THE AGE ITERATIVE METHODS FOR SOLVING LARGE LINEAR SYSTEMS OCCURRING IN DIFFERENTIAL EQUATIONS

BY

AB RAHMAN BIN AHMAD, B.Sc., M.Sc.

A Doctoral Thesis

Submitted in partial fulfillment of the requirements

for the award of Doctor of Philosophy

of Loughborough University of Technology

May, 1993.

IN THE NAME OF ALLAH

MOST BENEFICENT, MOST MERCIFUL

DECLARATION

This is to certify that I am responsible for the work submitted in this thesis, that the original work is my own except as specified in acknowledgements or in footnotes, and that neither the thesis nor the original work contained therein has been submitted to this or any other institution for a higher degree.

Ab Rahman bin Ahmad

DEDICATION

To

my wife,

Zaiton,

and my children,

Ab Al-Hafidz

Ab Al-Hadi

Ab Al-Hakam

Ab Al-Hakim

Husna

Huda

for their never ending patience and moral support.

ACKNOWLEDGEMENTS

First and foremost, I wish to give all the praise to Allah Almighty for with His mercy He has given me, the life and sustenance in this world, and giving me the strength and time to complete this research.

I would like to express my profound gratitude to my Director of Research, Prof. D.J. Evans and my supervisor, Dr. A. Benson, for giving me the opportunity in the first place; and subsequently, for their guidance and direction. They have directed my research since I came to Loughborough, and have had a great impact on what I have accomplished in these years and what I know today. Their thoroughness and sincerity in doing this, certainly will be a model for me in the years to come.

I am also very grateful to the staff of the Department of Computer Studies, especially to Dr. W. Yousif, the Programmer, for the useful discussion and assistance, and to Mr. G. S. Samra, the System Manager, for his invaluable help. Also, to my colleagues who at one time or the other giving moral support during the course of preparation of this thesis.

I wish to extend my gratitude to Jabatan Perkhidmatan Awam of the Government of Malaysia and Universiti Teknologi Malaysia for granting me the leave of absence and the financial support to complete this work.

Last but by no means least, I owe a greatest debt of love and gratitude to my father and mother for their seemingly unending prayer, perseverance and moral support during my stay in Loughborough.

The AGE Iterative Methods for Solving Large Linear Systems Occurring in Differential Equations

Abstract

The work presented in this thesis is wholly concerned with the Alternating Group Explicit (AGE) iterative methods for solving large linear systems occurring in solving Ordinary and Partial differential equations (ODEs and PDEs) using finite difference approximations.

The thesis spreads over four parts. Part I contains an introduction where the general terms and classification of ODEs and PDEs are explained. This is followed by a discussion of the basic mathematical concepts which are involved in developing the methods.

Part II is concerned with the numerical solution of ODEs. This part commences with the discussion of the existing well known iterative methods and a more recent method, i.e., the Alternating Group Explicit (AGE) to solve two-point boundary value problems. The AGE method will be discussed in great detail. This is followed by the application of the method for different boundaries, i.e., with a combination of Dirichlet, Neumann and periodic conditions and to show that it is comparable with other existing methods.

It can be shown that the accuracy of the AGE method can be improved by using the Numerov formula and this is presented next. Then, this is followed by the determination of optimal AGE acceleration parameters from the well known schemes such as Peaceman-Rachford (PR), Douglas, Douglas-Rachford (DR) and the scheme proposed by Guittet.

It can be shown further that, these schemes can take less computational time if they are written using different strategies, i.e., in the approaches used in the methods Coupled AGE (CAGE) and Smart AGE (SMAGE).

The multi-parameter strategy for the AGE method is presented next. It opens with the discussion of the existing multi-parameter studies such as PR, Wachspress, Young and Jordan. These multi-parameter strategies are then applied to the existing schemes and also to the second order methods of Chebyshev and Richardson.

Part III is mainly devoted to the solution of elliptic PDEs. This part commences with the determination of the optimal parameters of the AGE method by using the Douglas, DR and Guittet schemes. The region concerned is for a unit square for two dimensional (2D) problems involving the Laplace and Poisson equations and for the three dimensional (3D) problem in a unit cube. The Helmholtz equation is also considered. All problems are governed by Dirichlet boundary conditions.

Part III is continued with the solution for the 2D problem governed by Periodic and Neumann boundary conditions. This is, then followed by the application of a computational molecular form which was presented in Part I, for the PDEs. The discussion is only concerned with the 2D problem.

The nonstationary case, i.e., the application for more than one parameter is discussed next and involves three sections. Firstly, the discussion is devoted to the standard AGE for the 2D problem for, triangle and a hole in an unit square region but the results include the multi-parameter case for the 2D problem in a square region and the 3D problem in a unit cube. This is followed the section for Richardson's, the Chebyshev method and the section for the Explicit Alternating Direction (EAD) method for the unit square and cubic regions only.

Part IV concludes the thesis with the conclusions together with some suggestions for further work.

CONTENTS

PART II

THE NUMERICAL SOLUTION OF ORDINARY DIFFERENTIAL EQUATIONS

CHAPTER 3. THE SOLUTION OF THE TWO POINT BOUNDARY VALUE PROBLEM.

CHAPTER 4. THE OPTIMAL SINGLE AGE PARAMETER FOR ODE's.

PART IV

CONCLUSIONS

THE LISTS OF THE ALGORITHMS

PART I

INTRODUCTION

CHAPTER 1
INTRODUCTION

1.1 Introduction

A great many physical phenomena in science and engineering often involve the rate of change of unknown quantities called dependent variables such as temperature, pressure, ..., etc, with respect to one or more independent variables usually representing length or angles. These problems can usually be formulated in terms of differential equations.

A differential equation is defined as an equation involving a relation between the values of an unknown function and one or more of its derivatives. The mathematical formulation involving rates of change with only one independent variable is called an ordinary differential equation (ODE) or a set of such equations. Another class of differential equations which govern physical systems are partial differential equations (PDE) in which two or more independent variables are present in the differential equations.

Since the use of automatic digital computers is becoming extensive, it is becoming apparent that numerical methods are found to be the best alternative to the analytical solution in solving these problems. Moreover, the analytical solutions for these equations are extremely difficult to obtain.

Many approaches have been developed over the years for the treatment of both the ODE's and PDE's. There are two well known and widely used methods called finite difference (FD) and finite elements (FE). The finite difference methods, which are the main interest of this work, are applicable to both linear and nonlinear problems.

A full description of the finite difference methods will be given in Chapter 2.

1.2 The classification of differential equations

As mentioned earlier, the differential equations can be classified as an ODE or a PDE. The equations can be further classified as follows:

first order	*higher order*
single equation	*system of equations*
linear	*nonlinear*
homogenous	*inhomogenous*
initial value	*boundary value.*

Let us consider the two equations, i.e.,

$$A\frac{d^2U}{dx^2} + B\frac{dU}{dx} + CU + D = 0, \tag{1.2-1}$$

$$P\frac{\partial^2U}{\partial x^2} + Q\frac{\partial^2U}{\partial x\partial y} + R\frac{\partial^2U}{\partial y^2} + S\frac{\partial U}{\partial x} + T\frac{\partial U}{\partial y} + VU + W = 0. \tag{1.2-2}$$

Equation (1.2-1) is called the general second-order ordinary differential equation since its higher derivatives is of order two and U only depends on x. Equation (1.2-2) represents the general second-order partial differential equation. Its higher derivatives is also of order two and U depends on both x and y. Nearly all equations of higher order can be reduced to a system of first order equations. This system is solved by using the same methods as those used for a single equation.

Equations (1.2-1) and (1.2-2) are said to be linear if the coefficients A, ..., D, and P, ..., W are constants or function of one or both independent variable of x or x and y respectively. If those coefficients are functions of the dependent variable U, or its derivatives then the equations are nonlinear.

The equations are semi-linear applies only to when those coefficients are functions of the independent variables of x or x and y only. On the other hand, if the coefficients P, Q and R are functions of x, y , U, $\partial U/\partial x$ and $\partial U/\partial y$ but not of second derivatives then equation (1.2-2) is said to be quasilinear.

The term homogenous applies when D and W in equations (1.2-1) and (1.2-2) respectively is equal to zero, otherwise they are all inhomogenous.

Equation (1.2-2) can be classified further into three particular types, depending on the discriminant ($B^2 - 4AC$) being either positive, negative or zero. We say that equation (1.2-2) is of

Elliptic	type when	$B^2 - 4AC < 0$
Parabolic	type when	$B^2 - 4AC = 0.$
Hyperbolic	type when	$B^2 - 4AC > 0$

For example, the Laplace equation in two variables,

$$\nabla^2 U = \frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = 0, \tag{1.2-3}$$

and the Poisson equation

$$\nabla^2 U = \frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = f(x,y) \tag{1.2-4}$$

are of elliptic type, where ∇^2 is the two dimensional Laplace differential operator, i.e., $\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)$.

The diffusion or heat conduction equation

$$\frac{\partial^2 U}{\partial x^2} = \frac{1}{a^2}\frac{\partial U}{\partial t} \tag{1.2-5}$$

is parabolic where a^2 is a physical constant.

The hyperbolic type can be represented by the wave equation

$$\frac{\partial^2 U}{\partial x^2} = \frac{1}{c^2}\frac{\partial^2 U}{\partial t^2} \tag{1.2-6}$$

where c is the propagation velocity.

Equations (1.2-3) and (1.2-4) are generally associated with a steady-state or equilibrium problem. For example, the Laplace equation may describe the velocity potential for the steady flow of an incompressible non-viscous fluid and is the mathematical expression for the physical law that the rate at which such fluid enters a given region is equal to the rate at which it leaves it. In contrast, the electric potential, associated with a two dimensional electron distribution of charge density, may satisfy the Poisson equation, stating that the total electric flux through any close surface is equal to the total charge enclosed. On the other hand, in general, parabolic and hyperbolic PDE are the result from diffusion, equalization or oscillatory processes and the usual independent variables are time and space.

It is worth noting here that the form of the general (analytical) solution of a second order PDE depends very much on whether the equation is elliptic, parabolic or hyperbolic. The analysis is simplified if all the coefficients are constants.

The ODE and PDE are normally associated with the initial and/or boundary value problem. The solutions of these equations will depend on these given values. In the theory of ODEs, for example, the simple first order equation

$$\frac{dU}{dx} = U, \quad 0 < x < 1 \tag{1.2-7}$$

may have the general solution $U(x) = Ce^x$, where C is an arbitrary constant. If the initial condition $U(0) = 1$ is prescribed, then the exact solution can be written as $U(x) = e^x$. Similarly, some additional initial conditions must be supplied along with (1.2-1) and (1.2-2) in order to determine a specific solution. Thus, in general for the ODE, if $U(x_0)$, $dU(x_0)/dx$, ..., $dU^{(n-1)}(x_0)/dx$ are prescribed at one point $x = x_0$, we have an initial value problem.

For the PDE, since there are two independent variables x and y, then the added condition, which are actually the initial and/or boundary conditions, will have to be given along some curves, in the x-y plane. Depending on the type of equation that we are dealing with, in some cases the curve along which the information is given will be closed, i.e., when solving the elliptic problem. Here, the analytical solution $U(x,y)$ must satisfy the given PDE at every point of the area of integration which is bounded by the closed curve, and the given boundary condition at every point of the curve.

On the other hand, parabolic and hyperbolic equations may give rise to an open area of integration, since the initial and boundary conditions are known and normally located along the x-axis and along the parallel lines perpendicular to the x-axis respectively. Again, the solution $U(x,y)$ must satisfy the given initial and boundary conditions and satisfy the PDE at every point p of the infinite area of integration bounded by the open curve on the x-y plane.

In the ODE, if the initial conditions are prescribed at more than one point, then we have a boundary value problem. A simple example of a second order two point boundary value problem is,

$$\frac{d^2U}{dx^2} = 1, \qquad 0 < x < 1 \tag{1.2-8}$$

with the boundary conditions,

$$U(0) = 0, \; U(1) = 1.$$

The general solution is $U(x) = \frac{1}{2}x^2 + Bx + C$, where B and C are arbitrary constants. With these given conditions, we would obtain the specific solution, i.e, $U(x) = \frac{1}{2}x(x + 1)$.

This thesis will be concerned with the solution of the two-point boundary values problems in one dimension and the elliptic problems in two and three dimensions.

1.3 Types of boundary conditions

As mentioned earlier, the solution of both ODEs and PDEs must satisfy the given initial and/or boundary conditions. There are four main types of boundary conditions which arise frequently in the description of physical phenomena, namely:

1. *The First Boundary Value Problem.*

It is also called the *Dirichlet problem*, where the solution u has to satisfy the given values

$$U|_{\tau} = \phi, \text{ where } \phi = f(x,y) \tag{1.3-1}$$

on the boundary τ. If $\phi = 0$, we have the homogenous Dirichlet problem.

2. *The Second Boundary Value Problem.*

It is also known as the *Neumann problem*, where the solution U must satisfy the normal derivatives

$$\frac{\partial U}{\partial v}|_{\tau} = \psi, \text{ where } \psi = f(x,y) \tag{1.3-2}$$

on the boundary τ of the region.

3. *The Third Boundary Value Problem.*

This problem is sometimes called *Mixed* or *Robin's problem*, where the solution U must satisfy a combination of U and its derivatives

$$\left[\frac{\partial U}{\partial v} + h\right]_{\tau} = \psi, \quad \text{where } v = f(x,y) \text{ and } h = f(x) \tag{1.3-3}$$

on the boundary τ.

4. *The Fourth Boundary Value Problem.*

It is also referred to the *Periodic boundary problem*. Here, we seek the solution such that it has to satisfy the periodicity conditions, for example,

$$U|_x = U|_{x+l}, \qquad \frac{\partial U}{\partial v}|_x = \frac{\partial U}{\partial v}|_{x+l} \tag{1.3-4}$$

where l is called the period.

1.4 Properly posed and well-conditioned problems

In practical applications, since a particular solution for an ODE or a PDE has to satisfy the initial and/or boundary conditions, thus the appropriate numerical methods for solving these equations depends upon the nature of those conditions.

We say that these differential equations together with some given initial and/or boundary conditions is *properly* or *well posed* if the given boundary conditions are specified in such a way that there *exists one and only one solution* to the problem, furthermore this solution depends *continuously* on the given data.

A problem that is not properly posed is said to be *ill-posed* or nonproperly posed and *may not have* a solution.

The differential equation is said to be *well-conditioned* if every *small error* (*perturbation*) in the data of the properly posed problems results in a relatively *small change* in the solution, otherwise we say they are *ill-conditioned.*

We could illustrate the properly posed problem by considering the Laplace equation in two-dimension,

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = 0, \tag{1.4-1}$$

in the semi-strip $y > 0$, $-\frac{\pi}{2} \le x \le \frac{\pi}{2}$ under the conditions

$$U(-\frac{\pi}{2}, y) = U(\frac{\pi}{2}, y) = 0 \tag{1.4-2}$$

$$U(x,0) = 0 \tag{1.4-3}$$

$$\frac{\partial U}{\partial y}(x,0) = \phi(X) \tag{1.4-4}$$

with

$$\phi(-\frac{\pi}{2}) = \phi(\frac{\pi}{2}). \tag{1.4-5}$$

If we set $\phi(x) = 0$, then the only solution to this equation is

$$U(x,y) = 0.$$

On the other hand, if we take

$$\phi(x) = e^{-\sqrt{2n+1}} cos\ (2n+1)x \tag{1.4-6}$$

then the particular solution will be

$$U(x,y) = \frac{1}{2n+1}\ e^{-\sqrt{2n+1}} cos\ (2n+1)x\ sinh\ (2n+1)y. \tag{1.4-7}$$

It is easy to verify that the function ϕ and its derivatives for sufficiently large n differ by an arbitrary small amount from 0. Clearly, for any nonzero y, the function U has the form of a cosine function of arbitrarily large amplitude provided that n is large. Consequently, for sufficiently large n, this function differs by an arbitrarily large amount from the zero solution.

CHAPTER 2
BASIC MATHEMATICAL PRELIMINARIES

2.1 Basic matrix algebra

The solution of ordinary and partial differential equations by numerical approaches such as the finite difference method yields a system of linear, simultaneous equations which can be denoted in the matrix form $A\mathbf{u} = \mathbf{b}$. The iterative methods for solving such systems depend on some matrix properties of A, for example, irreducibility, diagonal dominance and positive definiteness of the coefficient matrix of the system. These and some other properties together with some basic facts about matrix theory will now be discussed in this chapter.

A matrix is defined as a two-dimensional array with each element denoted as $a_{i,j}$ where i specifies the row and j specifies the column of the array in which the element appears. A matrix A, for example, with N rows and M columns is said to be of size $(N \times M)$. We can denote this matrix as

$$A = [a_{i,j}] = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,M} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,M} \\ \cdots & \cdots & \cdots & \cdots \\ a_{N,1} & a_{N,2} & \cdots & a_{N,M} \end{bmatrix}. \qquad (2.1\text{-}1)$$

For $N = 1$, we would have a row matrix or row vector and for $M = 1$, we would have a column matrix or column vector. The vectors are usually denoted either by a small underlined or bold letter such as $\underline{a}$ or $\mathbf{a}$ which represent a vector. In this thesis, all vectors are written in bold. An element a_i represents the ith element of the vector $\mathbf{a}$. The term 'vector' without qualification will refer to a column vector.

Thus, a vector b, for example, whose elements are b_1, b_2, ..., b_N is said to be of order N is denoted by

$$\mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix}. \qquad (2.1\text{-}2)$$

A matrix A is said to be a square matrix of order N if $N = M$. In this thesis, the term matrices are used to imply a square matrix, unless otherwise stated. In addition, we shall use italic capital letters for all matrices.

The set of elements $a_{i,i}$, $i = 1, 2, \ldots, N$ of matrix A is a principal (main) diagonal of A. The transpose of a matrix $A = [a_{i,j}]$ is denoted as A^T and is obtained by interchanging the rows and columns of A, i.e., the element $a_{i,j}$ of A becomes $a_{j,i}$ of A^T. Thus a transpose of A in (2.1-1) becomes

$$A^T = \begin{bmatrix} a_{1,1} & a_{2,1} & \cdots & a_{N,1} \\ a_{1,2} & a_{2,2} & \cdots & a_{N,2} \\ \cdots & \cdots & \cdots & \cdots \\ a_{1,M} & a_{2,M} & \cdots & a_{N,M} \end{bmatrix}. \qquad (2.1\text{-}3)$$

The determinant of a square matrix A can be denoted either as $det(A)$ or $|A|$. An inverse of a given square matrix A, denoted by A^{-1}, if it exists, is also a square matrix such that

$$AA^{-1} = A^{-1}A = I \qquad (2.1\text{-}4)$$

where I is the identity (unit) matrix having the same order as A and is defined as follows:

$a_{1.1} = 1$, for all $i = 1, 2, \ldots, N$

$a_{i,j} = 0$, for all $i, j = 1, 2, \ldots, N$ and $i \neq j$.

If an inverse of A does exist, then A is non-singular, otherwise it is singular. On the other hand, A is non-singular if $det(A) \neq 0$ and singular if $det(A) = 0$.

If x and y are real numbers, then the conjugate of the complex number $a = x + iy$ is $a = x - iy$.

If the elements of a matrix A are complex numbers, the conjugate of A is the matrix $\bar{A}$ whose elements are conjugates of the corresponding elements of A, i.e., if $A = [a_{i,j}]$ then $\bar{A} = [\bar{a}_{i,j}]$.

The Hermitian transpose or conjugate transpose of A, denoted by A^H, is the transpose of $\bar{A}$ and also the conjugate of A^T, i.e.,

$$A^H = (\bar{A})^T = \bar{A}^T = [\bar{a}_{j,i}]. \tag{2.1-5}$$

The sum of the diagonal elements of a matrix A is called the trace of A, denoted by $trace(A)$, i.e.,

$$trace(A) = \sum_{i=1}^{N} a_{i,i}. \tag{2.1-6}$$

A permutation matrix $P = [p_{i,j}]$ is a matrix which has the entries of ones and zeroes with exactly only one non-zero entry in each row and each column. Thus, for example, consider the matrix P as

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{2.1-7}$$

is a permutation matrix of order 4. For any permutation matrix P we have,

$$PP^T = P^TP = I. \tag{2.1-8}$$

Hence $P^T = P^{-1}$.

For any two vectors **a** and **b**, both of the same order, say N, we define the inner product of **a** and **b** by

$$(\mathbf{a},\mathbf{b}) = \mathbf{a}^H\mathbf{b} = \sum_{i=1}^{N} \bar{a}_i b_i . \qquad (2.1\text{-}9)$$

Further, for any matrix A,

$$(\mathbf{a}, A\mathbf{b}) = (A^H\mathbf{a}, \mathbf{b}). \qquad (2.1\text{-}10)$$

Given a matrix $A = [a_{i,j}]$, the integers i and j are associated with rows and column with respect to A if $a_{i,j} \neq 0$ or $a_{j,i} \neq 0$.

We say, the matrix $A = [a_{i,j}]$ of order N is,

a). Symmetric, if $A = A^T$.

b). Orthogonal, if $A^T = A^{-1}$.

c). Hermitian, if $A^H = A$.

d). Null, usually denoted by 0 if $a_{i,j} = 0$, for $i, j = 1, 2, \ldots, N$.

e). Diagonal, if $a_{i,j} = 0$ for $i \neq j$, i.e., $|i-j| > 0$, where $|i-j|$ represents the modulus of any number $(i-j)$.

$$A = \begin{bmatrix} X & & & 0 \\ & X & & \\ & & \ddots & \\ & & & X \\ 0 & & & \end{bmatrix}$$

X denotes a possible non-zero element

Figure 2.1-1: Diagonal matrix

f). Banded, if $a_{i,j} = 0$, for $|i-j| > r$, where $2r+1$ is the bandwidth of A.

g). Tridiagonal, if $r = 1$, see figure 2.1-2.

$$A = \begin{bmatrix} X & X & & & \\ X & X & X & & 0 \\ & \ddots & \ddots & \ddots & \\ & & X & X & X \\ 0 & & & X & X \end{bmatrix}$$

X denotes a possible non-zero element

Figure 2.1-2: Tridiagonal matrix

h). Quindiagonal, if $r = 2$, see figure 2.1-3.

$$A = \begin{bmatrix} X & X & X & & & & \\ X & X & X & X & & 0 & \\ X & X & X & X & X & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & X & X & X & X & X \\ & 0 & & X & X & X & X \\ & & & & X & X & X \end{bmatrix}$$

X denotes a possible non-zero element

Figure 2.1-3: Quindiagonal matrix

i). Lower triangular, (strictly lower triangular), if $a_{i,j} = 0$, for $i < j$, $(i \leq j)$.

j). Upper triangular, (strictly upper triangular), if $a_{i,j} = 0$, for $i > j$, $(i \geq j)$.

k). Sparse, if a relatively large number of its elements $a_{i,j}$ are zero.

l). Dense, if a relatively large number of its elements $a_{i,j}$ are non-zero.

m). Block Diagonal, if each D_i, $i = 1, 2, \ldots, s$, (see figure 2.1-4) is a square matrix, but not necessarily of the same order.

$$A = \begin{bmatrix} D_1 & & & \\ & D_2 & & 0 \\ & & \ddots & \\ 0 & & & D_{s-1} \\ & & & & D_s \end{bmatrix}$$

Figure 2.1-4: Block Diagonal matrix

n). Block Tridiagonal, if each D_i, $i = 1, 2, \ldots, s$ is a square matrix, but not necessarily of the same order, while the E's and F's are rectangular matrices, as in figure 2.1-5.

$$A = \begin{bmatrix} D_1 & F_1 & & & \\ E_2 & D_2 & F_2 & & 0 \\ & E_3 & D_3 & F_3 & \\ & & \ddots & \ddots & \ddots \\ 0 & & E_{s-1} & D_{s-1} & F_{s-1} \\ & & & E_s & D_s \end{bmatrix}$$

Figure 2.1-5: Block Tridiagonal matrix

Young [1971], referred to this matrix as a *T*-matrix.

2.2 Diagonal dominance and irreducibility

First, we define diagonal dominance of a matrix *A*.

Definition 2.2-1

A matrix of order *N* is diagonally dominant if

$$|a_{i,i}| \geq \sum_{\substack{j=1 \\ j \neq i}}^{N} |a_{i,j}|, \text{ for all } 1 \leq i \leq N \qquad (2.2\text{-}1)$$

and for at least one *i*

$$|a_{i,i}| > \sum_{\substack{j=1 \\ j \neq i}}^{N} |a_{i,j}|. \qquad (2.2\text{-}2)$$

If (2.2-2) holds for all *i*, we say *A* has strong diagonal dominance.

The irreducibility of a matrix *A* as defined by Young [1971] is as follows:

Definition 2.2-2

A matrix of order *N* is said to be irreducible if $N = 1$ or if $N > 1$ and given any two non-empty disjoint subset *S* and *T* of *W*, the set of the first *N* positive integers, such that $S + T = W$, there exists $i \in S$ and $j \in T$ such that $a_{i,j} \neq 0$. Varga [1962] stated that a matrix of order 1 is irreducible if its single elements is non-zero, otherwise reducible.

If a matrix is irreducible, then we cannot reduce a subsystem of the form

$$\mathbf{A}\mathbf{u} = \mathbf{b} \tag{2.2-3}$$

i.e., the system of linear equations, which preserves the correspondence between the equations and the unknowns, and which can be determined independently of the larger system.

We state the following theorem of alternative definitions of irreducibility as given by Young [1971].

Theorem 2.2-1

A is irreducible if and only if there does not exist a permutation matrix P such that $P^{-1}AP$ has the form

$$P^{-1}AP = \begin{bmatrix} F & 0 \\ G & H \end{bmatrix} \tag{2.2-4}$$

where F and H are square matrices and where all elements of 0 vanish.

Theorem 2.2-2

A matrix of order N is irreducible if and only if $N = 1$ or, given any two distinct integers i and j with $i, j = 1, 2, \ldots, N$ then $a_{i,j} \neq 0$ or there exist $i_1, i_2, \ldots, i_s$ such that

$$a_{i,i_1} a_{i_1,i_2} \cdots a_{i_2,j} \neq 0. \tag{2.2-5}$$

The concept of irreducibility can be shown graphically. Given a matrix A we construct a directed graph of A as follows. Label any distinct N points in the planes as 1, 2, ..., N. For each i and j such that $a_{i,j} \neq 0$, draw an arrow form i to j. If $a_{i,j} \neq 0$ and $a_{j,i} \neq 0$, there will be an arrow from i to j and vice-versa. if $a_{i,i} \neq 0$, we can also draw a small loop which contains the points i, but this does not affect the irreducibility. The matrix is irreducible if and only if either $N = 1$ or else the graph is connected as follows.

Given any two distinct points i and j, then either there is an arrow from i to j or else there is a path of arrows from i to i_1, i_1 to i_2, ..., i_s to j.

For example, let us consider the matrix P given below.

$$P = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}. \tag{2.2-6}$$

The directed graph of (2.2-6) is given in figure 2.2-1. We can verify that the graph is connected.

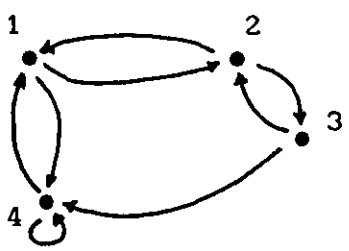


Figure 2.2-1

The matrix

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.2-7}$$

in which its directed graph is given in figure 2.2-2 is not irreducible.

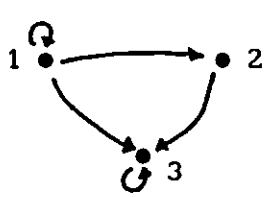


Figure 2.2-2

Its directed graph is not connected since there is no path from point 3 to point 1. Also, we cannot reach the point 1 or point 2 starting from point 2 or point 3 respectively.

We now state some fundamental theorems.

Theorem 2.2-3

If A is an irreducible matrix with diagonal dominance, then $det(A) \neq 0$ and none of the diagonal elements of A are zero.

Corollary 2.2-4

From theorem 2.2-3, if A is strongly diagonally dominant, then $det(A) \neq 0$.

We now have a sufficient condition for an Hermitian matrix to be positive definite.

Theorem 2.2-5

If A is an Hermitian matrix with non-negative diagonal elements and with diagonal dominance, then A is non-negative definite. If A is also irreducible or non-singular, then A is positive definite. We will discuss in more detail positive definite matrices in Section 2.5.

2.3 Eigenvalues and eigenvectors

Suppose that A is a matrix of order N and $\mathbf{x} \neq \mathbf{0}$ is a vector of the same order. An eigenvalue of A is a real or complex number λ such that

$$A\mathbf{x} = \lambda\mathbf{x}. \tag{2.3-1}$$

It is also called characteristic or latent root of A. An eigenvector of A is a vector $\mathbf{x}$ such that $\mathbf{x} \neq \mathbf{0}$ and (2.3-1) holds for some λ. This vector sometimes is also called the characteristic or latent vector of A.

