
c
        dimension ua(0:dim1,0:dim2)
        dimension xu(0:dim1,0:dim2)
c
        dimension va(0:dim1,0:dim2)
        dimension xv(0:dim1,0:dim2)
c
        dimension phia(0:dim1,0:dim2)
        dimension xphi(0:dim1,0:dim2)
c
        dimension dx1(0:dim1,0:dim2)
c
        dimension dx2(0:dim1,0:dim2)
c
        dimension dxx(0:dim1,0:dim2)
c
```

```
      dimension x(0:dim1),y(0:dim2)
      COMMON/shareall/m,n
c
      open(unit=5,file='wat_in')
      open(unit=6,file='wat_out')
c
c 'lasttime' is the number of time steps the program is required to run.
      read(5,*)eps,lasttime,maxit
      read(5,*)dt,ds,s
      read(5,*)istart,jstart,istep,jstep
c DATA DATA DATA DATA DATA
      m=10
      n=10
      dx=ds
      dy=ds
C  l is the length of the domain.
      l=(float(m)-1.d0)*ds
C
      g=10.d0
      pi=datan(1.d0)*4.d0
      phi0=dsqrt(10.d0*80.d0)
c s=2.8
c END  OF DATA, ..... END  OF DATA
c
      ratio=dt/ds
      lamda=phi0*ratio
      print*, ' mesh ratio =dt/ds= ',ratio
      print*, ' lamda=(dt/ds)* phi0 = ',lamda
c
      e2=-dt*phi0/(4.d0*ds)
C  Coeffs. that are used only in the AGE-Subroutine.
      e1=0.5d0-s
```

```
      di=1.d0/(0.25d0+e2*e2+s*s+s)
      e3=(0.5+s)*di
      e4=-e2*di
      e5=e1*e3-e4*e2
      e6=e1*e4+e2*e3
c
c  Calculating the meshpoints' coordinates
      call axes2(x,y,dx,dy)
      t=0.0d0
c
c Calculating the initial conditions from the theoretical solution.
      call theoretical(xu,xv,xphi,t,x,y,phi0,1)
      do 3434 lt=1,lasttime
      t1=t+dt
c Updating the analytical solution.
      call theoretical(ua,va,phia,t1,x,y,phi0,1)
c At level ADI=1
      level_adi=1
c A call   : to calc. the known rhs of eq(7:31) & eq(7:32)
      call rhv1(xu,xv,xphi,level_adi,dx1,dx2,dxx,e2)
      i1=1
      call evenage(xu,xv,xphi,level_adi,dx1,dx2,eps,i1,maxit,
     1                          e1,e2,e3,e4,e5,e6,di,s,convergent)
      if(.not.(convergent)) go to 11
c At level ADI=2
      do 1007 j=1,n
      do 1007 i=1,m
      xv(i,j)=dxx(i,j)
 1007 continue
      level_adi=2
c A call   : to calc. the known rhs of eq(7:34) & eq(7:33)
      call rhv1(xu,xv,xphi,level_adi,dx1,dx2,dxx,e2)
```

```
      i2=1
      call evenage(xu,xv,xphi,level_adi,dx1,dx2,eps,i2,maxit,
     1                              e1,e2,e3,e4,e5,e6,di,s,convergent)
      if(.not.(convergent)) go to 11
c
  231 er_sumu=0.d0
      er_sumv=0.d0
      er_sumphi=0.d0
      av_eru=0.d0
      av_erv=0.d0
      av_erphi=0.d0
c
      if(lt.ne.lasttime) go to 3435
      write(6,*)' dx=dy= ',ds,'   dt=',dt
      write(6,*)' acc. param. s = ',s
      write(6,*)' mesh ratio =dt/ds= ',ratio
      write(6,*)' lamda=ratio*phi0 = ',lamda
      write(6,*)'after  ',lt,' time steps',' i.e  at lt= ',lt
      write(6,*)'convergence occured after   ',i1,'/',i2,'  AGE-1D ',
     1  '  iterations'
c   Results given next.
c
      write(6,13)(x(i),i=istart,m,istep)
      write(6,*)'    '
   13 format(5x,'x direction = ',10(' ',1pe7.1),//)
      call abser(ua,va,phia,xu,xv,xphi,y,istart,jstart,istep,jstep,
     1 er_sumu,er_sumv,er_sumphi,av_eru,av_erv,av_erphi)
      write(6,*)'av_eru=',av_eru
      write(6,*)'av_erv=',av_erv
      write(6,*)'av_erphi=',av_erphi
      go to 3435
   11 write(*,*) 'stopped'
```