The equation (2.3-1) can be written as

$$(A - \lambda I)\mathbf{x} = 0. \tag{2.3-2}$$

The non-trivial solution, $\mathbf{x} \neq \mathbf{0}$ to equation (2.3-2) exists if and only if the matrix of the system is singular, i.e., leads to the theorem 2.3-1.

Theorem 2.3-1

The number λ is an eigenvalue of A if and only if

$$det(A - \lambda I) = 0. \tag{2.3-3}$$

Obviously, (2.3-3) is a polynomial equation, referred to as the characteristic equation of A of the form

$$(-1)^N det(A - \lambda I) = \lambda^N - trace(A)\lambda^{N-1} + \ldots + (-1)det(A)$$
$$= 0. \tag{2.3-4}$$

Since the sum and product of the roots of (2.3-4) are $trace(A)$ and $det(A)$ respectively, thus we have

Theorem 2.3-2

If A is a matrix of order N with eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_N$, then

$$det(A) = \prod_{i=1}^{N} \lambda_i, \qquad trace(A) = \sum_{i=1}^{N} \lambda_i. \tag{2.3-5}$$

The left-hand side of (2.3-3) is called the characteristic polynomial of A, which can also be written as,

$$\alpha_0 + \alpha_1\lambda + \ldots + \alpha_{N-1}\lambda^{N-1} + (-1)^N\lambda^N = 0 \tag{2.3-6}$$

where α_i, $i = 0, 1, \ldots, N-1$ are constants.

It is clear that, since the coefficient of λ^N is not zero, the equation (2.3-6) has always N roots either real or complex which are the N eigenvalues of the matrix A, namely as $\lambda_1, \lambda_2, \ldots, \lambda_N$ (not necessarily having the same values), each of them possessing a unique corresponding eigenvector.

In physical problems we rarely need to find all the eigenvalues of equations (2.3-2). In particular, it is often necessary to determine the largest eigenvalue in modulus, where it is often termed as the dominant eigenvalue or spectral radius. In this section we consider one of the methods for obtaining the dominant eigenvalue with its corresponding eigenvector, called the *'Power Method'*.

Before we proceed further, first let us define the term *'normalise'*.

Definition 2.3-1

A vector is said to be a normalised vector if it is multiplied by a scalar in order to limit the element size without changing its direction.

The most useful normalisation for our work is described as follows: Suppose that $\mathbf{u} = [u_1, u_2, \ldots u_N]^T$. Then, in order to normalise $\mathbf{u}$ we select a scalar α such that $\alpha = max|u_i|$, $1 < i < N$, and the normalised vector is then given by $[u_1/\alpha, u_2/\alpha, \ldots u_N/\alpha]^T$. This method of normalisation ensures that the modulus of every element of the vector is less than or equal to 1.

The Power Method

Let us consider a matrix A of order N which has the eigenvalues λ_i, $i = 1, 2, \ldots, N$ such that one of them, λ_1 say, is the dominant eigenvalue, i.e.,

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \ldots \geq |\lambda_N|. \tag{2.3-7}$$

By assumption, there exists N linearly independent eigenvectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$, such that their linear combination can be expressed as an arbitrary vector $\mathbf{u}^{(0)}$, i.e.,

$$\mathbf{u}^{(0)} = \sum_{i=1}^{N} \alpha_i \mathbf{x}_i \tag{2.3-8}$$

where α_i, $i = 1, 2, \ldots, N$ are constants coefficients, not all zero. If we successively multiply $\mathbf{u}^{(0)}$ by matrix A, such that

$$\mathbf{u}^{(1)} = A\mathbf{u}^{(0)} \tag{2.3-9}$$

$$\mathbf{u}^{(2)} = A\mathbf{u}^{(1)} = A^2\mathbf{u}^{(0)} \tag{2.3-10}$$

. . . . .

$$\mathbf{u}^{(k)} = A\mathbf{u}^{(k-1)} = A^k\mathbf{u}^{(0)} \tag{2.3-11}$$

and from (2.3-1), for any eigenvalue λ_i, we have

$$A\mathbf{x}_i = \lambda_i \mathbf{x}_i, \qquad 1 \leq i \leq N. \tag{2.3-12}$$

Hence

$$A^2\mathbf{x}_i = \lambda_i A\mathbf{x}_i = \lambda_i^2 \mathbf{x}_i \tag{2.3-13}$$

. . . . .

. . . . .

$$A^k\mathbf{x}_i = \lambda_i^{k-1} A\mathbf{x}_i = \lambda_i^k \mathbf{x}_i. \tag{2.3-14}$$

Now, from (2.3-8) and (2.3-11) we have

$$\begin{aligned} \mathbf{u}^{(k)} &= A^k \mathbf{u}^{(0)} \\ &= A^k \sum_{i=1}^{N} \alpha_i \mathbf{x}_i = \sum_{i=1}^{N} \alpha_i A^k \mathbf{x}_i \\ &= \sum_{i=1}^{N} \alpha_i \lambda_i^k \mathbf{x}_i \end{aligned} \tag{2.3-15}$$

or $$\mathbf{u}^{(k)} = \lambda_1^k [\alpha_1 \mathbf{x}_1 + \sum_{i=2}^{N} \alpha_i (\frac{\lambda_i}{\lambda_1})^k \mathbf{x}_i], \qquad \alpha_1 \neq 0. \tag{2.3-16}$$

Now, from the initial assumption (2.3-7), it follows that $|\lambda_i/\lambda_1| < 1$, $i = 2, 3, \ldots, N$. Thus, the term $(\lambda_i/\lambda_1)^k$, $i = 2, 3, \ldots, N$ will converge to zero as $k \longrightarrow \infty$ so that

$$\mathbf{u}^{(k)} \longrightarrow \lambda_1^k \alpha_1 \mathbf{x}_1. \tag{2.3-17}$$

Hence, we can deduce that the ratio between the jth component, $1 \leq j \leq N$ of $\mathbf{u}^{(k+1)}$ and $\mathbf{u}^{(k)}$ tends to λ_1, i.e.,

$$\lim_{k \longrightarrow \infty} \left[\frac{u_j^{(k+1)}}{u_j^{(k)}} \right] = \lambda_1 \tag{2.3-18}$$

and where $\mathbf{u}^{(k+1)}$ is the un-normalised eigenvector of A corresponding to the eigenvalue λ_1.

It is clear that the rate of convergence depends upon the ratio $|\lambda_2/\lambda_1|$, where $\lambda_2 = max\ |\lambda_i|$, $2 < i < N$. The smaller the values of this ratio, the faster the convergence.

In order to avoid the possibility of overflow, we always normalise the vector $\mathbf{u}^{(k+1)}$, i.e., we divide by the element of largest modulus in the vector produced.

We now summarise the algorithm for the Power Method as follows:

Let $\mathbf{u}^{(0)}$ be the initial vector. Let $\mathbf{v}^{(1)} = A\mathbf{u}^{(0)}$, and $\alpha^{(1)}$ be the element of the largest modulus in $\mathbf{v}^{(1)}$.

Define $\mathbf{u}^{(1)} = \mathbf{v}^{(1)}/\alpha^{(1)}$. Then, the iteration proceeds with the steps below.

Step 1: $\mathbf{v}^{(k+1)} = A\mathbf{u}^{(k)}$.

Step 2: Determine $\alpha^{(k+1)}$, i.e., the element of largest modulus in $\mathbf{v}^{(k+1)}$.

Step 3: $\mathbf{u}^{(k+1)} = \mathbf{v}^{(k+1)}/\alpha^{(k+1)}$

Step 4: if $|\mathbf{u}^{(k+1)} - \mathbf{u}^{(k)}| < \varepsilon$, where ε is a given tolerance, then terminate the iteration, otherwise repeat from *Step* 1.

At the end, i.e., if the condition in *Step* 4 is satisfied, then the values of $\mathbf{u}^{(k+1)}$ and $\alpha^{(k+1)}$ will be a good approximation to $\mathbf{x}_1$ and λ_1 respectively.

Definition 2.3-2

Given a matrix A of order N with eigenvalues λ_i, $1 < i < N$, then

$$\mathcal{S}(A) = \max_i |\lambda_i| \qquad (2.3\text{-}19)$$

is the spectral radius of A.

Definition 2.3-3

Two matrices A and B of order N are similar if there exists a non-singular matrix P such that

$$B = P^{-1}AP. \qquad (2.3\text{-}20)$$

Matrix B is said to be obtained from matrix A by a similarity transformation and if B is symmetric then P will be orthogonal, i.e., $P^{-1} = P^T$, and hence

$$B = P^T A P. \tag{2.3-21}$$

The advantage of such a transformation is that the eigenvalues of A and B are the same. This can be shown as follows:

Let λ and $\mathbf{x}$ be the eigenvalue and the eigenvector of the matrix A respectively. Hence

$$A\mathbf{x} = \lambda\mathbf{x} \tag{2.3-22}$$

then premultiply by P^{-1}, we have

$$P^{-1}A\mathbf{x} = \lambda P^{-1}\mathbf{x}. \tag{2.3-23}$$

Thus, if $\mathbf{u} = P^{-1}\mathbf{x}$, then

$$\mathbf{x} = P\mathbf{u}. \tag{2.3-24}$$

Substituting (2.3-24) into (2.3-23), we get

$$P^{-1}AP\mathbf{u} = \lambda\mathbf{u} \tag{2.3-25}$$

or $$B\mathbf{u} = \lambda\mathbf{u}. \tag{2.3-26}$$

Thus, λ is the eigenvalue of B and $\mathbf{u}$ is the corresponding eigenvector.

Theorem 2.3-1 (Gerschgorin's Theorem)

If $A = [a_{i,j}]$ is a matrix of order N, then all the eigenvalues of A lie in the union of the discs,

$$|\lambda - a_{i,i}| \le \sum_{\substack{j=1 \\ j\neq i}}^{N} |a_{i,j}|, \qquad 1 \le i \le N. \tag{2.3-27}$$

Proof:

Let λ and $\mathbf{x}$ be the eigenvalue and the eigenvector of matrix A respectively. We can normalise $\mathbf{x}$ so that $\max_i |x_i| = 1$. Hence, from (2.3-1), we have

$$\lambda\mathbf{x} = A\mathbf{x} \tag{2.3-28}$$

$$\lambda x_i = \sum_{j=1}^{N} a_{i,j} x_j, \qquad 1 \le i \le N \tag{2.3-29}$$

i.e., $$(\lambda - a_{i,i}) x_i = \sum_{\substack{j=1 \\ j \neq i}}^{N} a_{i,j} x_j, \qquad 1 \le i \le N. \tag{2.3-30}$$

Now, $|x_k| = 1$, then

$$|\lambda - a_{k,k}| \le \sum_{\substack{j=1 \\ j \neq k}}^{N} |a_{k,j}||x_j| \le \sum_{\substack{j=1 \\ j \neq k}}^{N} |a_{k,j}| = D_k. \tag{2.3-31}$$

Thus, the eigenvalue λ lies within the disc D_k, say, and since λ is arbitrary, then it follows that all the eigenvalues of A must lie in the union of discs, i.e.,

$$|\lambda - a_{i,i}| \le \sum_{\substack{j=1 \\ j \neq i}}^{N} |a_{i,j}|, \qquad 1 \le i \le N.$$

Corollary 2.3-1

If $A = [a_{i,j}]$ is a matrix of order N and we have

$$v_1 = \max_{1 \le i \le N} \sum_{j=1}^{N} |a_{i,j}|, \tag{2.3-32}$$

$$v_2 = \max_{1 \le j \le N} \sum_{i=1}^{N} |a_{i,j}|, \tag{2.3-33}$$

then

$$\rho(A) \le \min (v_1, v_2). \tag{2.3-34}$$

The condition (2.3-34) is a direct consequence of the facts that A and A^T have the same eigenvalues.

Now, let A be a tridiagonal matrix of order N

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

where a, b and c may be real or complex numbers, then the eigenvalues of A are given by

$$\lambda_k = a + 2\sqrt{bc}\ cos\ \frac{k\pi}{N+1}, \qquad k = 1, 2, \ldots, N. \tag{2.3-35}$$

For the proof, see Smith [1978], page 113.

2.4 Vector and matrix norms

It is important to have some measures of the size or magnitude of a vector or matrix. This measure is called norm and is denoted by $\| \cdot \|$.

Definition 2.4-1

The norm of a vector $\mathbf{u}$, denoted by $\| \mathbf{u} \|$, is a non-negative number satisfying the following three axioms:

1). $\| \mathbf{u} \| = 0$, for $\mathbf{u} = \mathbf{0}$ and $\| \mathbf{u} \| > 0$ if $\mathbf{u} \neq \mathbf{0}$. (2.4-1)

2). $\| \alpha\mathbf{u} \| = |\alpha| . \| \mathbf{u} \|$ for any complex scalar α, (2.4-2)

3). $\| \mathbf{u} + \mathbf{v} \| \leq \| \mathbf{u} \| + \| \mathbf{v} \|$ for any vectors $\mathbf{u}$ and $\mathbf{v}$. (2.4-3)

The axiom (2.4-3) is called the triangle inequality.

Also from (2.4-3), we have

$$\| \mathbf{u} - \mathbf{v} \| \geq \| \mathbf{u} \| - \| \mathbf{v} \|. \tag{2.4-4}$$

The most commonly used norms are the L_1, L_2 and L_∞ norms of $\mathbf{u}$ and they are defined as follows:

Definition 2.4-2

If $\mathbf{u} = [u_1, u_2, \ldots, u_N]^T$ is a vector of order N then

$$L_1 = \| \mathbf{u} \|_1 = \sum_{i=1}^{N} |u_i| \tag{2.4-5}$$

$$L_2 = \| \mathbf{u} \|_2 = \left[\sum_{i=1}^{N} |u_i|^2 \right]^{\frac{1}{2}}, \quad \text{(Euclidean norm)} \tag{2.4-6}$$

$$L_\infty = \| \mathbf{u} \|_\infty = \max_i |u_i|, \quad \text{(maximum or uniform norm).} \tag{2.4-7}$$

For these three special cases, hence we can define the general L_p for $p > 1$,

$$L_p = \| \mathbf{u} \|_p = \left[\sum_{i=1}^{N} |u_i|^p \right]^{\frac{1}{p}}. \tag{2.4-8}$$

Similarly, we can proceed to define the matrix norm.

Definition 2.4-3

A norm of a matrix A of order N, denoted as $\| A \|$, as a scalar satisfying the following axioms:

1). $\| A \| > 0$ and $\| A \| = 0$ if and only if $A = [0]$ (2.4-9)

2). $\| \alpha A \| = |\alpha| . \| A \|$ for any scalar α, (2.4-10)

3). $\| A + B \| \leq \| A \| + \| B \|$ for any matrices A and B. (2.4-11)

4). $\| AB \| \leq \| A \| . \| B \|$ for any matrices A and B. (2.4-12)

In similar fashion, L_1, L_2 and L_∞ are given by

$$L_1 = \| A \|_1 = \max_j \sum_{i=1}^{N} |a_{i,j}|, \tag{2.4-13}$$

(maximum absolute column sum)

$$L_2 = \| A \|_2 = (\text{maximum of } A^H A)^{1/2} \quad \text{(Spectral norm)} \tag{2.4-14}$$

$$L_\infty = \max_i \sum_{j=1}^{N} |a_{i,j}|, \quad \text{(maximum absolute row sum).} \tag{2.4-15}$$

A norm compatible with the L_2 vector norm is the Euclidean or Schür norm and is defined as follows:

$$L_2 = \| A \|_E = \left[\sum_{i,j} |a_{i,j}|^2 \right]^{1/2}. \tag{2.4-16}$$

Definition 2.4-4

A matrix norm $\| A \|$ is said to be compatible with a vector norm $\| \mathbf{u} \|$ if

$$\| A\mathbf{u} \| \leq \| A \| . \| \mathbf{u} \|, \text{ for all } \mathbf{u} \neq \mathbf{0}. \tag{2.4-17}$$

From the definition (2.3-4) we can show that for any matrix A of order N and any norm,

$$\mathcal{S}(A) \leq \| A \|. \tag{2.4-18}$$

Proof:

Let λ_i and $\mathbf{u}_i$ be an arbitrary eigenvalue and eigenvector of matrix A respectively, then

$$A\mathbf{u}_i = \lambda_i \mathbf{u}_i \tag{2.4-19}$$

and

$$|\lambda_i| \cdot \| \mathbf{u}_i \| = \| A\mathbf{u}_i \| \leq \| A \| \cdot \| \mathbf{u}_i \| \tag{2.4-20}$$

for any compatible norm. Thus, $|\lambda_i| \leq \| A \|$. Since $|\lambda_i|$ was arbitrarily chosen, hence from definition (2.3-1), then

$$\mathcal{S}(A) \leq \| A \|.$$

Definition 2.4-5

A matrix norm is said to be subordinate to the corresponding vector norm if it can be constructed in the following form:

$$\| A \| = \max_{\mathbf{u} \neq 0} \frac{\| A\mathbf{u} \|}{\| \mathbf{u} \|} \tag{2.4-21}$$

or equivalent to

$$\| A \| = \max \| A\mathbf{u} \|, \qquad \| \mathbf{u} \| = 1. \tag{2.4-22}$$

2.5 Positive definite and special matrices

There are many definitions for the property of positive definiteness of a matrix A.

Definition 2.5-1

If a matrix A of order N is Hermitian, and the quadratic form

$$(\mathbf{x}, A\mathbf{x}) > 0 \tag{2.5-1}$$

for all $\mathbf{x} \neq 0$, then A is positive definite.

The matrix A is non-negative definite if $(\mathbf{x}, A\mathbf{x}) \geq 0$. The other definition of positive definiteness of a matrix A, is stated by the following theorem.

Theorem 2.5-1

The necessary and sufficient condition for a Hermitian or a real symmetric matrix A to be positive definite is that, the eigenvalues of A are all positive.

Theorem 2.5-2

An irreducible, diagonally dominant matrix which is also symmetric and has positive real diagonal elements is positive definite. The proof of these theorems may be found in Young [1971].

The definition of some special matrices are given as follows:

Theorem 2.5-2

If $A = [a_{i,j}]$ is a real matrix of order N then it is said to be

1). an L-matrix if

$$a_{i,i} > 0, \quad i = 1, 2, \ldots, N \quad \text{and} \tag{2.5-2}$$

$$a_{i,j} \leq 0, \quad i, j = 1, 2, \ldots, N \tag{2.5-3}$$

2). a Stieltjes matrix if A is positive definite and if (2.5-3) holds

3). an M-matrix if A is non-singular matrix, if (2.5-3) holds and if $A^{-1} > 0$.

It should be noticed that, by stating $A > 0$, we mean that all elements of the matrix A are real and non-negative. A simple matrix A of order 3 which satisfies these three definition of special matrices is

$$A = \begin{bmatrix} 4 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 4 \end{bmatrix}$$

2.6 Property $\mathcal{A}$ and consistently ordered matrices

The property $\mathcal{A}$ of a matrix A of order N can be defined as follows:

Definition 2.6-1

A matrix $A = [a_{i,j}]$ of order N is said to have property $\mathcal{A}$ if there exists two disjoint subset S and T of $W = \{1, 2, \ldots, N\}$ such that if $i \neq j$ and if either $a_{i,j} \neq 0$ and $a_{j,i} \neq 0$, then $i \in S$ and $j \in T$ or else $i \in T$ and $j \in S$.

The property A can also be defined as follows:

Definition 2.6-2

A matrix A of order N has Property $\mathcal{A}$ if there exists a permutation matrix P such that PAP^T has the form

$$PAP^T = \begin{bmatrix} D_1 & F \\ E & D_2 \end{bmatrix} \tag{2.6-1}$$

where D_1 and D_2 are square diagonal matrices.

Now, we define the consistently ordered matrix as follows.

Definition 2.6-3

A matrix A of order N is consistently ordered if for some t there exists disjoint subsets $S_1, S_2, \ldots, S_t$ of $W = \{1, 2, \ldots, N\}$ such that

$$\sum_{k=1}^{N} S_k = W$$

and are such that if i and j are associated, then $j \in S_{k+1}$ if $j > i$ and $j \in S_{k-1}$ if $j < 1$ where S_k is the subset containing i.

The following theorem, can be used as an alternative definition of a consistently ordered matrix.

Theorem 2.6-1

If A is T-matrix, then A is consistently ordered.

The proof of this theorem can be referred to in Young [1971].

Our discussion also concern the ordering vector for the matrix A. This is defined as follows.

Definition 2.6-4

A column vector v of order N with integer elements, is an ordering vector for the matrix A of order N if for any pair of associated integers i and j with $i \neq j$ we have $|v_i - v_j| = 1$.

Further, we define a compatible ordering vector for the matrix A as follows:

Definition 2.6-5

An ordering vector $\mathbf{v} = [v_1, v_2, \ldots, v_N]^T$ for the matrix A of order N is a compatible ordering vector of A if:

1). $v_i - v_j = 1$ if i and j are associated and $i > j$.

2). $v_i - v_j = -1$ if i and j are associated and $i < j$.

We complete this section with the following theorem.

Theorem 2.6-2

There exists an ordering vector for a matrix A if and only if A has Property $\mathcal{A}$.

Moreover, if A is consistently ordered, then the matrix A has Property $\mathcal{A}$. Again, the proof may be referred to in Young [1971].

2.7 Rate of convergence

Let us define a linear stationary iterative method as

$$\mathbf{u}^{(k+1)} = B\mathbf{u}^{(k)} + C \qquad (2.7\text{-}1)$$

where B is called the *iteration matrix*.

Practically, even if the method (2.7-1) converges, it may converge very slowly. Hence, it is important to evaluate the effectiveness of an iterative method. To carry this out, we should consider both the computation required for each iteration and the number of iterations required for convergence at a given accuracy.

Before we proceed further, let us discuss the convergence of sequence of matrices.

A sequence of matrix $A^{(1)}$, $A^{(2)}$, $A^{(3)}$, ... all of the same order is said to converge to a limit A if

$$\lim_{k \longrightarrow \infty} A^{(k)} = A. \tag{2.7-2}$$

Theorem 2.7-1

The sequence of matrices $A^{(1)}$, $A^{(2)}$, $A^{(3)}$, ... converges to A if and only if for every matrix norm $\| \cdot \|_\beta$,

$$\lim_{k \longrightarrow \infty} \| A^{(k)} - A \|_\beta = 0. \tag{2.7-3}$$

Theorem 2.7-2 (Varga [1962], page 13)

If A is a matrix of order N, then A is convergent if and only if

$$\mathcal{S}(A) < 1. \tag{2.7-4}$$

Theorem 2.7-3

The matrix $I-B$ is non-singular and the series $I + B + B^2 + \ldots$ converges if and only if $\mathcal{S}(B) < 1$. Moreover, if $\mathcal{S}(B) < 1$, then

$$(I-B)^{-1} = I + B + B^2 + \ldots = \sum_{i=0}^{\infty} B^i. \tag{2.7-5}$$

The proof of these theorems may be found in Young [1971].

Now, we state the basic convergence criterion of the method (2.7-1).

We say that the method (2.7-1) converges if

$$\lim_{k \longrightarrow \infty} u_i^{(k)} = u_i, \text{ for all } i \tag{2.7-6}$$

and for all starting vectors $\mathbf{u}^{(0)}$.

Theorem 2.7-4

The iterative method (2.7-1) is convergent if and only if

$$\rho(B) < 1. \tag{2.7-7}$$

Thus, from equation (2.4.14), we have the following corollary.

Corollary 2.7-1

A sufficient condition for convergence of (2.7-1) is merely that

$$\| B \| < 1. \tag{2.7-8}$$

As we have stated earlier, the rate of convergence of the method (2.7-1) may be determined by calculating the number of iterations at a predetermined accuracy. In practice, the usual approach is to iterate until the norm of the error vector $\mathbf{e}^{(k)}$ is reduced to less than some given tolerance, say ε, of the norm of the initial vector $\mathbf{e}^{(0)}$. If we define the error vector after k iterations as

$$\mathbf{e}^{(k)} = \mathbf{u}^{(k)} - \mathbf{U} \tag{2.7-9}$$

where $\mathbf{U}$ is the exact vector solution of (2.2-3), then applying the method (2.7-1) we would have

$$\mathbf{e}^{(k+1)} = B\mathbf{e}^{(k)} \tag{2.7-10}$$

and hence

$$\begin{aligned} \mathbf{e}^{(k)} &= B\mathbf{e}^{(k-1)} \\ &= B^2\mathbf{e}^{(k-2)} = \ldots = B^k\mathbf{e}^{(0)}. \end{aligned} \tag{2.7-11}$$

From (2.7-11) we have,

$$\begin{aligned} \| \mathbf{e}^{(k)} \| &= \| B^k\mathbf{e}^{(0)} \| \\ &\leq \| B^k \| \, \| \mathbf{e}^{(0)} \|. \end{aligned} \tag{2.7-12}$$

Then if $\mathbf{e}^{(0)} \neq 0$, then

$$\frac{\| \mathbf{e}^{(k)} \|}{\| \mathbf{e}^{(0)} \|} \leq \| B^k \|. \tag{2.7-13}$$

We require,

$$\| e^{(k)} \| \leq \varepsilon \| e^{(0)} \| \tag{2.7-14}$$

where $\| \cdot \|$ denotes $\| \cdot \|_2$ as defined in section 2.4. By theorem (2.7-4) we know that $\| B^k \|$ converges to zero as $k \longrightarrow \infty$ if and only if $\rho(B) < 1$. Hence, equation (2.7-14) can be satisfied by choosing k sufficiently large so that

$$\| B^k \| \leq \varepsilon. \tag{2.7-15}$$

If k is large enough so that $\| B^k \| < 1$, then it follows that (2.7-15) may be written as

$$k \geq - \log \varepsilon \; / \; (- \frac{1}{k} \log \| B^k \|). \tag{2.7-16}$$

From this inequality, we can determine a lower bound for the number of iterations for the iterative method (2.7-1).

Definition 2.7-1

For any convergent iterative method of the form (2.7-1), the quantity

$$R_k(B) = - \frac{1}{k} \log \| B^k \| \tag{2.7-17}$$

is called the average rate of convergence after k iterations.

If $R_k(B_1) < R_k(B_2)$ for matrices B_1 and B_2 then for k iterations, B_2 is iteratively faster than B_1.

Definition 2.7-2

The asymptotic average rate of convergence is defined by

$$R(B) = \lim_{k \longrightarrow \infty} R_k(B) = - \log \rho(B). \tag{2.7-18}$$

It is true, since

$$\rho(B) = \lim_{k \longrightarrow \infty} (\| B^k \|)^{1/k} \tag{2.7-19}$$

as proved by Young [1971].

It is usual for iterative methods to converge slowly for substantially large problems corresponding to the values of $\mathcal{S}(B)$ only slightly less than one, and a rate of convergence nearly zero.

Now, upon replacing $\| B^k \|$ by $[\mathcal{S}(B)]^k$ in equation (2.7-16), we see that $\varepsilon \approx [\mathcal{S}(B)]^k$, we can determine a rough estimation of the number of iteration k, as

$$k \approx \frac{-\log \varepsilon}{-\log \mathcal{S}(B)} = \frac{-\log \varepsilon}{R(B)} . \qquad (2.7\text{-}20)$$

However, the value of k from (2.7-20) could be very much lower when compared with the number required, in which $\| B^k \|$ will behave like $k[\mathcal{S}(B)]^{k-1}$, rather than $[\mathcal{S}(B)]^k$, as mentioned by Young [1971]. In this case, the smallest value of k such that

$$k[\mathcal{S}(B)]^{k-1} \leq \varepsilon \qquad (2.7\text{-}21)$$

estimates the number of iteration required more accurately.

Finally, in this section, we discuss briefly the convergence test. There are many convergence tests which may be used in order to determine the number of iterations for a given tolerance. It is obvious that a different stopping criterion will yield a different number of iterations, but the better the test, the better the accuracy which is achieved. In this thesis, we shall use the average test, i.e.,

$$\frac{\| u_i^{(k+1)} - u_i^{(k)} \|}{\| 1 + u_i^{(k)} \|} < \varepsilon \qquad \text{for all } i, \qquad (2.7\text{-}22)$$

where $\varepsilon = 10^{-5}$.

For a small values of $u_i^{(k)}$, this test approximates the absolute test $\| u_i^{(k+1)} - u_i^{(k)} \|$, and for large values of $u_i^{(k)}$, it approximates the relative test, i.e., $(\| u_i^{(k+1)} - u_i^{(k)} \| / \| u_i^{(k)} \|)$, for all i.

2.8 Finite difference approximation

It is important to note that not all the ordinary and partial differential equations can be conveniently solved analytically. Very often the solution is so complicated that any attempt to evaluate it analytically is worthless. Furthermore, in partial differential equations, it is almost impossible to determine an analytical solution if some changes are made to the shape of the area of integration, or to the initial and/or boundary conditions.

With the high-speed computing machine of today at our disposal, we turn to discrete numerical methods of solving such differential equations, which are expected to be more amenable. In discrete numerical methods the continuous systems are reduced to *'equivalent'* discrete systems which are suitable for high-speed computer solution. The development of discrete approximations can proceed in several ways, notably finite difference methods, finite elements methods, variational methods and the method of lines. Here, we confine ourselves to the finite difference methods only.

Let us consider the general two-point boundary value problem

$$\frac{d^2U}{dx^2} = f(x,U,\frac{dU}{dx}), \qquad a < x < b \tag{2.8-1}$$

$$U(a) = \alpha, \qquad U(b) = \beta. \tag{2.8-2}$$

We will employ the finite difference methods to solve (2.8-1) numerically. In the finite difference procedure, we choose equal mesh points $a = x_0 < x_1 < x_2 < \ldots < x_{N-1} < x_N < x_{N+1} = b$ where $x_i = x_0 + ih$. It is obvious that our boundary condition is given by $x_0 = a$ and $x_{N+1} = b$, whilst the interior mesh points are x_i, $i = 1, 2, \ldots, N$. We next replace the derivatives of (2.8-1) by suitable difference quotients defined in terms of the points x_i.

The well known method for deriving finite difference approximations to problem (2.8-1) with a given condition (2.8-2) is based on finite Taylor's series expansion of the solution $U(x)$ to that problem.

A typical one is to use the centered difference approximations with the local truncation error

$$\frac{d}{dx}[U(x_i)] = \frac{U(x_{i+1}) - U(x_{i-1})}{2h} - \frac{h^2}{6}\left(\frac{d^3U}{dx^3}\right)_{x_i} + \ldots \tag{2.8-3}$$

$$\frac{d^2}{dx^2}[U(x_i)] = \frac{U(x_{i-1}) - 2U(x_i) + U(x_{i+1})}{h^2} - \frac{h^2}{12}\left(\frac{d^4U}{dx^4}\right)_{x_i} + \ldots \tag{2.8-4}$$

Now, let denote $u_i = U(x_i)$ and neglecting the truncation error by choosing h to be small, then

$$\frac{du}{dx}_i = \frac{u_{i+1} - u_{i-1}}{2h} \tag{2.8-5}$$

$$\frac{d^2u}{dx^2}_i = \frac{u_{i-1} - 2u_i + u_{i+1}}{h^2} . \tag{2.8-6}$$

When the approximations (2.8-5) and (2.8-6) are used in (2.8-1), we obtain a system of equations of the form

$$\frac{u_{i-1} - 2u_i + u_{i+1}}{h^2} = f(x_i, u_i, \frac{u_{i+1} - u_{i-1}}{2h}) \tag{2.8-7}$$

$$i = 1, 2, \ldots, N$$

$$u_0 = \alpha, \quad u_{N+1} = \beta. \tag{2.8-8}$$

It is obvious that the order of the accuracy is $O(h^2)$.

In (2.8-7) we have allocated $u_1, u_2, \ldots, u_N$ as the unknown vector since any solution $U(x)$ of (2.8-1) will probably not satisfy (2.8-7) exactly. We consider equation (2.8-7) as an approximation to equation (2.8-1) and we hope that for sufficiently small h, the solution u_i of (2.8-7) are good approximation to $U(x_i)$. In this case, $h = (b-a)/(N+1)$.

It should be noticed that the functions $f(x,U,\frac{dU}{dx})$ can either be linear or non-linear and we faced with solving a system of N equations in N unknowns. In practice, the function $f(x,U,\frac{dU}{dx})$ is frequently linear in U and $\frac{dU}{dx}$.

Equations (2.8-7) and (2.8-8) can be written in a matrix form

$$A\mathbf{u} = \mathbf{b}, \tag{2.8-9}$$

where A is a tridiagonal matrix, as given in (2.8-10),

$$A = \begin{bmatrix} a_1 & d_1 & & & \\ c_2 & a_2 & d_2 & & 0 \\ & \ddots & \ddots & \ddots & \\ & & c_{N-1} & a_{N-1} & d_{N-1} \\ 0 & & & c_N & a_N \end{bmatrix} \tag{2.8-10}$$

and the vector $\mathbf{u}$ and $\mathbf{b}$ are

$$\mathbf{u} = [u_1,\ u_2,\ \ldots\ u_{N-1},u_N]^T,\ \mathbf{b} = [b_1,\ b_2,\ \ldots\ b_{N-1},\ b_N]^T$$

respectively.

Let us consider the special second order equation

$$\frac{d^2U}{dx^2} = f(x,U), \qquad a < x < b \tag{2.8-11}$$

$$U(a) = \alpha, \qquad U(b) = \beta \tag{2.8-12}$$

in which $\frac{dU}{dx}$ does not appear explicitly. Such equations are common in trajectory problems or elsewhere. The *Numerov method*, based on the formula

$$u_{i-1} - 2u_i + u_{i+1} \approx \frac{h^2}{12}[f_{i-1} + 10f_i + f_{i+1}],\ i = 1,\ \ldots,\ N \tag{2.8-13}$$

where $f_i = f(x_i,U(x_i))$, is convenient here, since we do not need to include the computation of $\frac{dU}{dx}$. This method gives a better accuracy compared to the usual finite difference methods, i.e., with accuracy of order $O(h^4)$.

The derivation of the Numerov method can be shown as follows. Let us write the Numerov method with some undetermined coefficients as

$$u_{i+1} = Au_i + Bu_{i-1} + h^2[Cf_{i+1} + Df_i + Ef_{i-1}] + R. \qquad (2.8\text{-}14)$$

By Taylor's expansion,

$$u_{i+1} = u_i + h\left(\frac{dU}{dx}\right)_i + \frac{h^2}{2}\left(\frac{d^2U}{dx^2}\right)_i + \frac{h^3}{6}\left(\frac{d^3U}{dx^3}\right)_i + \frac{h^4}{24}\left(\frac{d^4U}{dx^4}\right)_i + \frac{h^5}{120}\left(\frac{d^5U}{dx^5}\right)_i + \frac{h^6}{720}\left(\frac{d^6U}{dx^6}\right)_i \qquad (2.8\text{-}15)$$

$$u_{i-1} = u_i - h\left(\frac{dU}{dx}\right)_i + \frac{h^2}{2}\left(\frac{d^2U}{dx^2}\right)_i - \frac{h^3}{6}\left(\frac{d^3U}{dx^3}\right)_i + \frac{h^4}{24}\left(\frac{d^4U}{dx^4}\right)_i - \frac{h^5}{120}\left(\frac{d^5U}{dx^5}\right)_i + \frac{h^6}{720}\left(\frac{d^6U}{dx^6}\right)_i \qquad (2.8\text{-}16)$$

$$h^2f_{i+1} = h^2\left(\frac{d^2U}{dx^2}\right)_i + h^3\left(\frac{d^3U}{dx^3}\right)_i + \frac{h^4}{2}\left(\frac{d^4U}{dx^4}\right)_i + \frac{h^5}{6}\left(\frac{d^5U}{dx^5}\right)_i + \frac{h^6}{24}\left(\frac{d^6U}{dx^6}\right)_i \qquad (2.8\text{-}17)$$

$$h^2f_{i-1} = h^2\left(\frac{d^2U}{dx^2}\right)_i - h^3\left(\frac{d^3U}{dx^3}\right)_i + \frac{h^4}{2}\left(\frac{d^4U}{dx^4}\right)_i - \frac{h^5}{6}\left(\frac{d^5U}{dx^5}\right)_i + \frac{h^6}{24}\left(\frac{d^6U}{dx^6}\right)_i \qquad (2.8\text{-}18)$$

and matching the powers of h through to the fourth power on both sides of equation (2.8-14), we have

$$A + B = 1, \qquad B = -1, \qquad \tfrac{1}{2}B + C + D + E = \tfrac{1}{2},$$

$$\tfrac{1}{6}B - C + E = -\tfrac{1}{6} \quad \text{and} \quad \tfrac{1}{24}B + \tfrac{1}{2}C + \tfrac{1}{2}E = \tfrac{1}{24}.$$

These give the values $A = 2$, $B = -1$, $C = E = \frac{1}{12}$, and $D = \frac{1}{6}$.

The fifth power also match voluntarily. If we pretend that all factors designated as $\left(\frac{d^6U}{dx^6}\right)$ are the same, then the local truncation error is $R = -\frac{h^6}{240}\left(\frac{d^6U}{dx^6}\right)_i$.

The form of (2.8-7) and (2.8-13) can now be solved directly or by using some well known iterative methods. We will study these methods in the next chapter.

2.9 Symbolic computation - an introduction to REDUCE and Mathematica

The purpose of this section is to introduce the new tool of symbolic algebraic computation (or symbolic computation, in short) and point out its potential in the numerical modelling and simulation of field problems.

Symbolic computation refers to the technique of manipulating on a computer, symbolic expressions that may not necessarily have numerical values. Therefore, techniques of symbolic computation can be used, among other things, to perform algebraic manipulations of mathematical formulae. Roughly, one can think of symbolic computation as a computerised version of the traditional *"pencil and paper"* manipulations of algebraic expressions commonly arising in Applied Mathematics. Thus, symbolic computation can significantly reduce the tedium of analytic calculations and *increase their reliability*. This capability enables one to carry on the analytical calculations before numerical computation start.

A number of symbolic manipulation systems suitable for manipulating algebraic expressions have been developed over the past few years such as *REDUCE, SYMBAL, MACSYMA* and *Mathematica*. In this section, we will discuss some capabilities of the symbolic computation *REDUCE* and a short introduction on *Mathematica*, i.e., a system for doing Mathematics by computer. We commence the discussion for the symbolic computation with an introduction to *REDUCE*.

REDUCE

REDUCE is a system for carrying out algebraic operations accurately on relatively complicated expressions. It can manipulate polynomials in a variety of forms, both expanding and factoring them, and extracting various parts of them as required. *REDUCE* can also do differentiation and integration, as well as the manipulation of the arrays and in the operations of matrices. These areas are the topics of interests to the physicists, mathematicians and engineers.

REDUCE is also designed to be an interactive system, so that an algebraic expression can be input and its value inspected before moving on to the next calculation. However, *REDUCE* can also be used in a batch mode by inputting a sequence of calculations and obtaining results without any possibility of interaction during calculations.

To show the interactive use of *REDUCE*, we shall give some examples which illustrates comprehensively the capabilities of the system. For example, one may wish to find the expansion of $(x + y)^4$. After logging the SUNA at Loughborough University of Technology successfully, one then enter (in lower case)

```
reduce
```

after which *REDUCE* will respond with a banner line and then prompt for the first line of input as follows:

```
REDUCE 3.4.1, 15-Jul-92 ...

1:
```

where (1:) is automatically assigned to the first command. We can now begin entering a command by typing a FORTRAN-like expression, terminated by a semi-colon as follows:

```
1: (x+y)**4;
```

The semi-colon indicates the end of expression. By pressing the RETURN key, the system would then input the expression, evaluate it, and return the result in a form like:

$$X^4 + 4*X^3*Y + 6*X^2*Y^2 + 4*X*Y^3 + Y^4$$

2:

where (2:) is automatically assigned to the second command. Input may be in the lower or upper case, but the lower case is converted to the upper case by the system, such that the output is in the upper case.

The results of a given calculation are saved in the variable WS (for workspace), so this can be used in the next calculation for further processing. For example, one could enter on line (2:) the expression

```
df(ws,y);
```

which calculates the derivatives of the previous evaluation with respect to *y*, and *REDUCE* responds with

$$4*(X^3 + 3*X^2*Y + 3*X*Y^2 + Y^3).$$

Alternatively,

```
int(ws,x);
```

would calculate the integral of the same expression with respect to *x* and *REDUCE* responds with

$$[X*(X^4 + 5*X^3*Y + 10*X^2*Y^2 + 10*X*Y^3 + 5*Y^4)]/5.$$

In many cases, it is necessary to use the result of one calculation in succeeding calculations. One way to do this is via an assignment for a variable, such as,

```
v := (x+y)**4;
```

If we now use *v* in later calculation, the value of the right hand side of the above will be used.

An important class of commands in *REDUCE* is that which defines substitutions for variables and expressions to be made during the evaluation of expressions. Such substitutions use forms of the command LET. LET rules stay in effect until replaced or CLEARed. For example, after assigning the expression $(x+y)^4$ to v, we can give numerical values to x and y and hence find the numerical values of v by using the command LET as follows:

```
let x = 3, y = 1;

v;
```

REDUCE responds with

```
256.
```

But if we want to have the value assigned to another variable u (say), we proceed as follows:

```
let x = 1, y = 2;

u := v;
```

REDUCE then responds with

```
U := 81.
```

Another powerful feature of the *REDUCE* system is the handling of symbolic matrix calculations which can be performed easily. For example to find the inverse matrix Q of a (4,4) matrix P given by

$$P = \begin{bmatrix} d & b & c & 0 \\ b & d & b & c \\ c & b & d & b \\ 0 & c & b & d \end{bmatrix}$$

and then evaluate the matrix S (say), which results from multiplying the matrix Q by the matrix R, where R is given by:

$$R = = \begin{bmatrix} -a & c & 1 \\ b & -a & c \\ c & b & -a \\ 1 & c & b \end{bmatrix}$$

The input to the system *REDUCE* can be written as follows:

The Input	Comments
`matrix p(4,4), q(4,4), r(4,3), s(4,3);`	Specify the dimension of the matrices *P*, *Q*, *R* and *S*.
`p:=mat((d,b,c,0),(b,d,b,c),` `(c,b,d,b),(0,c,b,d));`	Assign to the matrix *P* its elements.
`r:=mat((-a,c,1),(b,-a,c),(c,b,-a),` `(1,c,b));`	Assign to the matrix *R* its elements.
`q:=1/p;`	Find the inverse of matrix *P* and assign it to matrix *Q*
`s:=q*r;`	Evaluate the matrix product of *P* and *Q* and assign the results to *S*.

In many applications, it is necessary to load previously prepared *REDUCE* files into the system, or to write output on other files. *REDUCE* offers some commands for this purpose, two of these commands are IN and OUT. The command IN takes a list of file names as argument and directs the system to input each file (which should contain *REDUCE* statements and commands) into the system. For example,

```
in matrix-1, "matrix-2";
```

will first load file matrix-1, then matrix-2. Files to be read using IN should end with ; END;.

The command OUT takes a single file name as argument, and directs the output to that file from then on, until another OUT changes the output file, or SHUT closes it. For example,

```
out matrix-3;
```

will direct output to the file matrix-3.

To get out from the *REDUCE* system we use the command BYE. This command stops the execution of *REDUCE* and return us to the computer system monitor program.

Mathematica

Mathematica is a general computer software system and language intended for mathematical and other applications. It is currently used by many researchers, engineers and analysts, as well as students at school up to the university level. The applications of *Mathematica* span all areas of science, technology and business where quantitative methods are used.

Mathematica was first released in June 1988 for the Apple Macintosh computers. Versions for other workstations soon followed, including the version for 386-based IBM PC compatible was released which running under MS-DOS and Microsoft Windows. This version needs 640 kilobytes based memory and 4 megabytes Random Access Memory (RAM).

Mathematica can be used as a numerical and symbolic calculator where the question can be keyed in, and in return, *Mathematica* prints out the answer. It is can also be used as a visualisation system for function and data. *Mathematica* can be used as a high-level programming language in which we can create a program, either large or small. It may also be used as a modelling and data analysis environment.

Another facility is the creation of interactive documents that can mix texts, animated graphics and sound with active formulas and many others.

The *"dialog"* with *Mathematica* is fairly simple. The text on the lines labeled *In[n]:=* is the input that we enter, and the lines labeled *Out[n]:=* are the response from *Mathematica* for a given question. Let us consider an example as follows.

To solve the integral $\int x^4/(x^2-1)\,dx$. The dialog is as follows.

Enter the expression $x^4/(x^2-1)$ in *Mathematica*.

```
In[1]:= x^4/(x^2 - 1)
```

Mathematica will response by returning the answer

$$Out[1]:= \frac{x^4}{-1 + x^2}\,.$$

Now, we instruct *Mathematica* to integrate the expression by entering

```
In[2]:= Integrate[%,x]
```

and the result given by *Mathematica* is,

$$Out[2]:= x + \frac{x^3}{3} + \frac{\text{Log}[-1 + x]}{2} - \frac{\text{Log}[1 + x]}{2}\,.$$

Other examples can be obtained from Wolfram [1991].

There are several reasons why symbolic computations are useful in the context of modelling and simulation of field problems. Some of the reasons cited by Brown and Hearn [1978] are listed as follows.

1. Sometimes it is too costly, or even impossible, to solve an essentially numerical problem by purely numerical means because it involves too many variables, requires greater accuracy, or is presented in an ill-conditioned or intractable form. However, a symbolic transformation may reduce the dimensionality, evade a large source of round-off error, improve the ill-conditioning, and otherwise change the

problem into one that can be solved by standard numerical methods.

2. The algebraic result obtained via symbolic computation can be subsequently evaluated over a wide range of parameter values.

3. Symbolic computation provides an opportunity for realising the vital computational symbiosis between numerical experiments and symbolic theories.

4. Symbolic computation can be used to generate a much needed computational formula.

PART II

THE NUMERICAL SOLUTION OF ORDINARY DIFFERENTIAL EQUATIONS

CHAPTER 3

THE SOLUTION OF THE TWO POINT BOUNDARY VALUE PROBLEMS

3.1 Methods for solving the problem

The application of finite difference approximations to ordinary differential equations often generates an associated algebraic problem. In many cases, even for some nonlinear problem, one must solve a large set of a simultaneous linear equations

$$A\mathbf{u} = \mathbf{b} \qquad (3.1\text{-}1)$$

where A is a square (often sparse) matrix, $\mathbf{b}$ is a known column vector, and $\mathbf{u}$ is the unknown column vector.

Methods of solution for general computational problems, i.e., to obtain the solution vector $\mathbf{u}$, can be classified into two categories - the *direct* and *iterative* methods. Direct methods of which the solution of a tridiagonal system is typical, are those which yield the exact answer in a finite number of steps, if there were no round-off error. The accuracy of the final solution usually turn out to be satisfactory depending on the word length of the machine. The algorithm for such a method is often complicated and non-repetitive. These methods have usually been omitted from consideration because of excessive computer storage requirements, both in program and in the necessity to store many intermediate results for later use.

It is well known that iterative methods, utilizing the great speeds of modern-day computers are extensively used in large scale computations for solving equation (3.1-1) which arise from finite difference approximations to elliptic partial differential equations at present.

These methods consist of repeated application of a simple algorithm. They give the exact solution only as a limit of a sequence, even without consideration of round-off errors. In addition, they can be programmed to take advantage of the zero elements in A.

In an iterative method, one begins with an initial approximation and then successively modifies the approximation according to some rules. To be useful, the iteration must converge but it is not considered to be effective unless the convergence is rapid. We will discuss briefly the direct method for comparison, and later on, a greater detail on the iterative methods as these methods are our concern.

3.1.1 Problem formulation

Consider the two-point boundary value problem

$$-\frac{d^2U}{dx^2} + \rho U(x) = f(x), \qquad a \le x \le b \tag{3.1.1-1}$$

subject to the boundary condition

$$U(a) = \alpha, \; U(b) = \beta \tag{3.1.1-2}$$

where a and b denote the boundaries along the real axis $x > 0$. Here, we assume α, β and ρ are given constants with $\rho \ge 0$ and $f(x)$ is a real continuous function on $a \le x \le b$.

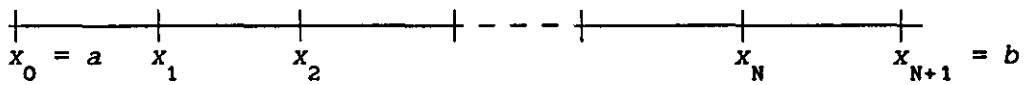
For simplicity, we place a uniform mesh of size,

$$h = (b-a)/(N+1) \tag{3.1.1-3}$$

on the interval $a \le x \le b$, and we denote the mesh points of the discrete problem by

$$\begin{aligned} x_i &= a + i(b-a)/(N+1) \\ &= a + ih, \qquad 0 \le i \le N+1 \end{aligned} \tag{3.1.1-4}$$

as illustrated



The well known method for deriving finite difference approximations to problem (3.1.1-1) is based on finite Taylor's series expansion of the solution $U(x)$ to problem (3.1.1-1).

Let us assume that the solution $U(x)$ of problem (3.1.1-1) with the boundary condition (3.1.1-2) is of class C^4 in $a \le x \le b$, i.e., d^4U/dx^4 exists and is continuous in the interval $[a,b]$. Now, denoting $U(x_i)$ by u_i, then the finite Taylor expansion for $u_{i\pm1}$ is given by

$$u_{i\pm1} = u_i \pm h\left(\frac{dU}{dx}\right)_i + \frac{h^2}{2!}\left(\frac{d^2U}{dx^2}\right)_i \pm \frac{h^3}{3!}\left(\frac{d^3U}{dx^3}\right)_i + \frac{h^4}{4!}\left(\frac{d^4U}{dx^4}\right)_{x = x_i + \theta_i^{\pm}h}$$

$$\text{where } 0 < |\theta_i^{\pm}| < 1. \qquad (3.1.1\text{-}5)$$

By using equation (3.1.1-5), an approximation to d^2U/dx^2 can be obtained as follows

$$-\left(\frac{d^2U}{dx^2}\right)_i = \frac{-u_{i-1} + 2u_i - u_{i+1}}{h^2} + \frac{h^2}{12}\left(\frac{d^4U}{dx^4}\right)_{x = x_i + \theta_i h}, \quad 0 \le |\theta_i| < 1. \qquad (3.1.1\text{-}6)$$

By substituting (3.1.1-6) in (3.1.1-1) gives

$$\frac{-u_{i-1} + 2u_i - u_{i+1}}{h^2} + \rho u_i = f_i - \frac{h^2}{12}M, \qquad 1 \le i \le N \qquad (3.1.1\text{-}7)$$

where $M = \dfrac{d^4u}{dx^4}_i$.

If we take h small, then the term $\frac{h^2}{12}M$ can be neglected. Thus, the equation becomes

$$\frac{-u_{i-1} + 2u_i - u_{i+1}}{h^2} + \rho u_i = f_i \qquad (3.1.1\text{-}8)$$

or simplified to

$$-u_{i-1} + 2gu_i - u_{i+1} = h^2 f_i, \qquad 1 \le i \le N \qquad (3.1.1\text{-}9)$$

where $g = 1 + \frac{1}{2}\rho h^2$, $u_0 = \alpha$ and $u_{N+1} = \beta$.

Equation (3.1.1-9) can be written in the matrix form $A\mathbf{u} = \mathbf{b}$, where

$$A = \begin{bmatrix} 2g & -1 & & & \\ -1 & 2g & -1 & & 0 \\ & \ddots & \ddots & \ddots & \\ & 0 & -1 & 2g & -1 \\ & & & -1 & 2g \end{bmatrix} \qquad (3.1.1\text{-}10)$$

$$\mathbf{u} = [u_1, u_2, \ldots, u_{N-1}, u_N]^T,$$

and $$\mathbf{b} = [\alpha+h^2f_1, h^2f_2, \ldots, h^2f_{N-1}, \beta+h^2f_N]^T.$$

The solution vector $\mathbf{u}$ of the system of linear equations is the discrete approximation to the exact solution $\mathbf{u}(x)$. The difference between this discrete approximation u_i and $U(x_i)$ can be made arbitrarily small for all $0 < i < N+1$ by choosing the mesh spacing h sufficiently small.

3.1.2 The Gaussian elimination (direct) method

This method, when applied to solve equation (3.1-1), is very efficient and stable against the growth of rounding error because of the positive definiteness property of A. For this problem, i.e., with the matrix A in (3.1.1-10), we may express the method as

$$b = 2g, \quad w_1 = -1/b;$$

$$w_i = -1/(b + w_{i-1}); \qquad 2 \le i \le N-1.$$

$$p_1 = h^2b_1/b; \quad p_i = (h^2b_i + p_{i-1})/(b + w_{i-1}); \qquad 2 \le i \le N.$$

$$z_N = p_N; \quad z_i = p_i - w_ip_{i+1}; \qquad 1 \le i \le N-1.$$

However, for problems with two or more dimensions, i.e., partial differential equations, it is difficult to apply the direct methods and the application of iterative methods poses lesser problems.

3.1.3 The Jacobi iterative method

The simplest of the iterative methods is that attributed to Jacobi, sometimes known as the method of *Simultaneous Displacements*. It is not widely used in practice, but its theoretical interest may provide a convenient starting point for our next discussion.

Thus, solving $A\mathbf{u} = \mathbf{b}$ will give us a reasonable approximation to the solution of the two-point boundary value problem (3.1.1-1).

The matrix A of (3.1.1-9) can be expressed as

$$A = 2g[I - B] \tag{3.1.3-1}$$

where B is NxN real symmetric matrix given explicitly by

$$B = \frac{1}{2g}\begin{bmatrix} 0 & 1 & & & \\ 1 & 0 & 1 & & 0 \\ & \ddots & \ddots & \ddots & \\ & 0 & 1 & 0 & 1 \\ & & & 1 & 0 \end{bmatrix}. \tag{3.1.3-2}$$

It is easy to verify that the vector $\mathbf{x}^{(j)}$, is the eigenvector of B for each j where its ith component $x_i^{(j)}$ defined as $sin\ (ij\pi h)$, $1 \le i \le N$ and the corresponding eigenvalue μ_j is given by

$$\mu_j = \frac{2\ cos\ (j\pi h)}{2g}, \quad 1 \le j \le N. \tag{3.1.3-3}$$

Clearly, the spectral radius $\rho(B) = max\ |\mu_j|$, $1 \le j \le N$ is less than one and thus, from (3.1.3-1), A is positive definite. Thus, since $\rho(B) < 1$, then A^{-1} can be expressed in the form of the convergent matrix infinite series

$$A^{-1} = \frac{1}{2g}(I-B)^{-1}$$

$$= \frac{1}{2g}(I + B + B^2 + \ldots\ ). \tag{3.1.3-4}$$

Since all entries of B from (3.1.3-2) are non-negative, then so are all the powers of B in (3.1.3-4). Hence A^{-1} only has the non-negative real entries.

Now, from (3.1.3-2), let us form

$$\mathbf{u} = B\mathbf{u} + \mathbf{c} \tag{3.1.3-5}$$

where $\mathbf{c} = \mathbf{b}/2g$ and consider the following iterative method, called *the Jacobi iterative method*

$$\mathbf{u}^{(k+1)} = B\mathbf{u}^{(k)} + \mathbf{c}, \qquad k > 0 \tag{3.1.3-6}$$

or in point form

$$u_i^{(k+1)} = u_i^{(k)} - \frac{1}{2g}\left[2gu_i^{(k)} - u_{i-1}^{(k)} - u_{i+1}^{(k)} - h^2c_i\right], \qquad 1 \le i \le N \tag{3.1.3-7}$$

where $\mathbf{u}^{(0)}$ is the initial guess of the unique solution of (3.1.3-6). We now seek to prove the convergence of the Jacobi iterative method.

We define the error $\mathbf{e}^{(k)} = \mathbf{u}^{(k)} - \mathbf{u}$ at each iteration, where $\mathbf{u}$ is the exact solution of (3.1.1-1), to obtain $\mathbf{e}^{(k+1)} = B\mathbf{e}^{(k)}$. Now, take the difference between (3.1.3-6) and (3.1.3-5), we have

$$\mathbf{u}^{(k+1)} - \mathbf{u} = B(\mathbf{u}^{(k)} - \mathbf{u})$$

$$\mathbf{e}^{(k+1)} = B\mathbf{e}^{(k)}. \tag{3.1.3-8}$$

Equation (3.1.3-8) can be deduced to

$$\mathbf{e}^{(k)} = B^k\mathbf{e}^{(0)}, \qquad k > 0. \tag{3.1.3-9}$$

The eigenvectors $\{\mathbf{x}^j\}_{k+1}^N$ of the real symmetric matrix B can be shown to form an orthonormal basis for the associated N dimension vector space. Thus, there exists some constants d_j such that

$$\mathbf{e}^{(0)} = \sum_{j=1}^{N} d_j\mathbf{x}^{(j)} \tag{3.1.3-10}$$

where $B\mathbf{x}^{(j)} = \mu_j\mathbf{x}^{(j)}$, and from (3.1.3-9) if follows that

$$\mathbf{e}^{(k)} = \sum_{j=1}^{N} (\mu_j)^k d_j\mathbf{x}^{(j)}, \qquad N \ge 0. \tag{3.1.3-11}$$

Since $|\mu_j| < 1$, $1 < j < N$, $\mathbf{e}^{(k)}$ clearly tends to the zero vector for any initial $\mathbf{e}^{(0)}$ and this method converges for any initial guess $\mathbf{u}^{(0)}$. In addition, each component of the error $\mathbf{e}^{(k)}$ is reduced at every iteration by the factor $\mathcal{S}(B)$, i.e.,

$$\begin{aligned}\mathcal{S}(B) &= max\ |\mu_j| \\ &= max\ \left| \frac{2\ cos\ (j\pi h)}{2g} \right| \\ &= max\ |\ 1 - \tfrac{1}{2}(\pi^2 + \rho)h^2 + O(h^4)\ | \end{aligned} \qquad (3.1.3\text{-}12)$$

Although the method (3.1.3-7) does converge, it can be extremely very slowly convergent for small h.