```
      stop
3435  t=t+dt
3434  continue
      end
```

C Subroutines follow next.

```
      subroutine axes2(x,y,dx,dy)
      implicit real*8(a-h,o-z),integer*2(i-n)
C Setting the coordinates for the computational grid .
      integer dim1,dim2
      parameter (dim1=11,dim2=11)
      dimension x(0:dim1),y(0:dim2)
      COMMON/shareall/m,n
c
      do 101 i=0,m+1
      x(i)=i*dx
101   continue
      do 102 j=0,n+1
      y(j)=j*dy
102   continue
      return
      end

      subroutine theoretical(u,v,phi,timelevel,x,y,phi0,l)
      implicit real*8(a-h,o-z),integer*2(i-n)
      integer dim1,dim2,dim3
      parameter (dim1=11,dim2=11)
      dimension u(0:dim1,0:dim2)
      dimension v(0:dim1,0:dim2)
      dimension phi(0:dim1,0:dim2)
      dimension x(0:dim1)
```

```
      dimension y(0:dim2)
      real*8 l
      COMMON/shareall/m,n
c
      write(*,*)'I am in theor. now wait please ! '
      pi=4.d0*datan(1.d0)
      do 1 i=1,m
         do 1 j=1,n
         u(i,j)=0.250d0*
     1   dsin((-dsqrt(2.d0)*phi0*timelevel+x(i)+y(j))*2.d0*pi/l)
         v(i,j)=0.250d0*
     1   dsin((-dsqrt(2.d0)*phi0*timelevel+x(i)+y(j))*2.d0*pi/l)
         phi(i,j)=(dsqrt(2.d0)/4.d0)*
     1   dsin((-dsqrt(2.d0)*phi0*timelevel+x(i)+y(j))*2.d0*pi/l)+phi0
1        continue
         return
         end

      subroutine rhv1(xu,xv,xphi,level_adi,dx1,dx2,dxx,e2)
      implicit real*8(a-h,o-z),integer*2(i-n)
      integer dim1,dim2
      parameter (dim1=11,dim2=11)
      real*8 lamda
      dimension xu(0:dim1,0:dim2)
      dimension xv(0:dim1,0:dim2)
      dimension xphi(0:dim1,0:dim2)
      dimension ua(0:dim1,0:dim2)
      dimension va(0:dim1,0:dim2)
      dimension phia(0:dim1,0:dim2)
      dimension rhst(0:dim1,0:dim2)
      dimension x(0:dim1)
      dimension y(0:dim2)
c
```

```
      dimension dx1(0:dim1,0:dim2)
      dimension dx2(0:dim1,0:dim2)
      dimension dxx(0:dim1,0:dim2)
c
      COMMON/shareall/m,n
c
      if(level_adi.eq.2) go to 655
c  The first known vector
c        input [ xphi,xu,xv,e2]
c
c          output [dx1,dx2,dxx]
c dx1 & dx2 are then used as input for the AGE subr. to get the
c  intermediate values for xu & xphi. The dxx itself represents
c   the intermediate value of xv ... see eqs. (7.31) & (7.32).
c
c dxx represents the value of z_{2}_{i,j}
      do 10 i=1,m
      dx1(i,1)=xu(i,1)
      dx2(i,1)=xphi(i,1)+e2*(xv(i,2)-xv(i,n))
      dxx(i,1)=xv(i,1)+e2*(xphi(i,2)-xphi(i,n))
      do 20 j=2,n-1
      dx1(i,j)=xu(i,j)
      dx2(i,j)=xphi(i,j)+e2*(xv(i,j+1)-xv(i,j-1))
      dxx(i,j)=xv(i,j)+e2*(xphi(i,j+1)-xphi(i,j-1))
20    continue
      dx1(i,n)=xu(i,n)
      dx2(i,n)=xphi(i,n)+e2*(xv(i,1)-xv(i,n-1))
      dxx(i,n)=xv(i,n)+e2*(xphi(i,1)-xphi(i,n-1))
10    continue
c
      go to 6568
c  The second known vector
```