For our model problem, we now write the algorithm for the Jacobi iterative method as follows:

Algorithm 3.1.3-1: The Jacobi iterative method.

Set $u_0^{(k)} = 0$, $u_{N+1}^{(k)} = 0$.

Step 1. To compute $\mathbf{u}^{(k+1)}$.

for $i = 1$ to N, compute

$$u_i^{(k+1)} = (b_i + u_{i-1}^{(k)} + u_{i+1}^{(k)})/2g.$$

Step 2. Repeat *Step* 1 until convergence is achieved.

3.1.4 The Gauss-Seidel iterative method

A simple modification of the Jacobi iterative method leads to the *Gauss-Seidel iterative method*, is a better practical proposition. It is also known as *the method of Successive Displacement*. Instead of waiting to use the improved value at the end of the current iteration which is to be substituted into the next iteration, we now use the updated value as soon as they become available.

Now, let us express the matrix B of (3.1.3-2) as the sum of $L + U$, where L and U are strictly the lower and upper triangular matrices respectively. Then from (3.1.3-5), we have

$$\mathbf{u} = (L+U)\mathbf{u} + \mathbf{c}. \qquad (3.1.4\text{-}1)$$

Since $U = L^T$, equation (3.1.4-1) then becomes

$$\mathbf{u} = (L+L^T)\mathbf{u} + \mathbf{c} \qquad (3.1.4\text{-}2)$$

and after some rearrangement, we will have

$$(I-L)\mathbf{u} = L^T\mathbf{u} + \mathbf{c}. \qquad (3.1.4\text{-}3)$$

Now, since L is strictly lower triangular, then $(I-L)$ is non-singular matrix. Thus, we can multiply (3.1.4-3) on both sides by $(I-L)^{-1}$ which will give the form

$$\mathbf{u} = (I-L)^{-1}L^T\mathbf{u} + (I-L)^{-1}\mathbf{c}. \qquad (3.1.4\text{-}4)$$

Now, we define the Gauss-Seidel method as,

$$\mathbf{u}^{(k+1)} = (I-L)^{-1}L^T\mathbf{u}^{(k)} + (I-L)^{-1}\mathbf{c}, \qquad k \geq 0. \qquad (3.1.4\text{-}5)$$

It appears that the method (3.1.4-5) is implicit. From the definition of matrix B in (3.1.3-2), however, the method can be written in point form as

$$u_i^{(k+1)} = u_i^{(k)} - \frac{1}{2g}\left[2gu_i^{(k)} - u_{i-1}^{(k+1)} - u_{i+1}^{(k)} - h^2c_i\right], \qquad 1 \leq i \leq N. \qquad (3.1.4\text{-}6)$$

It can be shown that (3.1.4-6) is an explicit method. Starting with $k = 1$, and taking an initial guess $\mathbf{u}_0 = \alpha$, as from equation (3.1.1-1), the new value $u_1^{(k+1)}$ in (3.1.4-6) is solved in terms of the old values of $u_1^{(k)}$ and $u_2^{(k)}$. Similarly, the new $u_2^{(k+1)}$ can be obtained in terms of the recent value of $u_1^{(k+1)}$ and the old value of $u_3^{(k)}$. In general, for the ith component, i.e., the new $u_i^{(k+1)}$ is computed from the latest value of $u_{i-1}^{(k+1)}$ and the old value of $u_{i+2}^{(k)}$.

We now seek the eigenvalues λ of the Gauss-Seidel iteration matrix,

$$\mathcal{L}_{GS} = (I-L)^{-1}L^T \tag{3.1.4-7}$$

that is the roots of $det(\lambda I - \mathcal{L}_{GS}) = 0$. Since $det(I-L) = 1$, then

$$det(I-L)det(\lambda I - \mathcal{L}_{GS}) = det[\lambda I - (\lambda L + L^T)]$$
$$= 0. \tag{3.1.4-8}$$

In the same manner as before, it can be shown that the vector $w^{(j)}$, with the ith component $w_i^{(j)}$, defined as $\tau^{1/2}sin(ij\pi h)$ is the eigenvector of $\tau L + L^T$ for any τ, with the corresponding eigenvalue $\tau^{1/2}\mu_j$. Thus, with $\tau = \lambda$, there is a natural pairing from (3.1.4-8) of the eigenvalues λ of $\mathcal{L}_{GS}$ with the eigenvalues μ of B by the relationship

$$\lambda = \mu^2 \tag{3.1.4-9}$$

The relation (3.1.4-9) can be shown as follows:

$$det(\mu I-B) = 0 \tag{3.1.4-10}$$

For our model problem, the matrix B in (3.1.3-2) gives

$$det \begin{vmatrix} \mu & 1 & & & \\ 1 & \mu & 1 & 0 & \\ & \ddots & \ddots & \ddots & \\ & 0 & 1 & \mu & 1 \\ & & & 1 & \mu \end{vmatrix} = 0 \tag{3.1.4-11}$$

Now, for the Gauss-Seidel method, i.e., from (3.1.4-8), then

$$det \begin{vmatrix} \lambda & -1 & & & \\ -\lambda & \lambda & -1 & 0 & \\ & \ddots & \ddots & \ddots & \\ & 0 & -\lambda & \lambda & -1 \\ & & & -\lambda & \lambda \end{vmatrix} = 0 \tag{3.1.4-12}$$

Now, for any non-singular matrix Q,

$$det(Q^{-1}\mathcal{L}_{GS}Q) = det(Q^{-1})det(\mathcal{L}_{GS})det(Q) \tag{3.1.4-13}$$

If we choose the appropriate Q, then from (3.1.4-12) $det(Q^{-1}\mathcal{L}_{GS}Q)$ could give us a symmetric matrix

$$det \begin{vmatrix} \lambda & -\lambda^{1/2} & & & \\ -\lambda^{1/2} & \lambda & -\lambda^{1/2} & & 0 \\ & \ddots & \ddots & \ddots & \\ & & -\lambda^{1/2} & \lambda & -\lambda^{1/2} \\ & 0 & & -\lambda^{1/2} & \lambda \end{vmatrix} = 0 \qquad (3.1.4\text{-}14)$$

Since $\lambda \neq 0$, then we can divide every element by $-\lambda^{1/2}$. Thus, the equation (3.1.4-14) becomes

$$det \begin{vmatrix} -\lambda^{1/2} & 1 & & & \\ 1 & -\lambda^{1/2} & 1 & & 0 \\ & \ddots & \ddots & \ddots & \\ & & 1 & -\lambda^{1/2} & 1 \\ & 0 & & 1 & -\lambda^{1/2} \end{vmatrix} = 0 \qquad (3.1.4\text{-}15)$$

Now, from (3.1.4-11) and (3.1.4-15), it is clear that

$$-\lambda^{1/2} = \mu \qquad (3.1.4\text{-}16)$$

which on squaring it, yields the identical form of (3.1.4-9).

Thus, from this relation, it can be concluded that the Gauss-Seidel iterative method does converge and its rate of convergence is twice as fast as that of the Jacobi iterative method.

For the model problem, the algorithm for the Gauss-Seidel iterative method may be written as follows:

Algorithm 3.1.4-1: The Gauss-Seidel (G-S) iterative method.

Set $u_0^{(k)} = 0$, $u_{N+1}^{(k)} = 0$.

Step 1. To compute $\mathbf{u}^{(k+1)}$.

for $i = 1$ to N, compute

$$u_i^{(k+1)} = (b_i + u_{i-1}^{(k+1)} + u_{i+1}^{(k)})/2g.$$

Step 2. Repeat *Step* 1 until convergence is achieved.

3.1.5 The Successive Overrelaxation (SOR) method

A further simple modification of our improved Gauss-Seidel iterative method leads to a method which is more powerful, called *the Successive Overrelaxation method* (SOR). It is also known as the *extrapolated Gauss Seidel method.*

If we multiply both sides of equation (3.1.4-2) by the real parameter ω, the *relaxation factor* and then add u to both sides of the resulting equation, then we have the new equation

$$\mathbf{u} + \mathbf{u}\omega = \mathbf{u} + \omega[(L + L^T)\mathbf{u} + \mathbf{c}] \qquad (3.1.5\text{-}1)$$

and after some rearrangement, (3.1.5-1) then becomes

$$(I-\omega L)\mathbf{u} = \{(1-\omega)I + \omega L^T\}\mathbf{u} + \omega\mathbf{c}. \qquad (3.1.5\text{-}2)$$

The matrix $(I-\omega L)$ is non-singular for any choice of ω, since L is a strictly lower triangular matrix. Thus, we can multiply both sides of equation (3.1.5-2) by $(I-\omega L)^{-1}$, which will give

$$\mathbf{u} = (I-\omega L)^{-1}[(1-\omega)I + \omega L^T]\mathbf{u} + \omega(I-\omega L)^{-1}\mathbf{c}. \qquad (3.1.5\text{-}3)$$

Now, the SOR method can be defined as

$$\mathbf{u}^{(k+1)} = (I-\omega L)^{-1}[(1-\omega)I + \omega L^T]\mathbf{u}^{(k)} + \omega(I-\omega L)^{-1}\mathbf{c}, \quad k \geq 0. \qquad (3.1.5\text{-}4)$$

Again, it appears that the method (3.1.5-4) is an implicit method. From (3.1.3-2), however, the method (3.1.5-4) may be rewritten equivalently in point form as

$$u_i^{(k+1)} = u_i^{(k)} - \frac{\omega}{2g}\left[2gu_i^{(k)} - u_{i-1}^{(k+1)} - u_{i+1}^{(k)} - h^2c_i\right], \quad 1 \leq i \leq N. \qquad (3.1.5\text{-}5)$$

It can be verified that the method (3.1.5-5) is an explicit iterative method as we have shown for the Gauss-Seidel method in the previous section.

In analogy with the Gauss-Seidel method, we simply seek the eigenvalues λ of the matrix

$$\mathcal{L}_\omega = (I-\omega L)^{-1}[(1-\omega)I + \omega L^T] \qquad (3.1.5\text{-}6)$$

i.e., the roots of $det(\lambda I - \mathcal{L}_\omega) = 0$. Since $det(I-\omega L) = 1$, then

$$det(I-\omega L)det(\lambda I-\mathcal{L}_\omega) = det[(\lambda+\omega-1)I - \omega(\lambda L + L^T)]$$

$$= 0. \qquad (3.1.5\text{-}7)$$

For our model problem, we have

$$det \begin{vmatrix} \lambda+\omega-1 & -\omega & & \\ -\omega\lambda & \lambda+\omega-1 & -\omega & 0 \\ & -\omega\lambda & \lambda+\omega-1 & -\omega \\ 0 & & -\omega\lambda & \lambda+\omega-1 \end{vmatrix} = 0. \qquad (3.1.5\text{-}8)$$

Having performed the same manipulation as for (3.1.4-12), then using (3.1.5-8), we will have the symmetric matrix

$$det \begin{vmatrix} \frac{\lambda+\omega-1}{\omega} & -\lambda^{1/2} & & \\ -\lambda^{1/2} & \frac{\lambda+\omega-1}{\omega} & -\lambda^{1/2} & 0 \\ & -\lambda^{1/2} & \frac{\lambda+\omega-1}{\omega} & -\lambda^{1/2} \\ 0 & & -\lambda^{1/2} & \frac{\lambda+\omega-1}{\omega} \end{vmatrix} = 0. \qquad (3.1.5\text{-}9)$$

Since $\lambda \neq 0$, then we can divide every element in (3.1.5-9) by $-\lambda^{1/2}$ gives

$$det \begin{vmatrix} D & 1 & & \\ 1 & D & 1 & 0 \\ & 1 & D & 1 \\ 0 & & 1 & D \end{vmatrix} = 0 \qquad (3.1.5\text{-}10)$$

where $D = -\frac{\lambda+\omega-1}{\omega\lambda^{1/2}}$.

Thus, from (3.1.4-11) and (3.1.5-10) we can conclude that

$$-\frac{\lambda+\omega-1}{\omega\lambda^{1/2}} = \mu \qquad (3.1.5\text{-}11)$$

and having squared it, we have

$$(\lambda+\omega-1)^2 = \lambda\omega^2\mu^2. \qquad (3.1.5\text{-}12)$$

It is obvious that, if we take $\omega = 1$, then the relation (3.1.5-12) will yield the form of (3.1.4-9). Clearly, the relation (3.1.5-12) shows that the SOR method does converge and the rate of convergence depends entirely on ω.

By putting $\omega = 1$, the rate of convergence for this method is the same as the Gauss-Seidel method. However, we can verify that there is only one value of ω which will give the best possible rate of convergence, i.e., that which minimizes the spectral radius of $\mathcal{L}_\omega$.

The determination of a suitable value for the relaxation factor ω of the SOR method is very important, and particularly the optimum value of ω, denoted by ω_b, which minimizes the spectral radius of the SOR iteration matrix and thereby maximize the rate of convergence of the method. For an arbitrary set of linear equations, there is no formula to determine ω_b, though a simple but time consuming procedure in order to estimate ω_b is to run the problem on the computer for a range value of ω that will give some idea of the values which shows the best rate of convergence. But, indeed it is well known that in Young [1971]

$$min\ \rho(\mathcal{L}_\omega) = \rho(\mathcal{L}_{\omega_b}) = \omega_b - 1$$

where

$$\omega_b = \frac{2}{1 + \sqrt{1 - \rho^2(B)}}. \qquad (3.1.5\text{-}13)$$

If μ is the spectral radius of B, then

$$\omega_b = \frac{2}{1 + \sqrt{1 - \mu^2}}. \qquad (3.1.5\text{-}14)$$

Thus, the spectral radius of $\mathcal{L}_{\omega_b}$ is given by

$$\begin{aligned}\rho(\mathcal{L}_{\omega_b}) &= \omega_b - 1 \\ &= \frac{2}{1 + \sqrt{1 - \mu^2}} - 1 \\ &= \frac{1 - \sqrt{1 - \mu^2}}{1 + \sqrt{1 - \mu^2}}.\end{aligned} \tag{3.1.5-15}$$

Using the result of (3.1.3-12), we see that

$$\rho(\mathcal{L}_{\omega_b}) = 1 - 2\,(\sqrt{\pi^2 + \rho})h + O(h^2), \qquad h > 0. \tag{3.1.5-16}$$

Now, we state two theorems about the range of ω that guarantee the convergent of the SOR method. We commence with the theorem of (Kahan [1958]).

Theorem 3.1.5-1 (Kahan [1958])

If $\mathcal{L}_\omega$ is the SOR iteration matrix, then,

$$\rho(\mathcal{L}_\omega) \geq |\omega - 1|, \tag{3.1.5-17}$$

for all real ω. Moreover, if the SOR method converges, then,

$$0 < \omega < 2. \tag{3.1.5-18}$$

Proof:

If λ_i are the eigenvalues of $\mathcal{L}_\omega$, then from (2.3-5) we have,

$$det(\mathcal{L}_\omega) = \prod_{i=1}^{N} \lambda_i \tag{3.1.5-19}$$

Hence, by (3.1.5-6) we have

$$\begin{aligned}det(\mathcal{L}_\omega) &= det[(I-\omega L)^{-1}((1-\omega)I + \omega L^T)] \\ &= det[(I-\omega L)^{-1}]det[(1-\omega)I + \omega L^T)].\end{aligned}$$

But $(I-\omega L)$ is a lower triangular matrix with diagonal elements equal to one and $[(1-\omega)I + \omega L^T]$ is an upper triangular matrix with diagonal elements equal to $1-\omega$. Hence,

$$det(\mathcal{L}_\omega) = 1.(1-\omega)^N. \qquad (3.1.5\text{-}20)$$

Then, from (3.1.5-19) and (3.1.5-20) we have,

$$\prod_{i=1}^{N} \lambda_i = (1-\omega)^N. \qquad (3.1.5\text{-}21)$$

But,

$$\rho(\mathcal{L}_\omega) = \max_i |\lambda_i|. \qquad (3.1.5\text{-}22)$$

and

$$|1-\omega|^N \leq \max_i |\lambda_i|^N,$$

which implies $\rho(\mathcal{L}_\omega) \geq |(1-\omega)|$.

Now, if the SOR method converges, then

$$\rho(\mathcal{L}_\omega) < 1,$$

which implies

$$|1 - \omega| < 1, \text{ i.e., } 0 < \omega < 2.$$

Theorem 3.1.5-2

Let A be a symmetric matrix with positive diagonal elements. Then, the SOR method converges if and only if A is positive definite and $0 < \omega < 2$.

The proof of this theorem can be found in Young [1971].

Let us summarize the principal findings of this section, i.e., the relationships between the spectral radii of the three iteration matrices as shown in *Table* 3.1.5-1, where A is a tridiagonal matrix.

Method	*Iteration Matrix*	*Spectral Radius*
Jacobi	B	μ
Gauss-Seidel	$(I-L)^{-1}L^T$	μ^2
SOR with $\omega = \omega_b$	$(I-\omega L)^{-1}[(1-\omega)I+\omega L^T]$	$\dfrac{1 - \sqrt{1 - \mu^2}}{1 + \sqrt{1 - \mu^2}}$

Table 3.1.5-1: Comparison of the Spectral Radius

From *Table* (3.1.5-1), we can verify that the relation

$$\mathcal{S}(\mathcal{L}_{\omega_b}) < \mathcal{S}(\mathcal{L}_{GS}) < \mathcal{S}(B) \qquad (3.1.5\text{-}23)$$

holds for any value of μ, $\mu < 1$.

Thus, so far, we may conclude that the SOR method is superior than the Gauss-Seidel method and far better than the Jacobi method. It is also obvious that having found the $\mathcal{S}(B)$, we may determine $\mathcal{S}(\mathcal{L}_{GS})$ or $\mathcal{S}(\mathcal{L}_{\omega_b})$ if we wish to solve a problem via the Gauss-Seidel or SOR method respectively.

For the model problem, the algorithm for the SOR method may be presented as follows.

Algorithm 3.1.5-1: The SOR method.

Set $u_0^{(k)} = 0$, $u_{N+1}^{(k)} = 0$.

Step 1. To compute $\mathbf{u}^{(k+1)}$.

for $i = 1$ to N, compute

$$u_i^{(k+1)} = \omega[(b_i + u_{i-1}^{(k+1)} + u_{i+1}^{(k)})/2g] + (1-\omega)u_i^{(k)}.$$

Step 2. Repeat *Step* 1 until convergence is achieved.

3.1.6 The Alternating Group Explicit (AGE) iterative method

We now consider a class of methods for solving equation (3.1-1) which is based on the *'splitting'* of the matrix A into two submatrices

$$A = G_1 + G_2. \qquad (3.1.6\text{-}1)$$

We shall be concerned here with the situation where G_1 and G_2 are either small (2x2) block systems or can be made so by a suitable permutation on their rows and corresponding columns. This procedure is *'convenient'* in the sense that the work required is much less than would be required to solve the original system (3.1-1) directly.

For our model problem, we have

$$G_1 = \begin{bmatrix} g & -1 & & & & & \\ -1 & g & & & & & \\ & & g & -1 & & & \\ & & -1 & g & & & \\ & & & & \ddots & & \\ & & & & & g & -1 \\ & & & & & -1 & g \end{bmatrix}, \qquad G_2 = \begin{bmatrix} g & & & & & & \\ & g & -1 & & & & \\ & -1 & g & & & & \\ & & & \ddots & & & \\ & & & & g & -1 & \\ & & & & -1 & g & \\ & & & & & & g \end{bmatrix}$$

if N is even, and

$$G_1 = \begin{bmatrix} g & -1 & & & & \\ -1 & g & & & & \\ & & \ddots & & & \\ & & & g & -1 & \\ & & & -1 & g & \\ & & & & & g \end{bmatrix}, \qquad G_2 = \begin{bmatrix} g & & & & & \\ & g & -1 & & & \\ & -1 & g & & & \\ & & & \ddots & & \\ & & & & g & -1 \\ & & & & -1 & g \end{bmatrix}$$

if N is odd.

Evidently, G_1 and G_2 satisfy the following conditions.

a) $(rI + G_1)$ and $(rI + G_2)$ are non-singular for any $r > 0$, where r is called the *acceleration parameter*.

b) For any vectors **c** and **d** and for any $r > 0$, it is practical to solve the systems

$$(rI + G_1)\mathbf{x} = \mathbf{c}, \qquad (rI + G_2)\mathbf{y} = \mathbf{d} \tag{3.1.6-2}$$

in explicit form since they consist of only the (2x2) subsystems.

Thus, (3.1-1) becomes

$$(G_1 + G_2)\mathbf{u} = \mathbf{b} \tag{3.1.6-3}$$

and by applying the AGE method, $\mathbf{u}^{(k+1)}$ can be determined in two sweeps, i.e.,

$$(rI + G_1)\mathbf{u}^{(k+1/2)} = \mathbf{b} + (rI - G_2)\mathbf{u}^{(k)} \tag{3.1.6-4}$$

$$(rI + G_2)\mathbf{u}^{(k+1)} = \mathbf{b} + (rI - G_1)\mathbf{u}^{(k+1/2)}. \tag{3.1.6-5}$$

Since $rI + G_1$ and $rI + G_2$ are non-singular, then the respective inverse

does exist. Thus, we can write the AGE method in the explicit form

$$\mathbf{u}^{(k+1/2)} = (rI + G_1)^{-1}[\mathbf{b} + (rI - G_2)\mathbf{u}^{(k)}] \qquad (3.1.6\text{-}6)$$

$$\mathbf{u}^{(k+1)} = (rI + G_2)^{-1}[\mathbf{b} + (rI - G_1)\mathbf{u}^{(k+1/2)}]. \qquad (3.1.6\text{-}7)$$

We now seek to analyse the convergence properties of the AGE method. If the true solution of (3.1-1) is $\mathbf{u}$, then

$$(G_1 + G_2)\mathbf{u} = \mathbf{b} \qquad (3.1.6\text{-}8)$$

and

$$(rI + G_1)\mathbf{u} = \mathbf{b} + (rI - G_2)\mathbf{u}. \qquad (3.1.6\text{-}9)$$

Therefore, from (3.1.6-4), we have

$$(rI + G_1)\mathbf{e}^{(k+1/2)} = -(rI - G_2)\mathbf{e}^{(k)} \qquad (3.1.6\text{-}10)$$

and similarly, from (3.1.6-5) it gives

$$(rI + G_2)\mathbf{e}^{(k+1)} = -(rI - G_1)\mathbf{e}^{(k+1/2)} \qquad (3.1.6\text{-}11)$$

and hence

$$\mathbf{e}^{(k+1)} = T_r\mathbf{e}^{(k)} \qquad (3.1.6\text{-}12)$$

where

$$T_r = (rI + G_2)^{-1}(rI - G_1)(rI + G_1)^{-1}(rI - G_2). \qquad (3.1.6\text{-}13)$$

We now prove the theorem 3.1.6-1.

Theorem 3.1.6-1.

If G_1 and G_2 are real positive definite matrices and if $r > 0$, then

$$\rho(T_r) < 1. \qquad (3.1.6\text{-}14)$$

Proof.

Evidently, T_r is similar to the matrix $\tilde{T}_r$, where

$$\tilde{T}_r = (rI + G_2)T_r(rI + G_2)^{-1}$$

$$= (rI - G_1)(rI + G_1)^{-1}(rI - G_2)(rI + G_2)^{-1} \qquad (3.1.6\text{-}15)$$

and

$$\|\tilde{T}_r\|_2 \le \|(rI - G_1)(rI + G_1)^{-1}\|_2 \|(rI - G_2)(rI + G_2)^{-1}\|_2 \qquad (3.1.6\text{-}16)$$

But since G_1 and G_2 are symmetric and since $(rI - G_1)$ commutes with $(rI + G_1)^{-1}$, we have

$$\|(rI - G_1)(rI + G_1)^{-1}\|_2 = \rho((rI - G_1)(rI + G_1)^{-1})$$

$$= \max_{\mu} \left|\frac{r - \mu}{r + \mu}\right| \qquad (3.1.6\text{-}17)$$

where μ ranges over all eigenvalues of G_1. But since G_1 is positive definite, its eigenvalues are positive. Therefore,

$$\|(rI - G_1)(rI + G_1)^{-1}\|_2 < 1. \qquad (3.1.6\text{-}18)$$

Similarly, $\|(rI - G_2)(rI + G_2)^{-1}\|_2 < 1$, and we have

$$\rho(T_r) = \rho(\tilde{T}_r) \le \| \tilde{T}_r \|_2 < 1. \qquad (3.1.6\text{-}19)$$

Hence, the convergence follows.

Even if G_1 and G_2 are not symmetric, convergence may still hold.

Theorem 3.1.6-2.

If there exists a real non-singular matrix P such that $\hat{G}_1 = P^{-1}G_1P$ *and* $\hat{G}_2 = P^{-1}G_2P$ *have positive eigenvalues and if* $r > 0$ *then* $\|(T_r)\|_2 < 1$.

Proof:

As in the proof of *Theorem* 3.1.6-1, by a similarity transformation, T_r is similar to $\tilde{T}_r$ given by (3.1.6-15); then $\tilde{T}_r$ in turn similar to

$$\hat{T}_r = P^{-1}\tilde{T}_rP$$

$$= (rI - \hat{G}_1)(rI + \hat{G}_1)^{-1}(rI - \hat{G}_2)(rI + \hat{G}_2)^{-1}. \qquad (3.1.6\text{-}20)$$

Since $\hat{G}_1$ and $\hat{G}_2$ have positive eigenvalues, then we can show that

$$\|(rI - \hat{G}_1)(rI + \hat{G}_1)^{-1}\|_2 < 1 \quad \text{and} \quad \|(rI - \hat{G}_2)(rI + \hat{G}_2)^{-1}\|_2 < 1.$$

Hence

$$\|(T_r)\|_2 \le \|(rI - \hat{G}_1)(rI + \hat{G}_1)^{-1}\|_2 \, \|(rI - \hat{G}_2)(rI + \hat{G}_2)^{-1}\|_2 < 1. \qquad (3.1.6\text{-}21)$$

Now when r is constant, i.e., the stationary case, the optimum r is given by

$$r = \sqrt{ab} \tag{3.1.6-22}$$

where a and b are the lower and upper bounds of the eigenvalues of G_1 and G_2 respectively. This can be shown as follows:

Let us now assume that G_1 and G_2 are real positive definite matrices and that the eigenvalues μ of G_1 and ν of G_2 lie in the ranges

$$0 < a \le \mu \le b, \qquad 0 < a \le \nu \le b. \tag{3.1.6-23}$$

Evidently, if $r > 0$ we have

$$\begin{aligned} \varphi(T_r) &\le \varphi((rI - G_1)(rI + G_1)^{-1})\,\varphi((rI - G_2)(rI + G_2)^{-1}) \\ &= \left(\max_{a\le\mu\le b}\left|\frac{r-\mu}{r+\mu}\right|\right)\left(\max_{a\le\nu\le b}\left|\frac{r-\nu}{r+\nu}\right|\right) \\ &= \left(\max_{a\le\mu\le b}\left|\frac{r-\mu}{r+\mu}\right|\right)^2 = \phi(a,b;r). \end{aligned} \tag{3.1.6-24}$$

Since $\frac{r-\gamma}{r+\gamma}$ is an decreasing function of γ, we have

$$\max_{a\le\gamma\le b}\left|\frac{r-\gamma}{r+\gamma}\right| = \max\left(\left|\frac{r-a}{r+a}\right|, \left|\frac{r-b}{r+b}\right|\right). \tag{3.1.6-25}$$

When $r = \sqrt{ab}$, then

$$\left|\frac{r-a}{r+a}\right| = \left|\frac{r-b}{r+b}\right| = \frac{\sqrt{b}-\sqrt{a}}{\sqrt{b}+\sqrt{a}}. \tag{3.1.6-26}$$

Moreover, if $0 < r < \sqrt{ab}$, we have

$$\left|\frac{r-b}{r+b}\right| - \frac{\sqrt{b}-\sqrt{a}}{\sqrt{b}+\sqrt{a}} = \frac{2\sqrt{b}(\sqrt{ab}-r)}{(b+r)(\sqrt{b}+\sqrt{a})} > 0, \tag{3.1.6-27}$$

and if $\sqrt{ab} < r$, then

$$\left|\frac{r-a}{r+a}\right| - \frac{\sqrt{b}-\sqrt{a}}{\sqrt{b}+\sqrt{a}} = \frac{2\sqrt{b}(r-\sqrt{ab})}{(r+a)(\sqrt{b}+\sqrt{a})} > 0. \tag{3.1.6-28}$$

Therefore $\phi(a,b;r)$ is minimized when $r = \sqrt{ab}$ and

$$\varphi(T_{\sqrt{ab}}) \le \phi(a,b;\sqrt{ab}) = \left(\frac{\sqrt{b}-\sqrt{a}}{\sqrt{b}+\sqrt{a}}\right)^2. \tag{3.1.6-29}$$

Thus, $r = \sqrt{ab}$ is optimum in the sense that the bound $\phi(a,b;r)$ for $\rho(T_r)$ is minimized.