```
c       The final value of xu is calculated first eq(7:34)
c       dx1 and dx2 represent the components of the rhs of eq(7:33).
 655    do 30 i=1,m
        do 30 j=1,n
         xu(i,j)=2.d0*xu(i,j)-dx1(i,j)
         dx1(i,j)=dxx(i,j)
         dx2(i,j)=2.d0*xphi(i,j)-dx2(i,j)
 30 continue
c
c
 6568   return
        end


        subroutine evenage(xu,xv,xphi,level_adi,dx1,dx2,eps,iters,maxit,
     1                                 e1,e2,e3,e4,e5,e6,di,s,flagon)
C subroutine for the use of the AGE-1D algorithm within an outer ADI iterative
C       Procedure.
        implicit real*8(a-h,o-z),integer*2(i-n)
        integer dim1,dim2
        parameter (dim1=11,dim2=11)
        dimension xu(0:dim1,0:dim2)
        dimension xv(0:dim1,0:dim2)
        dimension xphi(0:dim1,0:dim2)
        real*8 lsu(0:dim1,0:dim2)
        real*8 lsv(0:dim1,0:dim2)
        real*8 lsh(0:dim1,0:dim2)
        real*8 ls1(0:dim1,0:dim2)
        real*8 ls2(0:dim1,0:dim2)
c
        real*8 dx1(0:dim1,0:dim2)
        real*8 dx2(0:dim1,0:dim2)
        real*8 lamda
```

```
      COMMON/shareall/m,n
      logical flagon
      if(level_adi.eq.1) go to 700
      if(level_adi.eq.2) go to 701
C  Calculating g1 of equation (7.37)
c
 700  do 100 j=1,n
      do 100 i=1,m-1,2
      ls1(i,j)=dx1(i,j)-(e1*xu(i,j) - e2*xphi(i+1,j))
      ls2(i,j)=dx2(i,j)-(e1*xphi(i,j) - e2*xu(i+1,j))
c
      ls1(i+1,j)=dx1(i+1,j)-(e1*xu(i+1,j) + e2*xphi(i,j))
      ls2(i+1,j)=dx2(i+1,j)-(e1*xphi(i+1,j) + e2*xu(i,j))
  100 continue
c
c
      do 1119 j=1,n
c
      lsu(1,j)=dx1(1,j)-( e5*ls1(1,j) + e6*ls2(m,j) )
      lsh(1,j)=dx2(1,j)-( e5*ls2(1,j) + e6*ls1(m,j) )
c
      do 111 i=2,m-2,2
      lsu(i,j)=dx1(i,j) - ( e5*ls1(i,j) - e6*ls2(i+1,j) )
      lsh(i,j)=dx2(i,j) - ( e5*ls2(i,j) - e6*ls1(i+1,j) )
c
      lsu(i+1,j)=dx1(i+1,j) - ( e5*ls1(i+1,j) + e6*ls2(i,j)  )
      lsh(i+1,j)=dx2(i+1,j) - ( e5*ls2(i+1,j) + e6*ls1(i,j)  )
c
 111   continue
c
      lsu(m,j)=dx1(m,j) - ( e5*ls1(m,j) - e6*ls2(1,j) )
      lsh(m,j)=dx2(m,j) - ( e5*ls2(m,j) - e6*ls1(1,j) )
```