For the model problem, it is clear that the values of a and b are given by the roots of the (2x2) determinantal equation,

$$det \begin{vmatrix} \lambda-g & -1 \\ -1 & \lambda-g \end{vmatrix} = 0 \qquad (3.1.6\text{-}30)$$

and are given by

$$a = g - 1 = \rho h^2/2, \quad b = g + 1 = 2 + \rho h^2/2. \qquad (3.1.6\text{-}31)$$

Thus the optimum parameter is

$$r = \sqrt{ab} = h\sqrt{\rho} \qquad (3.1.6\text{-}32)$$

and the spectral radius is

$$\rho(T_{\sqrt{ab}}) = \left(\frac{\sqrt{b} - \sqrt{a}}{\sqrt{b} + \sqrt{a}} \right)^2$$

$$= 1 - 4\left(\frac{a}{b} \right)^{1/2} \text{ for } b >> a.$$

Thus, the result $\rho(T_{\sqrt{ab}}) = 1 - 2h\sqrt{\rho}$. (3.1.6-33)

This shows that the convergence rate of the AGE method is of $O(h)$ and similar to that given by equation (3.1.5-16) for the SOR method.

Now for the model problem, we show the algorithmic procedure for the AGE method, for an even number of points N. Now, G_1 and G_2 are

$$G_1 = \begin{bmatrix} g & -1 & & & & \\ -1 & g & & & & \\ & & g & -1 & & \\ & & -1 & g & & \\ & & & & \ddots & \\ & & & & g & -1 \\ & & & & -1 & g \end{bmatrix}, \qquad G_2 = \begin{bmatrix} g & & & & & \\ & g & -1 & & & \\ & -1 & g & & & \\ & & & \ddots & & \\ & & & & g & -1 \\ & & & & -1 & g \end{bmatrix}.$$

Hence, by applying the AGE method, one can determine $u^{(k+1/2)}$ and $u^{(k+1)}$ successively from equations (3.1.6-6) and (3.1.6-7). Obviously, the (2x2) submatrices of $(rI + G_1)$ and $(rI + G_2)$ are of the form

$$C = \begin{bmatrix} \alpha & -1 \\ -1 & \alpha \end{bmatrix} \qquad (3.1.6\text{-}34)$$

and the inverse of C is given by

$$C^{-1} = d\begin{bmatrix} \alpha & 1 \\ 1 & \alpha \end{bmatrix}, \qquad (3.1.6\text{-}35)$$

where $\alpha = r + g$, and $d = 1/(\alpha^2 - 1)$.

Hence, the vector $\mathbf{u}^{(k+1)}$ can be determined from $\mathbf{u}^{(k)}$ in two steps. One first determines $\mathbf{u}^{(k+1/2)}$ as follows

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ \cdots \\ \cdots \\ u_{N-1} \\ u_N \end{bmatrix}^{(k+\frac{1}{2})} = d \begin{bmatrix} \alpha & 1 & & & & & & \\ 1 & \alpha & & & & & & \\ & & \alpha & 1 & & & & \\ & & 1 & \alpha & & & & \\ & & & & \ddots & & & \\ & & & & & & \alpha & 1 \\ & & & & & & 1 & \alpha \end{bmatrix} \times \left\{ \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ \cdots \\ \cdots \\ b_{N-1} \\ b_N \end{bmatrix} + \begin{bmatrix} \beta & & & & & & \\ & \beta & 1 & & & & \\ & 1 & \beta & & & & \\ & & & \ddots & & & \\ & & & & \beta & 1 & \\ & & & & 1 & \beta & \\ & & & & & & \beta \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \cdots \\ \cdots \\ u_{N-2} \\ u_{N-1} \\ u_N \end{bmatrix}^{(k)} \right\}$$

which simplifies to

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ \cdots \\ \cdots \\ u_{N-1} \\ u_N \end{bmatrix}^{(k+\frac{1}{2})} = d \begin{bmatrix} \alpha & 1 & & & & & & \\ 1 & \alpha & & & & & & \\ & & \alpha & 1 & & & & \\ & & 1 & \alpha & & & & \\ & & & & \ddots & & & \\ & & & & & & \alpha & 1 \\ & & & & & & 1 & \alpha \end{bmatrix} \begin{bmatrix} b_1 + \beta u_1 \\ b_2 + \beta u_2 + u_3 \\ b_3 + u_2 + \beta u_3 \\ b_4 + \beta u_4 + u_5 \\ \cdots \\ \cdots \\ b_{N-1} + u_{N-2} + \beta u_{N-1} \\ b_N + \beta u_N \end{bmatrix}^{(k)}$$

and by using the values of $\mathbf{u}^{(k+1/2)}$ one determines $\mathbf{u}^{(k+1)}$

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \cdots \\ \cdots \\ u_{N-2} \\ u_{N-1} \\ u_N \end{bmatrix}^{(k+1)} = \begin{bmatrix} 1/\alpha & & & & & \\ & \alpha d & d & & & \\ & d & \alpha d & & & \\ & & & \ddots & & \\ & & & & \alpha d & d \\ & & & & d & \alpha d \\ & & & & & 1/\alpha \end{bmatrix} \times \left\{ \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \cdots \\ \cdots \\ b_{N-2} \\ b_{N-1} \\ b_N \end{bmatrix} + \begin{bmatrix} \beta & 1 & & & & \\ 1 & \beta & & & & \\ & & \beta & 1 & & \\ & & 1 & \beta & & \\ & & & & \ddots & \\ & & & & & \beta \quad 1 \\ & & & & & 1 \quad \beta \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ \cdots \\ \cdots \\ u_{N-1} \\ u_N \end{bmatrix}^{(k+\frac{1}{2})} \right\}$$

which simplifies to

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \cdots \\ \cdots \\ u_{N-2} \\ u_{N-1} \\ u_N \end{bmatrix}^{(k+1)} = \begin{bmatrix} 1/\alpha & & & & & \\ & \alpha d & d & & & \\ & d & \alpha d & & & \\ & & & \ddots & & \\ & & & & \alpha d & d \\ & & & & d & \alpha d \\ & & & & & 1/\alpha \end{bmatrix} \begin{bmatrix} b_1 + \beta u_1 + u_2 \\ b_2 + u_1 + \beta u_2 \\ b_3 + \beta u_3 + u_4 \\ \cdots \\ \cdots \\ b_{N-2} + u_{N-3} + \beta u_{N-2} \\ b_{N-1} + \beta u_{N-1} + u_N \\ b_N + u_{N-1} + \beta u_N \end{bmatrix}^{(k+\frac{1}{2})}$$

where $\beta = r - g$.

Thus, one can write the algorithm to compute $\mathbf{u}^{(k+1/2)}$ and $\mathbf{u}^{(k+1)}$ as follows.

Algorithm 3.1.6-1: The AGE method the model problem (3.1.1-1).

Set $u_1^{(k)} = 0$, $i = 0, \ldots, N+1$ and $\alpha_1 = \alpha d$.

Step 1. To compute $\mathbf{u}^{(k+1/2)}$. Set $i = 1$.

while $i \leq N-1$, compute

$$r_1 = b_i + u_{i-1}^{(k)} + \beta u_i^{(k)}$$

$$r_2 = b_{i+1} + \beta u_{i+1}^{(k)} + u_{i+2}^{(k)}$$

$$u_i^{(k+1/2)} = \alpha_1 r_1 + r_2 d$$

$$u_{i+1}^{(k+1/2)} = r_1 d + \alpha_1 r_2$$

$$i = i + 2.$$

Step 2. To compute $u^{(k+1)}$. Set $i = 2$.

$$u_1^{(k+1)} = (b_1 + \beta u_1^{(k+1/2)} + u_2^{(k+1/2)})/\alpha$$

while $i \leq N-2$, compute

$$r_1 = b_1 + u_{i-1}^{(k+1/2)} + \beta u_i^{(k+1/2)}$$

$$r_2 = b_{i+1} + \beta u_{i+1}^{(k+1/2)} + u_{i+2}^{(k+1/2)}$$

$$u_i^{(k+1)} = \alpha_1 r_1 + r_2 d$$

$$u_{i+1}^{(k+1)} = r_1 d + \alpha_1 r_2$$

$$i = i + 2.$$

$$u_N^{(k+1)} = (b_N + u_{N-1}^{(k+1/2)} + \beta u_N^{(k+1/2)})/\alpha$$

Step 3. Repeat *Step* 1 and *Step* 2 until convergence is achieved.

3.1.7 The computational complexity

Now, concerning arithmetic calculations, we will estimate the amount of operations per iteration, required to solve the model problem via the Jacobi iterative method, Gauss-Seidel iterative method, SOR method and the AGE method.

1. The Jacobi iterative method.

   From the algorithm 3.1.3-1, for N mesh points we need

   $2N$ additions + N multiplications.

2. The Gauss-Seidel iterative method.

   From the algorithm 3.1.4-1, for N mesh points we need

   $2N$ additions + N multiplications.

3. The SOR method.

From the algorithm 3.1.5-1, for N mesh points we need

$4N$ additions + $3N$ multiplications.

4. The AGE method.

From the algorithm 3.1.6-1, we evaluate the operations in two sweeps.

(a) First sweep, i.e., to compute $\mathbf{u}^{k+1/2}$.

To compute r_1 and r_2, 4 additions and 2 multiplications are needed. To compute $u_1^{k+1/2}$ and $u_{i+1}^{k+1/2}$, we require 2 additions + 4 multiplications. So, the amount of operations is 6 additions + 6 multiplications. Since the operations are done in pairs, then the total operations required for N mesh points are

$3N$ additions + $3N$ multiplications.

(b) Second sweep, i.e., to compute $\mathbf{u}^{k+1}$.

For the first point, we need 2 additions + 2 multiplications. For the last point, we need 2 additions + 2 multiplications. The computation for the points in between is similar to the one in the first sweep. Thus, the amount of operations for every iteration is 6 additions + 6 multiplications. Since, this is also done in pairs and involves N-2 mesh points, then the total operation is

$3N$-2 additions + $3N$-2 multiplications.

Now, for the whole algorithm, we need

$6N$-2 additions + $6N$-2 multiplications.

To summarise, the table 3.1.7-1 shows the amount of operations required for the iterative methods that have been discussed.

Method	Multiplication	Addition	Overall
Jacobi	N	$2N$	$3N$
G-S	N	$2N$	$3N$
SOR	$3N$	$4N$	$7N$
AGE	$6N-2$	$6N-2$	$12N-4$

Table 3.1.7-1: The amount of operations per iteration

3.1.8 Experimental results

Numerical results presented here are for the iterative methods described earlier in sections 3.1.3 - 3.1.6 for solving the two-point boundary value problem subject to Dirichlet boundary conditions. Four problems have been considered with some variations on Problem 1. Problems 1 and 2 are linear whilst the 3rd and 4th problem are non-linear. Problem 4 introduces the first derivative for which after replacing it with finite difference approximation, yields an unsymmetric matrix.

Problem 1 - A Linear Problem

$$-U'' + \rho U = (\rho+1)(\sin x + \cos x), \qquad 0 \le x \le \frac{\pi}{2}$$

$$U(0) = 1, \; U(\frac{\pi}{2}) = 1.$$

The exact solution is $U(x) = \sin x + \cos x$.

The results for various ρ are tabulated as follows.

Method	Jacobi	G-S	SOR		AGE	
N	iter	iter	ω	iter	r	iter
10	194	106	1.60	25	0.50	19
20	595	329	1.80	51	0.28	38
40	1816	1027	1.86-1.87	83	0.15	77
80	5293	3104	1.93	163	0.08	156
160	13896	8703	1.96	323	0.043	312

Table 3.1.8-1: Problem 1 with $\rho = 0$

Method	Jacobi	G-S	SOR		AGE	
N	iter	iter	ω	iter	r	iter
10	160	88	1.53-1.54	22	0.50	18
20	492	272	1.72	42	0.29	34
40	1514	852	1.84	82	0.16	72
80	4469	2602	1.92-1.93	163	0.08	147
160	12013	7421	1.96	323	0.044-0.045	289

Table 3.1.8-2: Problem 1 with $\rho = 1$

Method	Jacobi	G-S	SOR		AGE	
N	iter	iter	ω	iter	r	iter
10	24	15	1.12-1.14	11	1.21-1.27	6
20	66	38	1.30-1.35	21	0.57-0.66	11
40	204	113	1.53-1.57	40	0.33-0.35	21
80	650	362	1.72-1.73	75	0.18	42
160	2020	1147	1.85	144	0.10	84

Table 3.1.8-3: Problem 1 with $\rho = 50$

Method	Jacobi	G-S	SOR		AGE	
N	iter	iter	ω	iter	r	iter
10	19	13	1.05-1.14	11	1.1 - 1.6	6
20	51	30	1.25-1.33	19	0.5 - 0.6	10
40	156	87	1.49-1.54	35	0.3 - 0.4	19
80	497	276	1.69-1.71	66	0.2	37
160	1560	881	1.83	125	0.05	73

Table 3.1.8-4: Problem 1 with $\rho = 70$

Method	Jacobi	G-S	SOR		AGE	
N	iter	iter	ω	iter	r	iter
10	16	11	1.08-1.09	9	1.56-1.74	5
20	39	24	1.22-1.27	16	0.58-0.95	9
40	117	66	1.43-1.50	30	0.27-0.46	16
80	372	206	1.64-1.67	56	0.12-0.16	31
160	1175	660	1.80-1.81	106	0.06	61

Table 3.1.8-5: Problem 1 with $\rho = 100$

Method	Jacobi	G-S	SOR		AGE	
N	iter	iter	ω	iter	r	iter
10	11	8	1.04-1.05	7	2.08-2.73	4
20	24	15	1.11-1.17	12	0.86-1.36	7
40	66	38	1.34-1.36	21	0.57	11
80	208	116	1.55-1.57	40	0.17-0.30	22
160	665	371	1.73-1.74	76	0.08-0.09	42

Table 3.1.8-6: Problem 1 with $\rho = 200$

Method	Jacobi	G-S	SOR		AGE	
N	iter	iter	ω	iter	r	iter
10	8	6	1.00-1.06	6	4.0 - 5.0	4
20	15	11	1.06-1.11	9	1.6 - 1.9	5
40	38	23	1.19-1.27	16	0.6 - 0.9	9
80	116	65	1.43-1.47	29	0.3 - 0.4	16
160	372	206	1.65-1.66	55	0.12-0.14	30

Table 3.1.8-7: Problem 1 with $\rho = 400$

Problem 2 - A Linear Problem

$$-U'' + U = 2\sin x - x + 2, \qquad 0 \le x \le \pi$$

$$U(0) = 2, \; U(\pi) = 2 - \pi.$$

The exact solution is $U(x) = \sin x - x + 2$.

The result is tabulated in *Table* 3.1.8-8.

Method	Jacobi	G-S	SOR		AGE	
N	iter	iter	ω	iter	r	iter
10	115	56	1.42	15	0.55-0.61	15
20	360	179	1.67	31	0.30-0.34	30
40	1114	576	1.78	54	0.17-0.19	60
80	3550	1817	1.88	102	0.10	118
160	10463	5436	1.94	204	0.05	240

Table 3.1.8-8: Problem 2

Problem 3 - A Mildly Non-linear Problem.

$$U'' = \frac{3}{2}U^2, \qquad 0 \le x \le 1$$

$$U(0) = 4, \; U(1) = 1.$$

The exact solution is $U(x) = 4/(1 + x)^2$.

The result is tabulated in *Table* 3.1.8-9.

Method	Jacobi	G-S	SOR		AGE	
N	iter	iter	ω	iter	*r*	iter
10	197	73	1.48-1.49	20	0.55-0.59	18
20	603	234	1.69	36	0.32	35
40	1859	751	1.883	70	0.17	72
80	5576	2365	1.91	139	0.09	143
160	15733	7070	1.95	275	0.05	286

Table 3.1.8-9: Problem 3

Problem 4 - A Linear Problem.

$$U'' + xU' - U = xe^x, \qquad 0 \le x \le 1$$

$$U(0) = 1, \; U(1) = 1 + e.$$

The exact solution is $U(x) = x + e^x$.

The finite difference approximation used for $U'(x)$ is $(u_{N+1} - u_{N-1})/2h$.

The result is tabulated in *Table* 3.1.8-10.

Method	Jacobi	G-S	SOR		AGE	
N	iter	iter	ω	iter	*r*	iter
10	188	99	1.547-1.548	22	1.000-1.053	19
20	588	306	1.731-1.737	43	0.573-0.595	38
40	1854	963	1.851-1.859	83	0.312-0.320	76
80	5684	2961	1.921-1.933	163	0.166-0.171	153
160	16318	8568	1.958-1.965	323	0.090	305

Table 3.1.8-10: Problem 4

The results show that the relation

$$\rho(\mathcal{L}_{\omega_b}) < \rho(\mathcal{L}_{GS}) < \rho(B)$$

holds for any $\mu < 1$, the eigenvalue of the Jacobi iterative methods, as μ depends on the mesh size, h, but our main interest is to look for the performance of the AGE method.

The results show the viability of the AGE method to solve the two point boundary value problem especially when the matrix A is diagonally dominant. In the case where the matrix A is strongly diagonally dominant, i.e., for large ρ, the AGE method is superior than the SOR method in terms of the number of iterations.

However, for small ρ, where the matrix A is less diagonally dominant, the AGE method compares well with SOR. However, there are a few cases in the numerical experiments where the AGE method needs a slightly larger number of iterations compared to the SOR method.

The simplicity of the method also needs to be considered. The AGE method is shown to be comprised of simple (2x2) block submatrix vector multiplications. This method can be expressed explicitly with the right hand side terms written in a more compact form as will be shown in later sections. This form, will ease the computational effort especially for parallel computation.

By considering these results and the computational complexity for both the AGE and SOR methods, we can state that the methods are comparable. With the possible extensions to solving the multi-dimensional problems, the AGE method can be shown to be more competitive than the SOR method.

We will now discuss how to determine the optimal r for the AGE method. From the theory given in Section 3.1.6, we have $r = \sqrt{ab}$, where a and b are the smallest and largest eigenvalues of the matrices G_1 and G_2, with $a = \rho h^2/2$, and $b = 2 + a$. This relation, however, appears to give good results only if the matrix A is strongly diagonally dominant. The numerical results that indicate these agreements can be analysed as follows.

A very good agreement can be seen when ρ = 400, i.e., for Problem 1 as shown in the *Table* 3.1.8-11 below.

	The AGE method		
N	*r*	$r = \sqrt{ab}$	*a*
10	4.0 - 5.0	4.979	4.078
20	1.6 - 1.9	1.869	1.119
40	0.6 - 0.9	0.821	0.294
80	0.3 - 0.4	0.395	0.075
160	0.12-0.14	0.196	0.019

Table 3.1.8-11: Problem 1, ρ = 400, r_{exp} .vs. r_{theory}

This agreement, however, deteriorates as ρ become smaller. *Tables* 3.1.8-12 and 3.1.8-13 which represent the results for Problem 1 for ρ = 50 and 1 respectively, demonstrate this fact.

	The AGE method		
N	*r*	$r = \sqrt{ab}$	*a*
10	1.21-1.27	1.310	0.510
20	0.57-0.66	0.547	0.140
40	0.33-0.35	0.273	0.037
80	0.18	0.137	0.009
160	0.10	0.069	0.002

Table 3.1.8-12: Problem 1, ρ = 50, r_{exp}.vs. r_{theory}

	The AGE method		
N	*r*	$r = \sqrt{ab}$	*a*
10	0.50	0.143	0.010
20	0.29	0.075	0.003
40	0.16	0.038	0.001
80	0.08	0.019	0.000
160	0.044-0.045	0.010	0.000

Table 3.1.8-13: Problem 1, ρ = 1, r_{exp}.vs. r_{theory}

Based on these results, when $\rho \geq 50$, we may use $r = \sqrt{ab}$ as an estimation for the experimental value of r. For $0 \leq \rho < 50$, we may expect that the experimental value of $r > \sqrt{ab}$ or falls within the the aritho-geometric mean interval $[\sqrt{ab}, (a+b)/2]$, (Wachspress [1968]).

Further numerical experiments were completed on a variety of more difficult problems in order to ascertain the validity of this comparison between the AGE and SOR methods and the determination of optimal r. Again it was observed that in these examples the AGE matrices were only slightly diagonal dominant and therefore the AGE and SOR methods were comparable.

Also, the optimal r occurred in the given interval $[\sqrt{ab},\ (a+b)/2]$. However, for the problem 3 and 4, the theoretical determination of this interval and the bounds a and b are more complex.

3.2 The solution with different boundary conditions (b.c.)

There are several types of boundary conditions which arise frequently in the description of physical phenomena. These boundary conditions are needed in order to complete the formulation so that the problem becomes meaningful.

It has been shown that in Section 3.1.6, that the AGE method has been successfully applied to the two-point boundary value problem subject to the Dirichlet boundary conditions. In this section, we will investigate further the application of the AGE method when the problem is subject to other boundary conditions, i.e., periodic, Neumann and Combined.

Consider the second order ordinary differential equation

$$-\frac{d^2U}{dx^2} + \rho U(x) = f(x), \qquad a \le x \le b \tag{3.2-1}$$

where $f(x)$ is a real continuous function on $a \le x \le b$ and $\rho > 0$.

The boundary conditions associated to this problem is classified as follows:

1). Periodic, if $U(a) = U(b)$ and $U'(a) = U'(b)$. (3.2-2)

2). Neumann, if $U'(a) = \alpha$ and $U'(b) = \beta$. (3.2-3)

3). Combined, if $U(a) = \alpha$ and $U'(b) = \beta$.

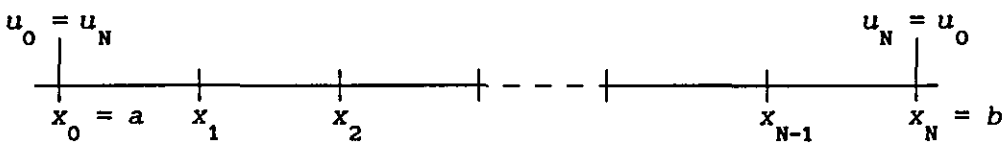
or $U'(a) = \alpha$ and $U(b) = \beta$. (3.2-4)

3.2.1 Periodic boundary conditions

For periodic boundary conditions, the mesh size is $h = (b-a)/N$, where N is the number of points on the interval $[a,b]$. Now, let us denote the mesh point x_i of the discrete problem as usual, i.e.,

$$x_i = a + ih, \qquad 0 \le i \le N. \tag{3.2.1-1}$$

Graphically, these points can be illustrated as



where $u_0 = u_N$. It is obvious that we are solving the N points in the interval $[a,b]$.

Now, denoting $U(x_i)$ by u_i and applying the finite difference (F.D.) central approximation to the second-order derivatives, and for small h, we have

$$\frac{-u_{i-1} + 2u_i - u_{i+1}}{h^2} + \rho u_i = f_i \tag{3.2.1-2}$$

which simplifies to

$$-u_{i-1} + 2gu_i - u_{i+1} = h^2 f_i, \qquad 1 \le i \le N. \tag{3.2.1-3}$$

where $g = 1 + \frac{1}{2}\rho h^2$. Since $u_0 = u_N$, then we have N equations for the N unknowns.

Equation (3.2.1-3) can be written in the matrix form $A\mathbf{u} = \mathbf{b}$, where

$$A = \begin{bmatrix} 2g & -1 & & & -1 \\ -1 & 2g & -1 & 0 & \\ & \ddots & \ddots & \ddots & \\ & 0 & -1 & 2g & -1 \\ -1 & & & -1 & 2g \end{bmatrix} \qquad (3.2.1\text{-}4)$$

$$\mathbf{u} = [u_1, u_2, \ldots, u_N]^T,$$

and $$\mathbf{b} = h^2[f_1, f_2, \ldots, f_N]^T.$$

It can be seen that the matrix A in (3.2.1-4) has two more non-zero elements, i.e., one at the top right corner, and the other one is at the bottom left corner. These two elements show that the matrix is derived from a problem with periodic boundary conditions.

Now, we split the matrix A in (3.2.1-4) into two submatrices as

$$A = G_1 + G_2 \qquad (3.2.1\text{-}5)$$

and first consider an even N which gives

$$G_1 = \begin{bmatrix} C & & & \\ & C & & 0 \\ & & \ddots & \\ & 0 & & C \\ & & & & C \end{bmatrix} \quad \text{and } G_2 = \begin{bmatrix} g & & & & -1 \\ & C & & 0 & \\ & & \ddots & & \\ & 0 & & C & \\ -1 & & & & g \end{bmatrix} \qquad (3.2.1\text{-}6)$$

with $C = \begin{bmatrix} g & -1 \\ -1 & g \end{bmatrix}$.

We now seek to analyse the convergence properties of the AGE method.

Evidently, G_1 and G_2 which consist of (2x2) block submatrices are positive definite and symmetric. Obviously, $(rI + G_1)$ and $(rI + G_2)$ are non-singular for any $r > 0$. Thus, their respective inverse does exist. By using the AGE method, the iteration matrix can be written as

$$T_r = (rI + G_2)^{-1}(rI - G_1)(rI + G_1)^{-1}(rI - G_2). \qquad (3.2.1\text{-}7)$$

We now need to prove that $\rho(T_r) < 1$.

Since G_1 and G_2 are symmetric and since $(rI - G_1)$ commutes with $(rI + G_1)^{-1}$ and by using a similarity transformation,

$$\tilde{T}_r = (rI + G_2)T_r(rI + G_2)^{-1} \tag{3.2.1-8}$$

then we have

$$\begin{aligned}
\rho(T_r) &= \rho(\tilde{T}_r) \\
&= \| T_r \|_2 \\
&= \|(rI - G_1)(rI + G_1)^{-1}(rI - G_2)(rI + G_2)^{-1}\|_2 \\
&\le \|(rI - G_1)(rI + G_1)^{-1}\|_2 \|(rI - G_2)(rI + G_2)^{-1}\|_2 .
\end{aligned} \tag{3.2.1-9}$$

It is obvious that

$$\begin{aligned}
\|(rI - G_1)(rI + G_1)^{-1}\|_2 &= \rho((rI - G_1)(rI + G_1)^{-1}) \\
&= \max_{\mu} \frac{|r - \mu|}{|r + \mu|} < 1.
\end{aligned} \tag{3.2.1-10}$$

The matrix G_2 can be transformed into compact (2x2) submatrices by a suitable permutation matrix P where $Q = PG_2P^T$ have the same eigenvalues as G_2. With this matrix permutation P, we have

$$Q = PG_2P^T = \begin{bmatrix} C & & & \\ & C & & 0 \\ & & \ddots & \\ & 0 & & C \end{bmatrix}. \tag{3.2.1-11}$$

Thus, from (3.2.1-11) we can conclude that

$$\begin{aligned}
\|(rI - G_2)(rI + G_2)^{-1}\|_2 &= \rho((rI - G_2)(rI + G_2)^{-1}) \\
&= \max_{\nu} \frac{|r - \nu|}{|r + \nu|} < 1.
\end{aligned} \tag{3.2.1-12}$$

Hence, $\rho(T_r) = \|(T_r)\|_2 < 1$. Thus, the AGE method is convergent.

Now we can apply the AGE method to determine $\mathbf{u}^{(k+1/2)}$ and $\mathbf{u}^{(k+1)}$ successively from equations (3.1.6-6) and (3.1.6-7). The vector $\mathbf{u}^{(k+1)}$ can be determined from $\mathbf{u}^{(k)}$ in two steps. One first determines $\mathbf{u}^{(k+1/2)}$ as follows:

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ \cdots \\ \cdots \\ u_{N-1} \\ u_N \end{bmatrix}^{(k+\frac{1}{2})} = d \begin{bmatrix} \alpha & 1 & & & & \\ 1 & \alpha & & & & \\ & & \alpha & 1 & & \\ & & 1 & \alpha & & \\ & & & & \ddots & \\ & & & & \alpha & 1 \\ & & & & 1 & \alpha \end{bmatrix} \times \left\{ \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ \cdots \\ \cdots \\ b_{N-1} \\ b_N \end{bmatrix} + \begin{bmatrix} \beta & & & & & 1 \\ & \beta & 1 & & & \\ & 1 & \beta & & & \\ & & & \ddots & & \\ & & & & \beta & 1 \\ & & & & 1 & \beta \\ 1 & & & & & \beta \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \cdots \\ \cdots \\ u_{N-2} \\ u_{N-1} \\ u_N \end{bmatrix}^{(k)} \right\}$$

which simplifies to

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ \cdots \\ \cdots \\ u_{N-1} \\ u_N \end{bmatrix}^{(k+\frac{1}{2})} = d \begin{bmatrix} \alpha & 1 & & & & \\ 1 & \alpha & & & & \\ & & \alpha & 1 & & \\ & & 1 & \alpha & & \\ & & & & \ddots & \\ & & & & \alpha & 1 \\ & & & & 1 & \alpha \end{bmatrix} \begin{bmatrix} b_1 + \beta u_1 + u_N \\ b_2 + \beta u_2 + u_3 \\ b_3 + u_2 + \beta u_3 \\ b_4 + \beta u_4 + u_5 \\ \cdots \\ \cdots \\ b_{N-1} + u_{N-2} + \beta u_{N-1} \\ b_N + u_1 + \beta u_N \end{bmatrix}^{(k)}$$

and by using the values of $u^{(k+1/2)}$ one determines $u^{(k+1)}$

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \cdots \\ \cdots \\ u_{N-2} \\ u_{N-1} \\ u_N \end{bmatrix}^{(k+1)} = d \begin{bmatrix} \alpha & & & & & 1 \\ & \alpha & 1 & & & \\ & 1 & \alpha & & & \\ & & & \ddots & & \\ & & & & \alpha & 1 \\ & & & & 1 & \alpha \\ 1 & & & & & \alpha \end{bmatrix} \times \left\{ \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \cdots \\ \cdots \\ b_{N-2} \\ b_{N-1} \\ b_N \end{bmatrix} + \right.$$

$$\begin{bmatrix} \beta & 1 & & & & \\ 1 & \beta & & & & \\ & & \beta & 1 & & \\ & & 1 & \beta & & \\ & & & & \ddots & \\ & & & & \beta & 1 \\ & & & & 1 & \beta \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ \cdots \\ \cdots \\ u_{N-1} \\ u_N \end{bmatrix}^{(k+\frac{1}{2})}$$

which simplifies to

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \cdots \\ \cdots \\ u_{N-2} \\ u_{N-1} \\ u_N \end{bmatrix}^{(k+1)} = d \begin{bmatrix} \alpha & & & & & & 1 \\ & \alpha & 1 & & & & \\ & 1 & \alpha & & & & \\ & & & \ddots & & & \\ & & & & \alpha & 1 & \\ & & & & 1 & \alpha & \\ 1 & & & & & & \alpha \end{bmatrix} \begin{bmatrix} b_1 + \beta u_1 + u_2 \\ b_2 + u_1 + \beta u_2 \\ b_3 + \beta u_3 + u_4 \\ \cdots \\ \cdots \\ b_{N-2} + u_{N-3} + \beta u_{N-2} \\ b_{N-1} + \beta u_{N-1} + u_N \\ b_N + u_{N-1} + \beta u_N \end{bmatrix}^{(k+\frac{1}{2})}$$

where $\alpha = r + g$ and $\beta = r - g$.