```
c
 1119    continue
c
        do 500 j=1,n
        do 500 i=1,m-1,2
        ls1(i,j)=e3*lsu(i,j) - e4*lsh(i+1,j)
        ls2(i,j)=e3*lsh(i,j) - e4*lsu(i+1,j)
c
        ls1(i+1,j)=e3*lsu(i+1,j) + e4*lsh(i,j)
        ls2(i+1,j)=e3*lsh(i+1,j) + e4*lsu(i,j)
 500    continue
c
        emax1=0.d0
        emax2=0.d0
        flagon=.true.
        do 880 j=1,n
        do 880 i=1,m
        er1=dabs(ls1(i,j)-xu(i,j))
        er2=dabs(ls2(i,j)-xphi(i,j))
        if(er1.gt.emax1) emax1=er1
        if(er2.gt.emax2) emax2=er2
 880    continue
c
        if((emax1.gt.eps).or.(emax2.gt.eps)) flagon=.false.
        do 991 j=1,n
        do 991 i=1,m
        xu(i,j)=ls1(i,j)
        xphi(i,j)=ls2(i,j)
991     continue
        if(flagon) go to 499
        if(iters.gt.maxit) then
        print*, 'max no. of iterations is exceeded in AGE1'
```

```
            go to 499
            endif
            iters=iters+1
            go to 700
c
c           AGE at the second level of the ADI method.
C  Calculating g2 of equation (7.39)
c
 701        do 200 i=1,m
            do 200 j=1,n-1,2
            ls1(i,j)=dx1(i,j)-(e1*xv(i,j) - e2*xphi(i,j+1))
            ls2(i,j)=dx2(i,j)-(e1*xphi(i,j) - e2*xv(i,j+1))
c
            ls1(i,j+1)=dx1(i,j+1)-(e1*xv(i,j+1) + e2*xphi(i,j))
            ls2(i,j+1)=dx2(i,j+1)-(e1*xphi(i,j+1) + e2*xv(i,j))
  200       continue
c
c
c Using the vector lsu here instead of lsv.
c
            do 9119 i=1,m
            lsu(i,1)=dx1(i,1)-( e5*ls1(i,1) + e6*ls2(i,n) )
            lsh(i,1)=dx2(i,1)-( e5*ls2(i,1) + e6*ls1(i,n) )
c
            do 911 j=2,n-2,2
            lsu(i,j)=dx1(i,j) - ( e5*ls1(i,j) - e6*ls2(i,j+1) )
            lsh(i,j)=dx2(i,j) - ( e5*ls2(i,j) - e6*ls1(i,j+1) )
c
            lsu(i,j+1)=dx1(i,j+1) - ( e5*ls1(i,j+1) + e6*ls2(i,j)  )
            lsh(i,j+1)=dx2(i,j+1) - ( e5*ls2(i,j+1) + e6*ls1(i,j)  )
c
 911        continue
```

```
c
        lsu(i,n)=dx1(i,n) - ( e5*ls1(i,n) - e6*ls2(i,1) )
        lsh(i,n)=dx2(i,n) - ( e5*ls2(i,n) - e6*ls1(i,1) )
 9119    continue
c
c
c
        do 9500 i=1,m
        do 9500 j=1,n-1,2
        ls1(i,j)=e3*lsu(i,j) - e4*lsh(i,j+1)
        ls2(i,j)=e3*lsh(i,j) - e4*lsu(i,j+1)
c
        ls1(i,j+1)=e3*lsu(i,j+1) + e4*lsh(i,j)
        ls2(i,j+1)=e3*lsh(i,j+1) + e4*lsu(i,j)
 9500   continue
c
c
c
        emax1=0.d0
        emax2=0.d0
        flagon=.true.
        do 800 j=1,n
        do 800 i=1,n
        er1=dabs(ls1(i,j)-xv(i,j))
        er2=dabs(ls2(i,j)-xphi(i,j))
        if(er1.gt.emax1) emax1=er1
        if(er2.gt.emax2) emax2=er2
800     continue
c
        if((emax1.gt.eps).or.(emax2.gt.eps)) flagon=.false.
        do 901 j=1,n
        do 901 i=1,m
        xv(i,j)=ls1(i,j)
        xphi(i,j)=ls2(i,j)
```

```
901     continue
        if(flagon) go to 499
        if(iters.gt.maxit) then
        print*, 'max no. of iterations is exceeded in AGE2'
        go to 499
        endif
        iters=iters+1
        go to 701
  499   return
        end



subroutine abser(ua,va,phia,xu,xv,xphi,y,istart,jstart,istep,jstep,
     1  er_sumu,er_sumv,er_sumphi,
     1  av_eru,av_erv,av_erphi)
c***********
C********** Subroutine for calculating the errors and printing the
C**********   results of the S. W. program.
c***********
        implicit real*8(a-h,o-z),integer*2(i-n)
        integer dim1,dim2
        parameter (dim1=11,dim2=11)
        dimension xu(0:dim1,0:dim2),xv(0:dim1,0:dim2),xphi(0:dim1,0:dim2)
        dimension ua(0:dim1,0:dim2),va(0:dim1,0:dim2),phia(0:dim1,0:dim2)
        dimension aeru(0:dim1),aerv(0:dim1),aerphi(0:dim1)
c
        dimension y(0:dim2)
        COMMON/shareall/m,n
c
         do 101 j=jstart,n,jstep
c
              do 11 i=istart,m,istep
```

```
c
            aeru(i)=dabs(ua(i,j)-xu(i,j))
            aerv(i)=dabs(va(i,j)-xv(i,j))
            aerphi(i)=dabs(phia(i,j)-xphi(i,j))
            er_sumu=er_sumu+aeru(i)
            er_sumv=er_sumv+aerv(i)
            er_sumphi=er_sumphi+aerphi(i)
      11    continue
c
            write(6,18)(ua(i,j),i=istart,m,istep)
            write(6,19)(xu(i,j),i=istart,m,istep)
            write(6,56)y(j),(aeru(i),i=istart,m,istep)
c
            write(6,1801)(va(i,j),i=istart,m,istep)
            write(6,1901)(xv(i,j),i=istart,m,istep)
            write(6,5601)y(j),(aerv(i),i=istart,m,istep)
c
            write(6,1802)(phia(i,j),i=istart,m,istep)
            write(6,1902)(xphi(i,j),i=istart,m,istep)
            write(6,5602)y(j),(aerphi(i),i=istart,m,istep)
     101    continue
c
            ipoints=(m-istart)/(istep)+1
            jpoints=(m-istart)/(istep)+1
            fprod=float(ipoints*jpoints)
            print*, 'ipoints = ',ipoints,' jpoints =  ', jpoints
            av_eru=er_sumu/fprod
            av_erv=er_sumv/fprod
            av_erphi=er_sumphi/fprod
     56     format(1x,'y= ',f8.1,' ','er_u =  ',10(' ',1pe8.1))
     18     format(/,5x,'Analytical U =',10(' ',f7.5))
     19     format(5x,'Numerical U = ',10(' ',f7.5),/)
```

```
5601   format(1x,'y= ',f8.1,' ','er_v = ',10(' ',1pe8.1),/)
1801   format(/,5x,'Analytical V =',10(' ',f7.5))
1901   format(5x,'Numerical V = ',10(' ',f7.5),//)
5602   format(1x,'y= ',f8.1,' ','er_phi = ',10(' ',1pe8.1))
1802   format(/,5x,'Analytical Phi =',10(' ',f7.3),/)
1902   format(5x,'Numerical Phi = ',10(' ',f7.3),//)
       print*, 'av_eru=',av_eru
       print*, 'av_erv=',av_erv
       print*, 'av_erphi=',av_erphi
       return
       end
```