Now, consider the case when N is odd where the matrices G_1 and G_2 are given as follows

$$G_1 = \begin{bmatrix} C & & & & \\ & C & & & \\ & & \ddots & & 0 \\ & & & C & \\ & 0 & & & 0 \end{bmatrix} \quad \text{and } G_2 = \begin{bmatrix} g & & & & -1 \\ & C & & & \\ & & \ddots & & 0 \\ & & & C & \\ C_0 & & & & C_1 \end{bmatrix} \qquad (3.2.1\text{-}13)$$

with $C = \begin{bmatrix} g & -1 \\ -1 & g \end{bmatrix}$, $C_0 = \begin{bmatrix} 0 & 0 \\ -1 & 0 \end{bmatrix}$ and $C_1 = \begin{bmatrix} g & -1 \\ -1 & 2g \end{bmatrix}$.

We now analyse the convergence of the AGE method.

It is obvious that G_1 is symmetric and singular. However, for any $r > 0$, $(rI + G_1)$ is non-singular and positive definite. Hence, its inverse does exist. We will show that

$$\|(rI - G_1)(rI + G_1)^{-1}\|_2 = \rho((rI - G_1)(rI + G_1)^{-1})$$

$$= \max_{\mu} \frac{|r - \mu|}{|r + \mu|} \leq 1 \qquad (3.2.1\text{-}14)$$

holds.

From (3.2.1-13), the eigenvalues of G_1 are 0, $g - 1$ (repeated) and $g + 1$ (repeated). So, $\rho((rI - G_1)(rI + G_1)^{-1}) \leq 1$. Thus, (3.2.1-14) holds.

We now analyse the matrix G_2 in (3.2.1-13). It is obvious that G_2 is symmetric and positive definite. By using a suitable permutation matrix P, it can be shown that

$$Q = PG_2P^T = \begin{bmatrix} C_2 & & & 0 \\ & C & & \\ & & \ddots & \\ & & & C \\ 0 & & & C \end{bmatrix}, \qquad (3.2.1\text{-}15)$$

where Q is symmetric and positive definite and possess the same eigenvalues as G_2, with

$$C_2 = \left[\begin{array}{c|cc} g & -1 & 0 \\ \hline -1 & 2g & -1 \\ 0 & -1 & g \end{array}\right]. \qquad (3.2.1\text{-}16)$$

Hence, for $\overset{*}{Q} = P(rI - G_2)(rI + G_2)^{-1}P^T$ we have

$$\overset{*}{Q} = \begin{bmatrix} D_2 & & & 0 \\ & D & & \\ & & \ddots & \\ & & & D \\ 0 & & & D \end{bmatrix}, \qquad (3.2.1\text{-}17)$$

where $D_2 = (rI - C_2)(rI + C_2)^{-1}$ and $D = (rI - C)(rI + C)^{-1}$.

For the method to converge, we need the norm

$$\|(rI - G_2)(rI + G_2)^{-1}\|_2 = \rho((rI - G_2)(rI + G_2)^{-1}) < 1.$$

From (3.2.1-16), the eigenvalues of C_2 are

$$\nu_1 = \tfrac{3}{2}g - \tfrac{1}{2}\sqrt{g^2 + 8}, \quad \nu_2 = g, \quad \nu_3 = \tfrac{3}{2}g + \tfrac{1}{2}\sqrt{g^2 + 8}. \qquad (3.2.1\text{-}18)$$

Since g depends on the coefficient ρ and the mesh size h, the least value of g is 1 when $\rho = 0$. When $g = 1$, the eigenvalues given by (3.2.1-18) are 0, 1 and 3. When $g > 1$, then it can be shown further that all the eigenvalues are positive. Thus, we can conclude that

$$\|(rI - G_2)(rI + G_2)^{-1}\|_2 = \rho((rI - G_2)(rI + G_2)^{-1})$$

$$= \max_{\nu} \frac{|r - \nu|}{|r + \nu|} < 1. \qquad (3.2.1\text{-}19)$$

Hence, $\rho(T_r) = \|(T_r)\|_2 < 1$. Thus, the AGE method is convergent.

Now, by using the matrices G_1 and G_2 given by (3.2.1-13), we write the algorithmic procedure for the AGE method when N is odd.

From (3.1.6-6) and (3.1.6-7), one can determine $\mathbf{u}^{k+1}$ in two sweeps in the usual manner. First, to determine $\mathbf{u}^{k+1/2}$.

$$\begin{bmatrix} u_1 \\ u_2 \\ \cdots \\ \cdots \\ u_{N-2} \\ u_{N-1} \\ u_N \end{bmatrix}^{(k+1/2)} = d \begin{bmatrix} \alpha & 1 & & & & \\ 1 & \alpha & & & & \\ & & \ddots & & & \\ & & & \alpha & 1 & \\ & & & 1 & \alpha & \\ & & & & & 1/rd \end{bmatrix} \times \left\{ \begin{bmatrix} b_1 \\ b_2 \\ \cdots \\ b_{N-2} \\ b_{N-1} \\ b_N \end{bmatrix} + \right.$$

$$\left. \begin{bmatrix} \beta & & & & & 1 \\ & \beta & 1 & & & \\ & 1 & \beta & & & \\ & & & \ddots & & \\ & & & & \beta & 1 \\ 1 & & & & 1 & \beta_1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \cdots \\ \cdots \\ u_{N-1} \\ u_N \end{bmatrix}^{(k)} \right\}$$

which simplifies to

$$\begin{bmatrix} u_1 \\ u_2 \\ \cdots \\ \cdots \\ u_{N-2} \\ u_{N-1} \\ u_N \end{bmatrix}^{(k+1/2)} = d \begin{bmatrix} \alpha & 1 & & & & \\ 1 & \alpha & & & & \\ & & \ddots & & & \\ & & & \alpha & 1 & \\ & & & 1 & \alpha & \\ & & & & & 1/rd \end{bmatrix} \begin{bmatrix} b_1 + \beta u_1 + u_N \\ b_2 + \beta u_2 + u_3 \\ \cdots \\ \cdots \\ b_{N-2} + u_{N-3} + \beta u_{N-2} \\ b_{N-1} + \beta u_{N-1} + u_N \\ b_N + u_1 + u_{N-1} + \beta_1 u_N \end{bmatrix}^{(k)}$$

Now, to determine $\mathbf{u}^{k+1}$.

From (3.2.1-13), we can write

$$(rI + G_2) = \begin{bmatrix} \alpha & & & & 0 & -1 \\ & \alpha & -1 & & & \\ & -1 & \alpha & 0 & & \\ & & & \ddots & & \\ & & & \alpha & -1 & \\ & 0 & & -1 & \alpha & \\ 0 & & & & \alpha & -1 \\ -1 & & & & -1 & \alpha_1 \end{bmatrix},$$

and the inverse is given by

$$(rI + G_2)^{-1} = \begin{bmatrix} p & & & & q & s \\ & \alpha d & d & & & \\ & d & \alpha d & 0 & & \\ & & & \ddots & & \\ & & & \alpha d & d & \\ & 0 & & d & \alpha d & \\ q & & & & p & s \\ s & & & & s & t \end{bmatrix},$$

where $p = (\alpha\alpha_1 - 1)d_1$, $q = d_1$, $s = \alpha d_1$ and $t = \alpha^2 d_1$, and for $\alpha = r + g$, $\alpha_1 = r + 2g$, $d = 1/(\alpha^2 - 1)$ and $d_1 = 1/[\alpha(\alpha\alpha_1 - 2)]$.

Thus, $\mathbf{u}^{(k+1)}$ is given by

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \cdots \\ \cdots \\ u_{N-3} \\ u_{N-2} \\ u_{N-1} \\ u_N \end{bmatrix}^{(k+1)} = \begin{bmatrix} p & & & & q & s \\ & \alpha d & d & & & \\ & d & \alpha d & 0 & & \\ & & & \ddots & & \\ & & & \alpha d & d & \\ & 0 & & d & \alpha d & \\ q & & & & p & s \\ s & & & & s & t \end{bmatrix} \times \left\{ \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \cdots \\ \cdots \\ b_{N-3} \\ b_{N-2} \\ b_{N-1} \\ b_N \end{bmatrix} + \right.$$

$$\begin{bmatrix} \beta & 1 & & & & & \\ 1 & \beta & & & & & \\ & & \beta & 1 & & & \\ & & 1 & \beta & & & \\ & & & & \ddots & & \\ & & & & & \beta & 1 \\ & & & & & 1 & \beta \\ & & & & & & r \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ \cdots \\ \cdots \\ u_{N-2} \\ u_{N-1} \\ u_N \end{bmatrix}^{(k+1/2)}$$

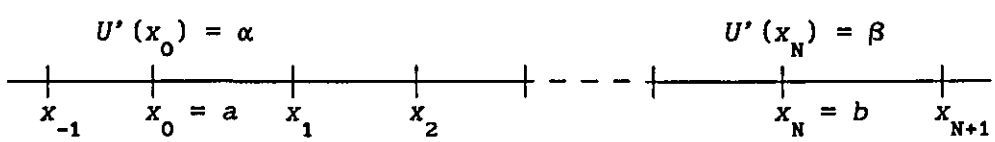
which simplifies to

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \cdots \\ \cdots \\ u_{N-3} \\ u_{N-2} \\ u_{N-1} \\ u_N \end{bmatrix}^{(k+1)} = \begin{bmatrix} p & & & & & q & s \\ & \alpha d & d & & 0 & & \\ & d & \alpha d & & & & \\ & & & \ddots & & & \\ & & 0 & & \alpha d & d & \\ & & & & d & \alpha d & \\ q & & & & & p & s \\ s & & & & & s & t \end{bmatrix} \begin{bmatrix} b_1 + \beta u_1 + u_2 \\ b_2 + u_1 + \beta u_2 \\ b_3 + \beta u_3 + u_4 \\ \cdots \\ \cdots \\ b_{N-3} + u_{N-4} + \beta u_{N-3} \\ b_{N-2} + \beta u_{N-2} + u_{N-1} \\ b_{N-1} + u_{N-2} + \beta u_{N-1} \\ b_N + r u_N \end{bmatrix}^{(k+1/2)}$$

where $\beta = r - g$, $\beta_1 = r - 2g$.

3.2.2 Neumann boundary conditions

Now with Neumann boundary conditions, the values become unknown at both ends of the boundary and the value of the derivatives are prescribed at both ends. The mesh points can be illustrated as



where the mesh size is $h = (b-a)/N$, and N is the number of points on the interval $[a,b]$. Now, let us denote the mesh point x_i of the discrete problem as usual, i.e.,

$$x_i = a + ih, \qquad 0 \le i \le N. \tag{3.2.2-1}$$

Since the values at the boundaries are prescribed in terms of the derivatives, then we have to estimate these values. To get a better accuracy, this estimation can be obtained by using the centred finite difference approximation, for small h

$$U'(x_N) = \frac{u_{N+1} - u_{N-1}}{2h} \quad \text{or} \quad u_{N+1} - u_{N-1} = 2hU'(x_N). \qquad (3.2.2\text{-}2)$$

From (3.2.2-2), the approximation for $U'(x_0)$ and $U'(x_N)$ are given by

$$u_1 - u_{-1} = 2hU'(x_0) \text{ and } u_{N+1} - u_{N-1} = 2hU'(x_N) \qquad (3.2.2\text{-}3)$$

respectively, where h is small.

By using the usual finite difference approximation, the difference equation for the model problem is given by

$$-u_{i-1} + 2gu_i - u_{i+1} = h^2 f_i, \qquad 0 \le i \le N \qquad (3.2.2\text{-}4)$$

where $g = 1 + \frac{1}{2}\rho h^2$.

When these equations are expressed in the matrix form $A\mathbf{u} = \mathbf{b}$, then we have

$$A = \begin{bmatrix} 2g & -1 & & & 0 \\ -1 & 2g & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2g & -1 \\ 0 & & & -1 & 2g \end{bmatrix}_{N+1,N+1} \qquad (3.2.2\text{-}5)$$

where $\mathbf{u} = [u_0, u_1, u_2, \ldots, u_N]^T$, and

$\mathbf{b} = [h^2 f_0 + u_{-1}, h^2 f_1, h^2 f_2, \ldots, h^2 f_{N-1}, h^2 f_N + u_{N+1}]^T$.

Now, rearrange (3.2.2-3) and put in the boundary conditions then we have

$$u_{-1} = u_1 - 2h\alpha \text{ and } u_{N+1} = u_{N-1} + 2h\beta. \qquad (3.2.2\text{-}6)$$

By substituting u_{-1} and u_{N+1} into the vector $\mathbf{b}$ and rearranging, we then have a new linear system of equations $A\mathbf{u} = \mathbf{b}$, where

$$A = \begin{bmatrix} 2g & -2 & & & 0 \\ -1 & 2g & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2g & -1 \\ 0 & & & -2 & 2g \end{bmatrix} \qquad (3.2.2\text{-}7)$$

$$\mathbf{u} = [u_0,\ u_1,\ u_2,\ \ldots,\ u_N]^T,$$

and $$\mathbf{b} = [h^2 f_0 - 2h\alpha,\ h^2 f_1,\ h^2 f_2,\ \ldots,\ h^2 f_{N-1},\ h^2 f_N + 2h\beta]^T.$$

We now solve the system with N+1 number of points, including the two points on the boundary, i.e., u_0 and u_N.

First, we consider the case when N is odd. By using the AGE method, we split the matrix A into two submatrices G_1 and G_2 as

$$G_1 = \begin{bmatrix} 0 & & & \\ & C & & 0 \\ & & \ddots & \\ & 0 & & C \\ & & & 0 \end{bmatrix} \text{ and } G_2 = \begin{bmatrix} C_1 & & & \\ & C & & 0 \\ & & \ddots & \\ & 0 & & C \\ & & & C_2 \end{bmatrix} \qquad (3.2.2\text{-}8)$$

where

$$C = \begin{bmatrix} g & -1 \\ -1 & g \end{bmatrix},\ C_1 = \begin{bmatrix} 2g & -2 \\ -1 & g \end{bmatrix} \text{ and } C_2 = \begin{bmatrix} g & -1 \\ -2 & 2g \end{bmatrix}. \qquad (3.2.2\text{-}9)$$

We now seek to analyse the convergence properties of the AGE method.

Evidently, G_1 is singular and symmetric. However, for any $r > 0$, $(rI + G_1)$ is non-singular and positive definite. Thus, its inverse, i.e., $(rI + G_1)^{-1}$ exists. Since the eigenvalues of G_1 are non-negative, it can be shown that the norm

$$\begin{aligned} \|(rI - G_1)(rI + G_1)^{-1}\| &= \rho((rI - G_1)(rI + G_1)^{-1}) \\ &= \max_{\mu} \left|\frac{r - \mu}{r + \mu}\right| \le 1 \end{aligned} \qquad (3.2.2\text{-}10)$$

holds.

Now, let us examine the matrix G_2. It is obvious that, for $\rho > 0$, the matrix G_2 given in (3.2.2-8) is non-singular with positive eigenvalues. Thus, convergence may still hold. Moreover, for any $r > 0$, $(rI + G_2)^{-1}$ exists, and we have

$$(rI - G_2)(rI + G_2)^{-1} = \begin{bmatrix} D_1 & & & \\ & D & & 0 \\ & & \ddots & \\ & 0 & & D \\ & & & & D_2 \end{bmatrix} \qquad (3.2.2\text{-}11)$$

where $D_1 = (rI - C_1)(rI + C_1)^{-1}$, $D = (rI - C)(rI + C)^{-1}$

and $D_2 = (rI - C_2)(rI + C_2)^{-1}$.

From (3.2.2-9), the respective eigenvalues of C, C_1 and C_2 are

$$\nu_{1,2} = g \pm 1 \quad \text{and} \quad \nu_{3,4} = \tfrac{3}{2}g \pm \tfrac{1}{2}\sqrt{g^2 + 8}. \qquad (3.2.2\text{-}12)$$

For $g > 1$, It is obvious that these eigenvalues are positive. When $g = 1$, the eigenvalues given by (3.2.2-12) are 0, 2 and 3.

Thus, we may conclude that

$$\|(rI - G_2)(rI + G_2)^{-1}\| = \max_{\nu} \left|\frac{r - \nu}{r + \nu}\right| \leq 1 \qquad (3.2.2\text{-}13)$$

holds.

The equals sign both in (3.2.2-10) and (3.2.2.13), because the eigenvalues in G_1 and G_2 can have the value zero. This implies that the AGE method is only convergent when $\rho > 0$, i.e., $g > 1$.

Now, we write the algorithmic procedure for the AGE method, i.e, to determine $\mathbf{u}^{(k+1)}$ in two steps by using the equation (3.1.6-6) and (3.1.6-7). From (3.2.2-9) and (3.2.2-11), we have

$$D = (rI + C)^{-1} = d \begin{bmatrix} \alpha & 1 \\ 1 & \alpha \end{bmatrix}, \quad D_1 = (rI + C_1)^{-1} = d_1 \begin{bmatrix} \alpha & 2 \\ 1 & \alpha_1 \end{bmatrix},$$

and $D_2 = (rI + C_2)^{-1} = d_1 \begin{bmatrix} \alpha_1 & 1 \\ 2 & \alpha \end{bmatrix}$. $\qquad (3.2.2\text{-}14)$

with $\alpha = r + g$, $\alpha_1 = r + 2g$, $d = \dfrac{1}{\alpha^2 - 1}$ and $d_1 = \dfrac{1}{\alpha\alpha_1 - 2}$.

I) To determine $u^{(k+1/2)}$.

$$\begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \cdots \\ \cdots \\ u_{N-2} \\ u_{N-1} \\ u_N \end{bmatrix}^{(k+\frac{1}{2})} = d \begin{bmatrix} 1/rd & & & & & \\ & \alpha & 1 & & & \\ & 1 & \alpha & & & \\ & & & \ddots & & \\ & & & & \alpha & 1 \\ & & & & 1 & \alpha \\ & & & & & 1/rd \end{bmatrix} \times \left\{ \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \cdots \\ \cdots \\ b_{N-2} \\ b_{N-1} \\ b_N \end{bmatrix} + \right.$$

$$\left. \begin{bmatrix} \beta_1 & 2 & & & & & \\ 1 & \beta & & & & & \\ & & \beta & 1 & & & \\ & & 1 & \beta & & & \\ & & & & \ddots & & \\ & & & & & \beta & 1 \\ & & & & & 1 & \beta \\ & & & & & 2 & \beta_1 \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ \cdots \\ \cdots \\ u_{N-3} \\ u_{N-2} \\ u_{N-1} \\ u_N \end{bmatrix}^{(k)} \right\}$$

which simplifies to

$$\begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \cdots \\ \cdots \\ u_{N-2} \\ u_{N-1} \\ u_N \end{bmatrix}^{(k+\frac{1}{2})} = d \begin{bmatrix} 1/rd & & & & & \\ & \alpha & 1 & & & \\ & 1 & \alpha & & & \\ & & & \ddots & & \\ & & & & \alpha & 1 \\ & & & & 1 & \alpha \\ & & & & & 1/rd \end{bmatrix} \begin{bmatrix} b_0 + \beta_1 u_0 + 2u_1 \\ b_1 + u_0 + \beta u_1 \\ b_2 + \beta u_2 + u_3 \\ \cdots \\ \cdots \\ b_{N-2} + u_{N-3} + \beta u_{N-2} \\ b_{N-1} + \beta u_{N-1} + u_N \\ b_N + 2u_{N-1} + \beta_1 u_N \end{bmatrix}^{(k)}$$

II) To determine $u^{(k+1)}$.

$$\begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ \cdots \\ \cdots \\ u_{N-3} \\ u_{N-2} \\ u_{N-1} \\ u_N \end{bmatrix}^{(k+1)} = \begin{bmatrix} p & r & & & & & \\ d_1 & q & & & & & \\ & & \alpha d & d & & & \\ & & d & \alpha d & & & \\ & & & & \ddots & & \\ & & & & & \alpha d & d \\ & & & & & d & \alpha d \\ & & & & & & & q & d_1 \\ & & & & & & & r & p \end{bmatrix} \times \left\{ \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ \cdots \\ \cdots \\ b_{N-3} \\ b_{N-2} \\ b_{N-1} \\ b_N \end{bmatrix} + \begin{bmatrix} r & & & & & & \\ & \beta & 1 & & & & \\ & 1 & \beta & & & & \\ & & & \ddots & & & \\ & & & & \beta & 1 & \\ & & & & 1 & \beta & \\ & & & & & & r \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \cdots \\ \cdots \\ u_{N-2} \\ u_{N-1} \\ u_N \end{bmatrix}^{(k+\frac{1}{2})} \right\}$$

which simplifies to

$$\begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ \cdots \\ \cdots \\ u_{N-3} \\ u_{N-2} \\ u_{N-1} \\ u_N \end{bmatrix}^{(k+1)} = \begin{bmatrix} p & r & & & & & \\ d_1 & q & & & & & \\ & & \alpha d & d & & & \\ & & d & \alpha d & & & \\ & & & & \ddots & & \\ & & & & & \alpha d & d \\ & & & & & d & \alpha d \\ & & & & & & & q & d_1 \\ & & & & & & & r & p \end{bmatrix} \begin{bmatrix} b_0 + ru_0 \\ b_1 + \beta u_1 + u_2 \\ b_2 + u_1 + \beta u_2 \\ b_3 + \beta u_3 + u_4 \\ \cdots \\ \cdots \\ b_{N-3} + u_{N-4} + \beta u_{N-3} \\ b_{N-2} + \beta u_{N-2} + u_{N-1} \\ b_{N-1} + u_{N-2} + \beta u_{N-1} \\ b_N + ru_N \end{bmatrix}^{(k+\frac{1}{2})}$$

where $\beta = r - g$, $\beta_1 = r - 2g$, $p = \alpha d_1$, $q = \alpha_1 d_1$ and $r = 2d_1$.

Now we examine G_2 for the case when N is even. Let us consider the splitting of A into G_1 and G_2 as follows:

$$G_1 = \begin{bmatrix} 0 & & & \\ C & \ddots & 0 & \\ & \ddots & \ddots & \\ & 0 & C & C_2 \end{bmatrix} \quad \text{and} \quad G_2 = \begin{bmatrix} C_1 & C & & \\ & \ddots & \ddots & \\ & 0 & \ddots & C \\ & & & 0 \end{bmatrix} \tag{3.2.2-15}$$

where C_1 and C_2 as given in (3.2.2-9).

It is obvious both G_1 and G_2 are singular. However, for any $r > 0$, $(rI + G_1)$ and $(rI + G_2)$ are non-singular. Thus, their inverse exists. For convergence, we need

$$\|(rI - G_1)(rI + G_1)^{-1}\| = \max_{\mu} \left|\frac{r - \mu}{r + \mu}\right| < 1 \tag{3.2.2-16}$$

and

$$\|(rI - G_2)(rI + G_2)^{-1}\| = \max_{\nu} \left|\frac{r - \nu}{r + \nu}\right| < 1. \tag{3.2.2-17}$$

It has been shown that the eigenvalues of C, C_1 and C_2 are positive for $g > 1$. Again, when $g = 1$, we have a similar situation as in the case when N is even. Thus, we can conclude that (3.2.2-17) and (3.2.2-18) hold, i.e., the AGE method is convergent for $\rho > 0$.

Now, we write the algorithmic procedure for the AGE method, i.e., to determine $\mathbf{u}^{(k+1)}$ in two steps by using the equation (3.1.6-6) and (3.1.6-7). The matrices $(rI + G_1)^{-1}$ and $(rI + G_2)^{-1}$ can be determined by using (3.2.2-14).

I) To determine $\mathbf{u}^{(k+1/2)}$.

$$\begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_{N-3} \\ u_{N-2} \\ u_{N-1} \\ u_N \end{bmatrix}^{(k+\frac{1}{2})} = \left[\begin{array}{c|cc|c|cc} 1/r & & & & & \\ \hline & \alpha d & d & & & \\ & d & \alpha d & & & \\ \hline & & & \ddots & & \\ \hline & & & & \alpha d & d \\ & & & & d & \alpha d \\ \hline & & & & & q \quad d_1 \\ & & & & & r \quad p \end{array}\right] \times \left\{ \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_{N-3} \\ b_{N-2} \\ b_{N-1} \\ b_N \end{bmatrix} + \right.$$

$$\begin{bmatrix} \beta_1 & 2 & & & & & \\ 1 & \beta & & & & & \\ & & \beta & 1 & & & \\ & & 1 & \beta & & & \\ & & & & \ddots & & \\ & & & & & \beta & 1 \\ & & & & & 1 & \beta \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ \cdots \\ \cdots \\ u_{N-2} \\ u_{N-1} \\ u_N \end{bmatrix}^{(k)} \Bigg\}$$

which simplifies to

$$\begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_{N-3} \\ u_{N-2} \\ u_{N-1} \\ u_N \end{bmatrix}^{(k+\frac{1}{2})} = \begin{bmatrix} 1/r & & & & & \\ & \alpha d & d & & & \\ & d & \alpha d & & & \\ & & & \ddots & & \\ & & & & \alpha d & d \\ & & & & d & \alpha d \\ & & & & & \end{bmatrix} \begin{bmatrix} b_0 + \beta_1 u_0 + 2u_1 \\ b_1 + u_0 + \beta u_1 \\ b_2 + \beta u_2 + u_3 \\ \vdots \\ b_{N-3} + u_{N-4} + \beta u_{N-3} \\ b_{N-2} + \beta u_{N-2} + u_{N-1} \\ b_{N-1} + u_{N-2} + \beta u_{N-1} \\ b_N + r u_N \end{bmatrix}^{(k)}$$

II) To determine $\mathbf{u}^{(k+1)}$.

$$\begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ \cdots \\ \cdots \\ u_{N-2} \\ u_{N-1} \\ u_N \end{bmatrix}^{(k+1)} = \begin{bmatrix} p & r & & & & & \\ d_1 & q & & & & & \\ & & \alpha d & d & & & \\ & & d & \alpha d & & & \\ & & & & \ddots & & \\ & & & & & \alpha d & d \\ & & & & & d & \alpha d \\ & & & & & & 1/r \end{bmatrix} \times \left\{ \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ \cdots \\ \cdots \\ b_{N-2} \\ b_{N-1} \\ b_N \end{bmatrix} + \right.$$

$$\left. \begin{bmatrix} r & & & & & & \\ & \beta & 1 & & & & \\ & 1 & \beta & & & & \\ & & & \ddots & & & \\ & & & & \beta & 1 & \\ & & & & 1 & \beta & \\ & & & & & \beta & 1 \\ & & & & & 2 & \beta_1 \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_{N-3} \\ u_{N-2} \\ u_{N-1} \\ u_N \end{bmatrix}^{(k+\frac{1}{2})} \right\}$$

which simplifies to

$$\begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ \cdots \\ \cdots \\ u_{N-2} \\ u_{N-1} \\ u_N \end{bmatrix}^{(k+1)} = \left[\begin{array}{cc|cc|cc|c} p & r & & & & & & \\ d_1 & q & & & & & & \\ \hline & & \alpha d & d & & & & \\ & & d & \alpha d & & & & \\ \hline & & & & \ddots & & & \\ \hline & & & & & \alpha d & d & \\ & & & & & d & \alpha d & \\ \hline & & & & & & & 1/r \end{array}\right] \begin{bmatrix} b_0 + ru_0 \\ b_1 + \beta u_1 + u_2 \\ b_2 + u_1 + \beta u_2 \\ b_3 + \beta u_3 + u_4 \\ \cdots \\ \cdots \\ b_{N-2} + u_{N-3} + \beta u_{N-2} \\ b_{N-1} + \beta u_{N-1} + u_N \\ b_N + 2u_{N-1} + \beta_1 u_N \end{bmatrix}^{(k+\frac{1}{2})}$$

where $\beta = r - g$, $\beta_1 = r - 2g$, $p = \alpha d_1$, $q = \alpha_1 d_1$ and $r = 2d_1$.

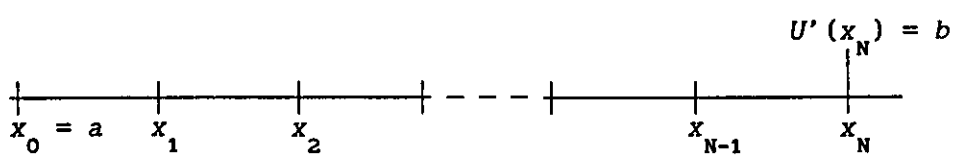
3.2.3 Combined boundary conditions

Let consider the combined boundary conditions for the problem (3.2-1) as

$$U(a) = \alpha, \qquad U'(b) = \beta \tag{3.2.3-1}$$

The left boundary is prescribed as a Dirichlet boundary condition, but the right boundary is given in derivative form. As for a Neumann boundary condition, this boundary can be approximated by using the centred difference formula (3.2.2-2).

Let us consider the mesh size $h = (b-a)/N$, then the value on the boundary can be taken as $u_o = \alpha$ and $U'(x_N) = \beta$. Graphically, the points can be illustrated as follows:



Thus, we solve N equations with N unknowns, including the value u_N at the boundary.

By using the usual finite difference approximation, for small h, the difference equation for the model problem is given by

$$-u_{i-1} + 2gu_i - u_{i+1} = h^2 f_i, \qquad 1 \le i \le N \tag{3.2.3-2}$$

where $g = 1 + \rho h^2/2$.

In matrix form, i.e., $A\mathbf{u} = \mathbf{b}$, equation (3.2.3-2) will give

$$A = \begin{bmatrix} 2g & -1 & & & 0 \\ -1 & 2g & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2g & -1 \\ 0 & & & -1 & 2g \end{bmatrix} \tag{3.2.3-3}$$

$$\mathbf{u} = [u_1, u_2, \ldots, u_N]^T,$$

and $$\mathbf{b} = [h^2 f_1 + \alpha, h^2 f_2, \ldots, h^2 f_{N-1}, h^2 f_N + u_{N+1}]^T.$$

By using the approximation (3.2.2-3), then we have

$$u_{N+1} = u_{N-1} + 2h\beta \tag{3.2.3-5}$$

Thus, the new system of linear equations becomes

$$A = \begin{bmatrix} 2g & -1 & & & 0 \\ -1 & 2g & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2g & -1 \\ 0 & & & -2 & 2g \end{bmatrix} \tag{3.2.3-6}$$

$$\mathbf{u} = [u_1, u_2, \ldots, u_N]^T,$$

and $$\mathbf{b} = [h^2 f_1 + \alpha, h^2 f_2, \ldots, h^2 f_{N-1}, h^2 f_N + 2h\beta]^T.$$

We now seek to analyse the convergence properties of the AGE method. First, let us consider the case when N is even. We split the matrix A of (3.2.3-6) into two submatrices G_1 and G_2, as

$$G_1 = \begin{bmatrix} g & & & & \\ & c & & 0 & \\ & & \ddots & & \\ & & & c & \\ 0 & & & & 0 \end{bmatrix} \text{ and } G_2 = \begin{bmatrix} c & & & & \\ & c & & 0 & \\ & & \ddots & & \\ & & & c & \\ 0 & & & & c_1 \end{bmatrix}, \tag{3.2.3-7}$$

where

$$C = \begin{bmatrix} g & -1 \\ -1 & g \end{bmatrix}, \quad C_1 = \begin{bmatrix} g & -1 \\ -2 & 2g \end{bmatrix}. \tag{3.2.3-8}$$

Evidently, G_1 is singular. However, for any $r > 0$, $(rI + G_1)$ is non-singular. Thus, $(rI + G_1)^{-1}$ exist. It is obvious that the smallest eigenvalue of G_1 in (3.2.3-7) is zero. Thus, it can be shown that, the norm

$$\|(rI - G_1)(rI + G_1)^{-1}\| = \max_{\mu} \left|\frac{r - \mu}{r + \mu}\right| \leq 1 \tag{3.2.3-9}$$

holds.

For convergence, we need to show that the norm

$$\|(rI - G_2)(rI + G_2)^{-1}\| = \max_{\nu} \left|\frac{r - \nu}{r + \nu}\right| < 1 \tag{3.2.3-10}$$

also holds.

It has been shown in the previous sections that the eigenvalues of C and C_1 are positive for $g > 1$. Since these eigenvalues belong to the matrix G_2 in (3.2.3-7), then (3.2.3-10) holds.

Thus, from (3.2.3-9) and (3.2.3-10), the AGE method is convergent. From (3.2.3-8), we have

$$D = (rI + C)^{-1} = d \begin{bmatrix} \alpha & 1 \\ 1 & \alpha \end{bmatrix},$$

$$\text{and} \quad D_1 = (rI + C_1)^{-1} = d_1 \begin{bmatrix} \alpha_1 & 1 \\ 2 & \alpha \end{bmatrix}, \tag{3.2.3-11}$$

where $\alpha = r + g$, $\alpha_1 = r + 2g$, $d = \dfrac{1}{\alpha^2 - 1}$ and $d_1 = \dfrac{1}{\alpha\alpha_1 - 2}$.

Now, we write the algorithmic procedure for the AGE method, i.e., to determine $\mathbf{u}^{(k+1)}$ in two steps by using the equation (3.1.6-6) and (3.1.6-7).

I) To determine $\mathbf{u}^{(k+1/2)}$.

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \cdots \\ \cdots \\ u_{N-2} \\ u_{N-1} \\ u_N \end{bmatrix}^{(k+\frac{1}{2})} = \begin{bmatrix} 1/\alpha & & & & & & \\ & \alpha d & d & & & & \\ & d & \alpha d & & & & \\ & & & \ddots & & & \\ & & & & \alpha d & d & \\ & & & & d & \alpha d & \\ & & & & & & 1/r \end{bmatrix} \times \left\{ \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \cdots \\ \cdots \\ b_{N-2} \\ b_{N-1} \\ b_N \end{bmatrix} + \right.$$

$$\left. \begin{bmatrix} \beta & 1 & & & & & \\ 1 & \beta & & & & & \\ & & \ddots & & & & \\ & & & \beta & 1 & & \\ & & & 1 & \beta & & \\ & & & & & \beta & 1 \\ & & & & & 2 & \beta_1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \cdots \\ \cdots \\ u_{N-3} \\ u_{N-2} \\ u_{N-1} \\ u_N \end{bmatrix}^{(k)} \right\}$$

which simplifies to

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \cdots \\ \cdots \\ u_{N-2} \\ u_{N-1} \\ u_N \end{bmatrix}^{(k+\frac{1}{2})} = \begin{bmatrix} 1/\alpha & & & & & & \\ & \alpha d & d & & & & \\ & d & \alpha d & & & & \\ & & & \ddots & & & \\ & & & & \alpha d & d & \\ & & & & d & \alpha d & \\ & & & & & & 1/r \end{bmatrix} \begin{bmatrix} b_1 + \beta u_1 + u_2 \\ b_2 + u_1 + \beta u_2 \\ b_3 + \beta u_3 + u_4 \\ \cdots \\ \cdots \\ b_{N-2} + u_{N-3} + \beta u_{N-2} \\ b_{N-1} + \beta u_{N-1} + u_N \\ b_N + 2u_{N-1} + \beta_1 u_N \end{bmatrix}^{(k)} .$$

II) To determine $\mathbf{u}^{(k+1)}$.

$$\begin{bmatrix} u_1 \\ u_2 \\ \cdots \\ \cdots \\ u_{N-3} \\ u_{N-2} \\ u_{N-1} \\ u_N \end{bmatrix}^{(k+1)} = \begin{bmatrix} \alpha d & d & & & & \\ d & \alpha d & & & & \\ & & \ddots & & & \\ & & & \alpha d & d & \\ & & & d & \alpha d & \\ & & & & & q \quad d_1 \\ & & & & & r \quad p \end{bmatrix} \times \left\{ \begin{bmatrix} b_1 \\ b_2 \\ \cdots \\ \cdots \\ b_{N-3} \\ b_{N-2} \\ b_{N-1} \\ b_N \end{bmatrix} + \right.$$

$$\begin{bmatrix} \beta & & & & & \\ & \beta & 1 & & & \\ & 1 & \beta & & & \\ & & & \ddots & & \\ & & & & \beta & 1 \\ & & & & 1 & \beta \\ & & & & & r \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \cdots \\ \cdots \\ u_{N-2} \\ u_{N-1} \\ u_N \end{bmatrix}^{(k+\frac{1}{2})}$$

which simplifies to

$$\begin{bmatrix} u_1 \\ u_2 \\ \cdots \\ \cdots \\ u_{N-3} \\ u_{N-2} \\ u_{N-1} \\ u_N \end{bmatrix}^{(k+1)} = \begin{bmatrix} \alpha d & d & & & & \\ d & \alpha d & & & & \\ & & \ddots & & & \\ & & & \alpha d & d & \\ & & & d & \alpha d & \\ & & & & & q \quad d_1 \\ & & & & & r \quad p \end{bmatrix} \begin{bmatrix} b_1 + \beta u_1 \\ b_2 + \beta u_2 + u_3 \\ \cdots \\ \cdots \\ b_{N-3} + u_{N-4} + \beta u_{N-3} \\ b_{N-2} + \beta u_{N-2} + u_{N-1} \\ b_{N-1} + u_{N-2} + \beta u_{N-1} \\ b_N + r u_N \end{bmatrix}^{(k+\frac{1}{2})}$$

where $\beta = r - g$, $\beta_1 = r - 2g$, $p = \alpha d_1$, $q = \alpha_1 d_1$ and $r = 2d_1$.

We now, analyse the convergence properties of the AGE method when N is odd. Consider the splitting of A into G_1 and G_2 as follows:

$$G_1 = \begin{bmatrix} C & & & & \\ & C & & 0 & \\ & & \ddots & & \\ & 0 & & C & \\ & & & & 0 \end{bmatrix} \quad \text{and } G_2 = \begin{bmatrix} g & & & & \\ & C & & 0 & \\ & & \ddots & & \\ & 0 & & C & \\ & & & & C_1 \end{bmatrix} \qquad (3.2.3\text{-}12)$$

where $C = \begin{bmatrix} g & -1 \\ -1 & g \end{bmatrix}$ and $C_1 = \begin{bmatrix} g & -1 \\ -2 & 2g \end{bmatrix}$.

Evidently, G_1 is singular. However, for any $r > 0$, $(rI + G_1)$ is non-singular and positive definite. Thus, its inverse exists. Since the eigenvalues of G_1 are non-negative, then

$$\|(rI - G_1)(rI + G_1)^{-1}\|_2 = \rho((rI - G_1)(rI + G_1)^{-1})$$

$$= \max_{\mu} \left| \frac{r - \mu}{r + \mu} \right| \leq 1 \qquad (3.2.2\text{-}13)$$

holds.

Evidently. the matrix G_2 is not symmetric but all the eigenvalues are positive. Thus, convergence may still hold. Hence, we can conclude that the norm

$$\|(rI - G_2)(rI + G_2)^{-1}\| = \max_{\nu} \left|\frac{r - \nu}{r + \nu}\right| < 1 \qquad (3.2.2\text{-}14)$$

holds.

Thus, from (3.2.2-14) and (3.2.2-15), the AGE method is convergent.

Now, by using the equation (3.1.6-6) and (3.1.6-7), we write the algorithmic procedure for the AGE method, i.e., to determine $\mathbf{u}^{(k+1)}$ in two steps as follows:

I) To determine $\mathbf{u}^{(k+1/2)}$.

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \cdots \\ \cdots \\ u_{N-1} \\ u_N \end{bmatrix}^{(k+\frac{1}{2})} = \begin{bmatrix} \alpha d & d & & & & \\ d & \alpha d & & & & \\ & & \ddots & & & \\ & & & \alpha d & d & \\ & & & d & \alpha d & \\ & & & & & 1/r \end{bmatrix} \times \left\{ \begin{bmatrix} b_1 \\ b_2 \\ \cdots \\ \cdots \\ b_{N-2} \\ b_{N-1} \\ b_N \end{bmatrix} + \begin{bmatrix} \beta & & & & & \\ & \beta & 1 & & & \\ & 1 & \beta & & & \\ & & & \ddots & & \\ & & & & \beta & 1 \\ & & & & 1 & \beta \\ & & & & & \beta & 1 \\ & & & & & 2 & \beta_1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \cdots \\ u_{N-3} \\ u_{N-2} \\ u_{N-1} \\ u_N \end{bmatrix}^{(k)} \right\}$$

which simplifies to

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \cdots \\ \cdots \\ u_{N-1} \\ u_N \end{bmatrix}^{(k+\frac{1}{2})} = \begin{bmatrix} \alpha d & d & & & & \\ d & \alpha d & & & & \\ & & \ddots & & & \\ & & & \alpha d & d & \\ & & & d & \alpha d & \\ & & & & & 1/r \end{bmatrix} \begin{bmatrix} b_1 + \beta u_1 \\ b_2 + \beta u_2 + u_3 \\ \cdots \\ \cdots \\ b_{N-2} + u_{N-3} + \beta u_{N-2} \\ b_{N-1} + \beta u_{N-1} + u_N \\ b_N + 2u_{N-1} + \beta_1 u_N \end{bmatrix}^{(k)}.$$

II) To determine $\mathbf{u}^{(k+1)}$.

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \cdots \\ \cdots \\ u_{N-3} \\ u_{N-2} \\ u_{N-1} \\ u_N \end{bmatrix}^{(k+1)} = \begin{bmatrix} 1/\alpha & & & & & \\ & \alpha d & d & & & \\ & d & \alpha d & & & \\ & & & \ddots & & \\ & & & & \alpha d & d \\ & & & & d & \alpha d \\ & & & & & \begin{matrix} q & d_1 \\ r & p \end{matrix} \end{bmatrix} \times \left\{ \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \cdots \\ \cdots \\ b_{N-3} \\ b_{N-2} \\ b_{N-1} \\ b_N \end{bmatrix} + \begin{bmatrix} \beta & 1 & & & & \\ 1 & \beta & & & & \\ & & \ddots & & & \\ & & & \beta & 1 & \\ & & & 1 & \beta & \\ & & & & & r \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \cdots \\ \cdots \\ u_{N-2} \\ u_{N-1} \\ u_N \end{bmatrix}^{(k+\frac{1}{2})} \right\}$$

which simplifies to

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \cdots \\ \cdots \\ u_{N-3} \\ u_{N-2} \\ u_{N-1} \\ u_N \end{bmatrix}^{(k+1)} = \begin{bmatrix} 1/\alpha & & & & & \\ & \alpha d & d & & & \\ & d & \alpha d & & & \\ & & & \ddots & & \\ & & & & \alpha d & d \\ & & & & d & \alpha d \\ & & & & & \begin{matrix} q & d_1 \\ r & p \end{matrix} \end{bmatrix} \begin{bmatrix} b_1 + \beta u_1 + u_2 \\ b_2 + u_1 + \beta u_2 \\ b_3 + \beta u_3 + u_4 \\ \cdots \\ \cdots \\ b_{N-3} + u_{N-4} + \beta u_{N-3} \\ b_{N-2} + \beta u_{N-2} + u_{N-1} \\ b_{N-1} + u_{N-2} + \beta u_{N-1} \\ b_N + r u_N \end{bmatrix}^{(k+\frac{1}{2})}$$

where $\beta = r - g$, $\beta_1 = r - 2g$, $p = \alpha d_1$, $q = \alpha_1 d_1$ and $r = 2d_1$.

3.2.4 Experimental results

Numerical results presented here are for the SOR and AGE methods for solving the two-point boundary value problem subject to boundary conditions described in Sections 3.2.1, 3.2.2 and 3.2.3.

Four problems have been considered except for the case for periodic boundary conditions. Problem 1 is presented in greater detail with some variations on ρ.

I) The results for periodic boundary conditions.

Problem 1 - A Linear Problem

$$-U'' + \rho U = (\rho+1)(\sin x + \cos x), \qquad 0 \le x \le 2\pi,$$

$$U(0) = U(2\pi), \text{ and } U'(0) = U'(2\pi).$$

The exact solution is $U(x) = \sin x + \cos x$.

The results for various ρ are tabulated as follows.

Method	SOR		AGE	
N	ω	iter	r	iter
10	1.10-1.15	30	1.04-1.08	10
20	1.44	58	0.59	16
40	1.68-1.69	110	0.30	31
80	1.84	204	0.13	62
160	1.92	374	0.06	109

Table 3.2.4-1: Problem 1 with $\rho = 0$

Method	SOR		AGE	
N	ω	iter	r	iter
11	1.29-1.33	32	0.97-0.98	11
21	1.45-1.47	61	0.52-0.54	20
41	1.69	112	0.26	35
81	1.84	206	0.13	64
161	1.92	377	0.06	110

Table 3.2.4-2: Problem 1 with $\rho = 0$

Method	SOR		AGE	
N	ω	iter	r	iter
10	1.25-1.27	17	1.41-1.42	7
20	1.53	32	0.71-0.72	13
40	1.74	61	0.36	23
80	1.82	113	0.18	42
160	1.91	208	0.06	73

Table 3.2.4-3: Problem 1 with $\rho = 1$

Method	SOR		AGE	
N	ω	iter	r	iter
11	1.26-1.31	19	0.67-1.07	11
21	1.40-1.42	36	0.62-0.63	15
41	1.66-1.67	63	0.29-0.31	26
81	1.82-1.83	115	0.14	44
161	1.91	210	0.06	74

Table 3.2.4-4: Problem 1 with $\rho = 1$

Method	SOR		AGE	
N	ω	iter	r	iter
10	0.99-1.12	9	1.69-2.91	5
20	1.15-1.27	15	0.67-1.26	8
40	1.42-1.43	26	0.54-0.57	13
80	1.66-1.67	47	0.27	22
160	1.82-1.85	83	0.13	36

Table 3.2.4-5: Problem 1 with $\rho = 7$

Method	SOR		AGE	
N	ω	iter	r	iter
11	1.04-1.10	9	2.67-3.15	7
21	1.21-1.22	15	0.83-1.21	9
41	1.42-1.45	27	0.33-0.56	14
81	1.67	47	0.27	23
161	1.85	83	0.13	39

Table 3.2.4-6: Problem 1 with $\rho = 7$

Method	SOR		AGE	
N	ω	iter	r	iter
10	0.98-1.04	5	13.16-16.02	3
20	1.03-1.04	6	3.66-4.90	4
40	1.10-1.15	10	1.29-1.80	6
80	1.31-1.35	17	0.69-0.70	9
160	1.55-1.58	31	0.28-0.34	16

Table 3.2.4-7: Problem 1 with $\rho = 70$

Method	SOR		AGE	
N	ω	iter	r	iter
11	0.99-1.04	5	19.61-23.40	6
21	1.00-1.09	7	6.26-6.43	6
41	1.12-1.14	10	1.85-2.29	8
81	1.33-1.34	17	0.56-0.76	10
161	1.56-1.57	31	0.29-0.33	16

Table 3.2.4-8: Problem 1 with $\rho = 70$

These results show a substantial gain in terms of the number of iterations of the AGE method over the SOR method. These improvements, are due to the commutative properties of the matrices G_1 and G_2 for the periodic case. We now analyse the estimation of the optimal value of r.

For the case when N is even, the bounds of the eigenvalues of the matrices G_1 and G_2 are $a = g - 1$ and $b = g + 1$, and when N is odd, we have the eigenvalues for the matrix G_1 i.e., 0, g-1, g+1, and for the matrix G_2, the eigenvalues are $\frac{3}{2}g - \frac{1}{2}\sqrt{g^2 + 8}$, g and $\frac{3}{2}g + \frac{1}{2}\sqrt{g^2 + 8}$.

Kellogg and Spanier [1965], has stated that we can choose the smallest nonzero eigenvalue instead of zero, to be the lower bound. In this case, by inspection, the lower bound is $a = g - 1$ and it is obvious that the upper bound is given $b = \frac{3}{2}g + \frac{1}{2}\sqrt{g^2 + 8}$.

The numerical results presented in *Tables* 3.2.4-9 - 3.2.4-14 show some good agreement between the theoretical and experimental values of r, especially for the larger values of N where it can be seen that the optimal r adheres more closely to $r = \sqrt{ab}$.

	The AGE method		
N	r	$\sqrt{ab}$	$(a+b)/2$
10	1.41-1.42	0.659	1.197
20	0.62-0.63	0.318	1.049
40	0.29-0.31	0.158	1.012
80	0.14	0.079	1.003
160	0.06	0.039	1.001

Table 3.2.4-9: Problem 1, $\rho = 1$, $r \in [\sqrt{ab}, (a+b)/2]$

	The AGE method		
N	r	$\sqrt{ab}$	$(a+b)/2$
11	0.67-1.07	0.731	1.718
21	0.62-0.63	0.371	1.560
41	0.29-0.31	0.188	1.516
81	0.14	0.095	1.504
161	0.06	0.048	1.501

Table 3.2.4-10: Problem 1, $\rho = 1$, $r \in [\sqrt{ab}, (a+b)/2]$

	The AGE method		
N	r	$r = \sqrt{ab}$	a
10	1.69-2.91	2.162	1.382
20	0.67-1.26	0.900	0.345
40	0.54-0.57	0.424	0.086
80	0.27	0.209	0.022
160	0.13	0.104	0.005

Table 3.2.4-11: Problem 1, $\rho = 7$, r_{exp}. vs. r_{theory}

	The AGE method		
N	r	$r = \sqrt{ab}$	a
11	2.67-3.15	2.386	1.142
21	0.83-1.21	1.052	0.313
41	0.33-0.56	0.508	0.082
81	0.27	0.253	0.021
161	0.13	0.127	0.050

Table 3.2.4-12: Problem 1, $\rho = 7$, r_{exp}. vs. r_{theory}

	The AGE method		
N	r	$r = \sqrt{ab}$	a
10	13.16-16.02	14.784	13.817
20	3.66-4.90	4.341	3.454
40	1.29-1.80	1.573	0.864
80	0.69-0.70	0.692	0.216
160	0.28-0.34	0.333	0.054

Table 3.2.4-13: Problem 1, $\rho = 70$, r_{exp}. vs. r_{theory}

	The AGE method		
N	r	$r = \sqrt{ab}$	a
11	19.61-23.40	16.896	11.419
21	6.26-6.43	5.222	3.133
41	1.85-2.29	1.905	0.822
81	0.56-0.76	0.840	0.211
161	0.29-0.33	0.406	0.053

Table 3.2.4-14: Problem 1, $\rho = 70$, r_{exp}.vs. r_{theory}

From these results, we may say that the value of $\rho \geq 12$ would give a clear indication that the optimal value of $r = \sqrt{ab}$. When $0 \leq \rho < 12$, we may expect that $r > \sqrt{ab}$ or $r \in [\sqrt{ab}, (a+b)/2]$. Hence, this value can be used as a guide to determine the experimental value of r. This result is in agreement with the result given in Section 3.1.8 when the interval of integration is taken into consideration.

II) The results for Neumann boundary conditions.

Problem 1 - A Linear Problem

$$-U'' + \rho U = (\rho+1)(\sin x + \cos x), \qquad 0 \leq x \leq \pi,$$

$$U'(0) = 1, \quad U'(\pi) = -1.$$

The exact solution is $U(x) = \sin x + \cos x$.

The results for various ρ are tabulated as follows.

Method	SOR		AGE	
$N+1$	ω	iter	r	iter
10	1.516-1.534	20	0.142-0.148	18
20	1.625	34	0.072-0.075	29
40	1.797	65	0.035	58
80	1.929	138	0.017-0.018	115
160	1.965	249	0.008-0.009	227

Table 3.2.4-15: Problem 1 with $\rho = 1$

Method	SOR		AGE	
$N+1$	ω	iter	r	iter
11	1.398	19	0.125-0.155	20
21	1.640	35	0.067-0.069	33
41	1.802	66	0.034-0.035	60
81	1.930	139	0.017	116
161	1.965	251	0.008	228

Table 3.2.4-16: Problem 1 with $\rho = 1$

Method	SOR		AGE	
$N+1$	ω	iter	r	iter
10	1.23-1.24	12	0.83-1.05	10
20	1.50	23	0.26-0.42	17
40	1.72	44	0.13-0.19	31
80	1.81	89	0.06	58
160	1.90	168	0.03	113

Table 3.2.4-17: Problem 1 with $\rho = 5$

Method	SOR		AGE	
$N+1$	ω	iter	r	iter
11	1.28	12	0.79-0.94	11
21	1.52	24	0.24-0.41	18
41	1.72	47	0.12-0.19	32
81	1.81	90	0.06	59
161	1.90	171	0.03	114

Table 3.2.4-18: Problem 1 with $\rho = 5$

Method	SOR		AGE	
$N+1$	ω	iter	r	iter
10	0.99-1.05	6	8.28-8.77	6
20	1.05-1.14	10	2.18-2.65	8
40	1.25-1.33	18	0.68-0.78	10
80	1.51-1.55	33	0.28-0.34	17
160	1.72	61	0.15-0.16	30

Table 3.2.4-19: Problem 1 with $\rho = 70$

Method	SOR		AGE	
$N+1$	ω	iter	r	iter
11	1.01-1.03	6	7.07-7.11	6
21	1.08-1.13	10	2.17-2.39	8
41	1.27-1.33	18	0.60-0.83	11
81	1.52-1.54	33	0.29-0.33	17
161	1.72-1.73	62	0.15	30

Table 3.2.4-20: Problem 1 with $\rho = 70$

Problem 2 - A Linear Problem

$$-U'' + U = -x, \qquad 0 \le x \le 1,$$

$$U'(0) = (1 + 4e - e^2)/(e^2 - 1),$$

$$U'(1) = (e^2 + 3)/(e^2 - 1).$$

The exact solution is $U(x) = \dfrac{2e}{e^2 - 1}(e^x - e^{-x}) - x.$

The results are tabulated in Table 3.2.4-21 and 3.2.4-22.

Method	SOR		AGE	
$N+1$	ω	iter	r	iter
10	1.812	48	0.016-0.018	29
20	1.908	96	0.0084	45
40	1.954	175	0.0041	83
80	1.977	351	0.002	163
160	1.989	661	0.001	323

Table 3.2.4-21: Problem 2

Method	SOR		AGE	
$N+1$	ω	iter	r	iter
11	1.829	53	0.016-0.018	29
21	1.912	102	0.008	45
41	1.955	181	0.004	85
81	1.977	363	0.002	165
161	1.989	668	0.001	322

Table 3.2.4-22: Problem 2

Problem 3 - A Mildly Non-linear Problem

$$-U'' + \frac{3}{2}U^2 = 0, \qquad 0 \le x \le 1,$$

$$U'(0) = -8, \quad U'(1) = -1.$$

The exact solution is $U(x) = \frac{4}{(1+x)^2}$.

The results are tabulated in *Tables* 3.2.4-23 and 3.2.4-24.

Method	SOR		AGE	
$N+1$	ω	iter	r	iter
10	1.592-1.593	22	0.353-0.423	33
20	1.789-1.791	44	0.165-0.184	62
40	1.894	79	0.084-0.086	117
80	1.947	145	0.041-0.044	227
160	1.974	274	0.022	440

Table 3.2.4-23: Problem 3

Method	SOR		AGE	
$N+1$	ω	iter	r	iter
10	1.623-1.628	25	0.348-0.381	37
20	1.799-1.804	46	0.154-0.173	65
40	1.897	83	0.080-0.085	120
80	1.948	150	0.041-0.043	229
160	1.974	270	0.022	442

Table 3.2.4-24: Problem 3

Problem 4 - A Linear Problem

$$U'' + xU' - U = xe^x \qquad 0 \le x \le 1$$

$$U'(0) = 2, \quad U'(1) = 1 + e$$

The exact solution is $U(x) = x + e^x$.

The results are tabulated in *Tables* 3.2.4-25 and 3.2.4-26.

Method	SOR		AGE	
$N+1$	ω	iter	r	iter
10	1.809	52	0.248-0.327	62
20	1.906	100	0.141-0.150	114
40	1.953	188	0.069-0.072	226
80	1.9769	361	0.034-0.040	456
160	1.988	770	0.021	902

Table 3.2.4-25: Problem 4

Method	SOR		AGE	
$N+1$	ω	iter	r	iter
11	1.826	58	0.253-0.278	63
21	1.910	108	0.131-0.114	119
41	1.954	195	0.069-0.073	232
81	1.977	372	0.036-0.040	461
161	1.988	789	0.021	908

Table 3.2.4-26: Problem 4

The AGE method again, appears to be competitive than the SOR method, especially when the problem is linear and the matrix A derived is strongly diagonally dominant. However, for the nonlinear problems, the AGE method again needs more iterations.

The estimation for the optimal value of r may be determined as follows. For the linear problems governed by Neumann boundary conditions, the relation $r = \sqrt{ab}$ will only be satisfied when ρ is large enough. In many cases, however, as shown in the tables below, the value of r lie in the interval $[a,\sqrt{ab}]$, where $a = g-1$, and $b = \frac{3}{2}g + \frac{1}{2}\sqrt{g^2 + 8}$.

	The AGE method		
$N+1$	r	a	$\sqrt{ab}$
11	0.125-0.155	0.049	0.390
21	0.067-0.069	0.012	0.193
41	0.034-0.035	0.003	0.096
81	0.017	0.001	0.048
161	0.008	0.000	0.024

Table 3.2.4-27: Problem 1, $\rho = 1$, $r \in [a,\sqrt{ab}]$

	The AGE method		
$N+1$	r	a	$\sqrt{ab}$
10	0.83-1.05	0.305	1.035
20	0.26-0.42	0.068	0.461
40	0.13-0.19	0.016	0.222
80	0.06	0.004	0.109
160	0.03	0.001	0.054

Table 3.2.4-28: Problem 1, $\rho = 5$, $r \in [a,\sqrt{ab}]$

	The AGE method		
$N+1$	r	a	$\sqrt{ab}$
11	7.07-7.11	3.454	5.674
21	2.17-2.39	0.864	1.969
41	0.60-0.83	0.216	0.852
81	0.29-0.33	0.054	0.408
161	0.15	0.013	0.202

Table 3.2.4-29: Problem 1, $\rho = 70$, $r \in [a,\sqrt{ab}]$

	The AGE method			
$N+1$	r	a	b	$\sqrt{ab}$
11	17.30-20.05	9.870	21.920	14.709
21	4.90-5.88	2.467	7.438	4.284
41	1.54-1.93	0.617	4.054	1.581
81	0.46-0.58	0.154	3.259	0.709
161	0.23-0.24	0.039	3.064	0.344

Table 3.2.4-30: Problem 1, $\rho = 200$

	The AGE method		
$N+1$	r	a	$\sqrt{ab}$
11	0.015-0.016	0.005	0.123
21	0.008	0.001	0.061
41	0.004	0.000	0.031
81	0.002	0.000	0.015
161	0.001	0.000	0.008

Table 3.2.4-31: Problem 2, $r \in [a,\sqrt{ab}]$

From these results, we conclude that, for $0 < \rho < 5$, we may use the interval $[a,\sqrt{ab}]$ to estimate the experimental value of r, otherwise use $[\sqrt{ab},b]$ for $\rho \geq 5$. For other problems, when g is variable, we may consider any value of r in the interval $[a,b]$.

III) The results for combined boundary conditions.

Problem 1 - A Linear Problem

$$-U'' + \rho U = (\rho+1)(\sin x + \cos x), \qquad 0 \leq x \leq \pi$$

$$U(0) = 1, \; U'(\pi) = -1.$$

The exact solution is $U(x) = \sin x + \cos x$.

The results for various ρ are tabulated as follows.

Method	SOR		AGE	
N	ω	iter	r	iter
10	1.742	35	0.288-0.291	38
20	1.863	65	0.149-0.151	73
40	1.929	134	0.076-0.077	141
80	1.964	270	0.039	277
160	1.982	546	0.020	521

Table 3.2.4-32: Problem 1 with $\rho = 0$

Method	SOR		AGE	
N	ω	iter	r	iter
11	1.762	39	0.225-0.286	44
21	1.869	68	0.143-0.153	83
41	1.931	135	0.081-0.083	157
81	1.964	276	0.044	307
161	1.982	549	0.023-0.024	605

Table 3.2.4-33: Problem 1 with $\rho = 0$

Method	SOR		AGE	
N	ω	iter	r	iter
10	1.50-1.51	20	0.41-0.51	19
20	1.72	35	0.21-0.25	35
40	1.85	65	0.12	63
80	1.92	127	0.06	123
160	1.960	227	0.034	235

Table 3.2.4-34: Problem 1 with $\rho = 1$

Method	SOR		AGE	
N	ω	iter	r	iter
11	1.539-1.546	21	0.419-0.462	20
21	1.729-1.732	37	0.225-0.256	38
41	1.853	67	0.134-0.138	72
81	1.923	122	0.073-0.076	142
161	1.961	222	0.041	280

Table 3.2.4-35: Problem 1 with $\rho = 1$

Method	SOR		AGE	
N	ω	iter	r	iter
10	1.25	11	0.75-1.06	11
20	1.50	21	0.27-0.28	16
40	1.70	41	0.11-0.18	31
80	1.84	81	0.05-0.09	61
160	1.91-1.92	161	0.04-0.06	121

Table 3.2.4-36: Problem 1 with $\rho = 5$

Method	SOR		AGE	
N	ω	iter	r	iter
11	1.28-1.30	13	0.76-0.87	11
21	1.51-1.52	23	0.39-0.42	18
41	1.71	42	0.22-0.23	34
81	1.84	82	0.122-0.128	67
161	1.91-1.92	162	0.07	133

Table 3.2.4-37: Problem 1 with $\rho = 5$

Method	SOR		AGE	
N	ω	iter	r	iter
10	1.01-1.03	6	6.96-7.15	6
20	1.06-1.13	10	2.04-2.43	8
40	1.24-1.33	18	0.67-0.78	10
80	1.49-1.54	33	0.29-0.33	17
160	1.70-1.72	61	0.13	31

Table 3.2.4-38: Problem 1 with $\rho = 70$

Method	SOR		AGE	
N	ω	iter	r	iter
11	0.98-1.08	7	5.38-6.92	7
21	1.13	10	1.92-2.20	8
41	1.27-1.33	18	0.66-0.74	10
81	1.50-1.53	33	0.22-0.37	19
161	1.71	61	0.10-0.11	36

Table 3.2.4-39: Problem 1 with $\rho = 70$

Problem 2 - A Linear Problem

$$-U'' + U = -x, \qquad 0 \le x \le 1$$

$$U(0) = 0, \; U'(1) = (e^2 + 3)/(e^2 - 1).$$

The exact solution is $U(x) = \frac{2e}{e^2 - 1}(e^x - e^{-x}) - x$.

The results are tabulated in Table 3.2.4-40 and 3.2.4-41.

Method	SOR		AGE	
N	ω	iter	r	iter
10	1.699-1.703	31	0.310-0.316	30
20	1.835-1.841	61	0.160-0.163	58
40	1.918	105	0.081-0.083	114
80	1.957	204	0.043	225
160	1.979	370	0.022-0.023	442

Table 3.2.4-40: Problem 2

Method	SOR		AGE	
N	ω	iter	r	iter
11	1.722-1.726	34	0.283-0.291	33
21	1.842-1.848	64	0.152-0.156	61
41	1.918	107	0.079-0.082	117
81	1.958	202	0.042	228
161	1.979	375	0.022	444

Table 3.2.4-41: Problem 2

Problem 3 - A Mildly Non-linear Problem

$$-U'' + \frac{3}{2}U^2 = 0, \qquad 0 \le x \le 1$$

$$U(0) = 4, \quad U'(1) = -1.$$

The exact solution is $U(x) = \frac{4}{(1+x)^2}$.

The results are tabulated in Table 3.2.4-42 and 3.2.4-43.

Method	SOR		AGE	
N	ω	iter	r	iter
10	1.584-1.595	22	0.387-0.413	27
20	1.769-1.782	42	0.194-0.205	52
40	1.877-1.887	82	0.097-0.100	100
80	1.935-1.941	162	0.050-0.051	197
160	1.966-1.970	332	0.026	386

Table 3.2.4-42: Problem 3

Method	SOR		AGE	
N	ω	iter	r	iter
11	1.615-1.626	24	0.387-0.413	30
21	1.779-1.792	44	0.194-0.205	56
41	1.879-1.889	84	0.097-0.100	112
81	1.936-1.942	164	0.050-0.051	219
161	1.966-1.970	334	0.10.0266	429

Table 3.2.4-43: Problem 3

Problem 4a - A Linear Problem

$$U'' + xU' - U = xe^x \qquad 0 \le x \le 1,$$

$$U(0) = 1,\ U'(1) = 1 + e$$

The exact solution is $U(x) = x + e^x$.

The results are tabulated in Table 3.2.4-44 and 3.2.4-45.

Method	SOR		AGE	
N	ω	iter	r	iter
10	1.719	32	0.603-0.609	31
20	1.848	61	0.312	60
40	1.921-1.922	122	0.159-0.161	120
80	1.959-1.961	242	0.079-0.082	239
160	1.979-1.982	482	0.042	474

Table 3.2.4-44: Problem 4a

Method	SOR		AGE	
N	ω	iter	r	iter
11	1.741	34	0.546-0.557	34
21	1.854	64	0.298	63
41	1.923-1.924	125	0.155-0.158	123
81	1.961-1.962	245	0.078-0.082	242
161	1.979-1.982	485	0.043	474

Table 3.2.4-45: Problem 4a

Problem 4b - A Linear Problem

$$U'' + xU' - U = xe^x \qquad 0 \le x \le 1,$$

$$U'(0) = 2,\ U(1) = 1 + e$$

The exact solution is $U(x) = x + e$. Set $h = 1/(N+1)$.

The results are tabulated in Table 3.2.4-46 and 3.2.4-47.

Method	SOR		AGE	
N	ω	iter	r	iter
9	1.652	29	0.715-0.735	29
19	1.819	58	0.366-0.373	59
39	1.907-1.908	118	0.186-0.188	120
79	1.953	238	0.101	261
159	1.976	478	0.053	517

Table 3.2.4-46: Problem 4b

Method	SOR		AGE	
N	ω	iter	r	iter
12	1.728	37	0.527-0.558	37
22	1.842	67	0.297-0.309	65
42	1.914	127	0.162-0.166	121
82	1.955	247	0.086	234
162	1.977	487	0.045	458

Table 3.2.4-47: Problem 4b

The results have shown that the AGE method has a competitive edge over the SOR method in solving the linear problems governed by combined boundary conditions, especially when the matrix A is strongly diagonally dominant. For other problems, the AGE method needs slightly more iterations for convergence. This is due to the fact that one of the boundaries is prescribed in derivatives which will allow the errors to propagate towards this boundary and be reflected. Consequently, this will slow down the rate of convergence, as shown by the experimental results.

As these problems have one end point prescribed in derivative form, we would expect that the bounds of the eigenvalues be similar to the previous one, i.e., $a = g - 1$ and $b = \frac{3}{2}g + \frac{1}{2}\sqrt{g^2 + 8}$. As a result, the optimal value of r will either be within the interval $[a, \sqrt{ab}]$ or $[\sqrt{ab}, b]$. *Tables* 3.2.4-48 - 3.2.4-54 confirm these arguments.

	The AGE method			
N	r	$\sqrt{ab}$	a	$(a+b)/2$
10	0.41-0.51	0.390	0.049	1.566
20	0.21-0.25	0.193	0.012	1.516
40	0.12	0.096	0.003	1.504
80	0.06	0.048	0.001	1.501
160	0.034	0.024	0.000	1.500

Table 3.2.4-48: Problem 1, $\rho = 1$, $r \in [\sqrt{ab}, (a+b)/2]$

	The AGE method			
N	r	$\sqrt{ab}$	a	$(a+b)/2$
11	0.42-0.46	0.354	0.041	1.554
21	0.23-0.25	0.184	0.011	1.515
41	0.13-0.14	0.094	0.003	1.504
81	0.07	0.048	0.001	1.501
161	0.04	0.024	0.000	1.500

Table 3.2.4-49: Problem 1, $\rho = 1$, $r \in [\sqrt{ab}, (a+b)/2]$

	The AGE method	
N	r	$\sqrt{ab}$
10	0.75-1.06	0.918
20	0.27-0.28	0.438
40	0.11-0.18	0.216
80	0.05-0.09	0.108
160	0.04-0.06	0.054

Table 3.2.4-50: Problem 1, $\rho = 5$, $r \approx \sqrt{ab}$

	The AGE method	
N	r	$\sqrt{ab}$
11	0.76-0.87	0.826
21	0.39-0.42	0.416
41	0.22-0.23	0.211
81	0.122-0.128	0.106
161	0.07	0.053

Table 3.2.4-51: Problem 1, $\rho = 5$, $r \approx \sqrt{ab}$

	The AGE method		
N	r	$\sqrt{ab}$	a
10	1.36-1.62	1.376	0.439
20	0.35-0.60	0.629	0.123
40	0.19	0.307	0.031
80	0.09-0.10	0.152	0.008
160	0.05	0.076	0.002

Table 3.2.4-52: Problem 1, $\rho = 10$, $r \in [a, \sqrt{ab}]$

	The AGE method			
N	r	$\sqrt{ab}$	a	b
11	5.38-6.92	4.830	2.855	2.855
21	1.92-2.20	1.845	0.783	4.347
41	0.66-0.74	0.829	0.205	3.346
81	0.22-0.37	0.403	0.053	3.088
161	0.10-0.11	0.201	0.013	3.022

Table 3.2.4-53: Problem 1, $\rho = 70$, $r \in [a, \sqrt{ab}]$

	The AGE method			
N	r	$\sqrt{ab}$	a	$(a+b)/2$
11	0.283-0.291	0.111	0.004	1.506
21	0.152-0.156	0.058	0.001	1.503
41	0.079-0.082	0.030	0.000	1.500
81	0.042	0.015	0.000	1.500
161	0.022	0.008	0.000	1.500

Table 3.2.4-54: Problem 2, $r \in [\sqrt{ab}, (a+b)/2]$

The results for Problem 1 show some agreement with the same problem governed by Neumann boundary conditions. It is found that when $\rho = 5$, the value of r agrees with the theoretical value given by $\sqrt{ab}$. When $0 < \rho < 5$, the experimental r falls within $[\sqrt{ab}, (a+b)/2]$. For larger ρ, i.e., $\rho > 5$, the value of r tends to divert to the interval $[\sqrt{ab}, b]$ especially for a smaller number of points. In other cases, the value of r is within the interval $[a, \sqrt{ab}]$. For the other problems, it can be shown that the value of r lies in the interval $[\sqrt{ab}, (a+b)/2]$.

3.3 On improving the accuracy of the AGE method

We have already shown in Section 3.1 and 3.2, that the finite difference approximations are successfully applied to solve the two-point boundary value problem (3.1.1-1) subject to different boundary conditions. With this approximation, the truncation error is usually of order $O(h^2)$. This error, however, can be made smaller by using a more accurate formula, i.e., Numerov's formula. With this formula, the accuracy of the solution is greater since the truncation error of the Numerov formula is of order $O(h^4)$.

In this section, we shall investigate the application of the Numerov formula on the two-point boundary value problem subject to the Dirichlet boundary conditions. For comparison, we commence with the discussion of the replacement of the problem by the usual finite difference equations.

3.3.1 The standard (2nd order) finite difference formula

Let consider a more general two-point boundary value problem

$$-U'' + q(x)U = f(x), \qquad a \le x \le b \tag{3.3.1-1}$$

subject to the boundary condition

$$U(a) = \alpha, \; U(b) = \beta. \tag{3.3.1-2}$$

Here, α, β are given real constants, and $f(x)$ and $q(x)$ are given real continuous functions on $a \le x \le b$, with $q(x) \ge 0$. We place a uniform mesh of size,

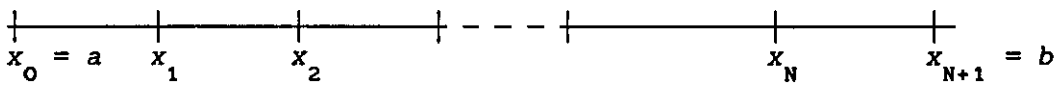
$$h = (b-a)/(N+1) \tag{3.3.1-3}$$

on the interval $a \le x \le b$, and we denote the mesh points of the discrete problem by

$$x_i = a + i(b-a)/(N+1)$$

$$= a + ih, \qquad 0 \le i \le N+1 \qquad (3.3.1\text{-}4)$$

as illustrated



By applying the centred finite difference approximation, for a small h, the equation (3.3.1-1) can now be written as

$$-\frac{u_{i-1} - 2u_i + u_{i+1}}{h^2} + q_i u_i = f_i \qquad (3.3.1\text{-}5)$$

or simplified to

$$-u_{i-1} + 2g_i u_i - u_{i+1} = h^2 f_i, \qquad 1 \le i \le N \qquad (3.3.1\text{-}6)$$

where $g_i = 1 + \frac{1}{2}q_i h^2$. On the boundaries we have $u_0 = \alpha$, $u_{N+1} = \beta$.

The equation (3.3.1-6) can be written in the matrix form

$$\mathbf{A}\mathbf{u} = \mathbf{b} \qquad (3.3.1\text{-}7)$$

where

$$A = \begin{bmatrix} 2g_1 & -1 & & & \\ -1 & 2g_2 & -1 & & 0 \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2g_{N-1} & -1 \\ 0 & & & -1 & 2g_N \end{bmatrix}$$

$$\mathbf{u} = [u_1, u_2, \ldots, u_{N-1}, u_N]^T,$$

and $$\mathbf{b} = [\alpha+h^2 f_1, h^2 f_2, \ldots, h^2 f_{N-1}, \beta+h^2 f_N]^T.$$

Now, split the matrix A into two submatrices

$$A = G_1 + G_2 \qquad (3.3.1\text{-}8)$$

where

$$G_1 = \begin{bmatrix} g_1 & -1 & & & & \\ -1 & g_2 & & & & \\ & & g_3 & -1 & & \\ & & -1 & g_4 & & \\ & & & & \ddots & \\ & & & & g_{N-1} & -1 \\ & & & & -1 & g_N \end{bmatrix} \qquad (3.3.1\text{-}9)$$

$$G_2 = \begin{bmatrix} g_1 & & & & & \\ & g_2 & -1 & & & \\ & -1 & g_3 & & & \\ & & & \ddots & & \\ & & & g_{N-2} & -1 & \\ & & & -1 & g_{N-1} & \\ & & & & & g_N \end{bmatrix} \qquad (3.3.1\text{-}10)$$

if N is even, and

$$G_1 = \begin{bmatrix} g_1 & -1 & & & \\ -1 & g_2 & & & \\ & & \ddots & & \\ & & g_{N-2} & -1 & \\ & & -1 & g_{N-1} & \\ & & & & g_N \end{bmatrix} \qquad (3.3.1\text{-}11)$$

$$G_2 = \begin{bmatrix} g_1 & & & & \\ & g_2 & -1 & & \\ & -1 & g_3 & & \\ & & & \ddots & \\ & & & g_{N-1} & -1 \\ & & & -1 & g_N \end{bmatrix} \qquad (3.3.1\text{-}12)$$

if N is odd.

Thus, (3.3.1-7) becomes

$$(G_1 + G_2)\mathbf{u} = \mathbf{b}. \tag{3.3.1-13}$$

Evidently, G_1 and G_2 are symmetric and positive definite. Thus, for any $r > 0$, $(rI + G_1)$ and $(rI + G_2)$ are also symmetric and positive definite. Now, by applying AGE method, $\mathbf{u}^{(k+1)}$ can be determined in two sweeps, i.e.,

$$(rI + G_1)\mathbf{u}^{(k+1/2)} = \mathbf{b} + (rI - G_2)\mathbf{u}^{(k)} \tag{3.3.1-14}$$

$$(rI + G_2)\mathbf{u}^{(k+1)} = \mathbf{b} + (rI - G_1)\mathbf{u}^{(k+1/2)} \tag{3.3.1-15}$$

or explicitly as

$$\mathbf{u}^{(k+1/2)} = (rI + G_1)^{-1}[\mathbf{b} + (rI - G_2)\mathbf{u}^{(k)}] \tag{3.3.1-16}$$

$$\mathbf{u}^{(k+1)} = (rI + G_2)^{-1}[\mathbf{b} + (rI - G_1)\mathbf{u}^{(k+1/2)}]. \tag{3.3.1-17}$$

The iteration matrix is given by

$$T_r = (rI + G_2)^{-1}(rI - G_1)(rI + G_1)^{-1}(rI - G_2). \tag{3.3.1-18}$$

By a similarity transformation, we then have

$$\begin{aligned}\tilde{T}_r &= (rI + G_2)T_r(rI + G_2)^{-1} \\ &= (rI - G_1)(rI + G_1)^{-1}(rI - G_2)(rI + G_2)^{-1}.\end{aligned} \tag{3.3.1-19}$$

Since G_1 and G_2 are positive definite, then their eigenvalues are positive. Moreover, since G_1 and G_2 are symmetric, then

$$\begin{aligned}\|(rI - G_1)(rI + G_1)^{-1}\| &= \rho((rI - G_1)(rI + G_1)^{-1}) \\ &= \max_{\mu} \frac{|r - \mu|}{|r + \mu|} < 1\end{aligned} \tag{3.3.1-20}$$

and $\|(rI - G_2)(rI + G_2)^{-1}\| = \rho((rI - G_2)(rI + G_2)^{-1})$

$$= \max_{\nu} \frac{|r - \nu|}{|r + \nu|} < 1. \tag{3.3.1-21}$$

Thus, from (3.3.1-20) and (3.3.1-21), we then have $\rho(\tilde{T}_r) = \rho(T_r) < 1$. Hence, the AGE method is convergent.

It is obvious that the (2X2) submatrices of $(rI + G_1)$, $(rI + G_2)$, $(rI - G_1)$ and $(rI - G_2)$, are respectively of the form

$$\hat{G} = \begin{bmatrix} \alpha_i & -1 \\ -1 & \alpha_{i+1} \end{bmatrix} \text{ and } \tilde{G} = \begin{bmatrix} \beta_i & 1 \\ 1 & \beta_{i+1} \end{bmatrix}$$

where $\alpha_i = r + g_i$, $\beta_i = r - g_i$, and the inverse of $\hat{G}$ is given by

$$\hat{G}^{-1} = d_i \begin{bmatrix} \alpha_{i+1} & 1 \\ 1 & \alpha_i \end{bmatrix}$$

where $d_i = 1/(\alpha_i \alpha_{i+1} - 1)$, $i = 1, 2, \ldots, N/2$.

By using equations (3.3.1-16) and (3.3.1-17), we now write the algorithm for the AGE method for N even with G_1 and G_2 in (3.3.1-9) and (3.3.1-10) to compute $\mathbf{u}^{(k+1/2)}$ and $\mathbf{u}^{(k+1)}$. In computation, we need the arrays to store the values of g_i, α_i and β_i. The values d_i can be computed as a variable.

Algorithm 3.3.1-1: The AGE method for the model problem (3.3.1-1).

Set $u_i^{(k)} = 0$, $i = 0, \ldots, N+1$.

Step 1. To compute $\mathbf{u}^{(k+1/2)}$. Set $i = 1$.

while $i \le N-1$, compute

$$r_1 = b_i + u_{i-1}^{(k)} + \beta_i u_i^{(k)}$$

$$r_2 = b_{i+1} + \beta_{i+1} u_{i+1}^{(k)} + u_{i+2}^{(k)}$$

$$d = 1/(\alpha_i \alpha_{i+1} - 1)$$

$$u_i^{(k+1/2)} = (\alpha_{i+1} r_1 + r_2) d$$

$$u_{i+1}^{(k+1/2)} = (r_1 + \alpha_i r_2) d$$

$$i = i + 2.$$

Step 2. To compute $\mathbf{u}^{(k+1)}$. Set $i = 2$.

$$u_1^{(k+1)} = (b_1 + \beta_1 u_1^{(k+1/2)} + u_2^{(k+1/2)})/\alpha_1$$

while $i \le N-2$, compute

$$r_1 = b_i + u_{i-1}^{(k+1/2)} + \beta_i u_i^{(k+1/2)}$$

$$r_2 = b_{i+1} + \beta_{i+1} u_{i+1}^{(k+1/2)} + u_{i+2}^{(k+1/2)}$$

$$d = 1/(\alpha_i \alpha_{i+1} - 1)$$

$$u_i^{(k+1)} = (\alpha_{i+1} r_1 + r_2) d$$

$$u_{i+1}^{(k+1)} = (r_1 + \alpha_i r_2) d$$

$$i = i + 2.$$

$$u_N^{(k+1)} = (b_N + u_{N-1}^{(k+1/2)} + \beta_N u_N^{(k+1/2)})/\alpha_N$$

Step 3. Repeat *Step* 1 and *Step* 2 until convergence is achieved.

3.3.2 The Numerov-AGE formula

To achieve greater accuracy in the numerical solution, one can use the well known recurrence solution called the Numerov formula which is given by

$$u_{i-1} - 2u_i + u_{i+1} = \frac{1}{12} h^2 (u''_{i-1} + 10u''_i + u''_{i+1}) \qquad (3.3.2\text{-}1)$$

where $u''_i = U''(x_i)$, to replace equation (3.3.1-1).

The finite difference equation (3.3.2-1) has been shown to have the truncation error of $O(h^4)$. Thus, it is obviously superior to the $O(h^2)$ accuracy of the finite difference formula used in equation (3.3.1-5). This equation is then used at each point $1 \le i \le N$ in the interval (a,b) to construct the matrix A and its constituent component G_1 and G_2 which form the AGE method.

Now, rewrite equation (3.3.1-1) as

$$U'' = q(x)U - f(x). \qquad (3.3.2\text{-}2)$$

Let $u''_i = U''(x_i)$, $q_i = q(x_i)$ and $f_i = f(x_i)$, we then have

$$u''_{i-1} = q_{i-1} u_{i-1} - f_{i-1}, \quad u''_i = q_i u_i - f_i,$$

and $$u''_{i+1} = q_{i+1} u_{i+1} - f_{i+1}. \qquad (3.3.2\text{-}3)$$

By substituting (3.3.2-3) in (3.3.2-1), we have the new equation

$$u_{i-1} - 2u_i + u_{i+1} = \frac{1}{12}h^2[q_{i-1}u_{i-1} - f_{i-1} + 10(q_iu_i - f_i) + q_{i+1}u_{i+1} - f_{i+1}]$$

or

$$-u_{i-1} + 2u_i - u_{i+1} = -\frac{1}{12}h^2[q_{i-1}u_{i-1} - f_{i-1} + 10(q_iu_i - f_i) + q_{i+1}u_{i+1} - f_{i+1}] \qquad (3.3.2\text{-}4)$$

which simplifies to

$$s_iu_{i-1} + 2g_iu_i + t_iu_{i+1} = w_i \qquad (3.3.2\text{-}5)$$

where $s_i = -1 + \frac{1}{12}h^2q_i$, $g_i = 1 + \frac{5}{12}h^2q_i$, $t_i = -1 + \frac{1}{12}h^2q_i$,

and $w_i = \frac{1}{12}h^2(f_{i-1} + 10f_i + f_{i+1})$, for $1 \le i \le N$.

The equation (3.3.2-5) can be written in the matrix form $A\mathbf{u} = \mathbf{b}$, where

$$\mathbf{u} = [u_1, u_2, \ldots, u_{N-1}, u_N]^T, \quad \mathbf{b} = [w_1 - s_1, w_2, \ldots, w_{N-1}, w_N - t_N]^T,$$

and the matrix A is given by

$$A = \begin{bmatrix} 2g_1 & t_1 & & & \\ s_2 & 2g_2 & t_2 & & \\ & \ddots & \ddots & \ddots & \\ & & s_{N-1} & 2g_{N-1} & t_{N-1} \\ & 0 & & s_N & 2g_N \end{bmatrix}.$$

Now, split the matrix A into two submatrices $A = G_1 + G_2$, where

$$G_1 = \left[\begin{array}{cc|cc|cc} g_1 & t_1 & & & & \\ s_2 & g_2 & & & & \\ \hline & & g_3 & t_3 & & \\ & & s_4 & g_4 & & \\ \hline & & & & \ddots & \\ \hline & & & & g_{N-1} & t_{N-1} \\ & & & & s_N & g_N \end{array}\right] \qquad (3.3.2\text{-}6)$$

$$G_2 = \begin{bmatrix} g_1 & & & & \\ & g_2 & t_2 & & \\ & s_3 & g_3 & & \\ & & & & \\ & & & g_{N-2} & t_{N-2} \\ & & & s_{N-1} & g_{N-1} \end{bmatrix} \qquad (3.3.1\text{-}7)$$

if N is even, and

$$G_1 = \begin{bmatrix} g_1 & t_1 & & & \\ s_2 & g_2 & & & \\ & & & & \\ & & g_{N-2} & t_{N-2} & \\ & & s_{N-1} & g_{N-1} & \\ & & & & g_N \end{bmatrix} \qquad (3.3.2\text{-}8)$$

$$G_2 = \begin{bmatrix} g_1 & & & & \\ & g_2 & t_2 & & \\ & s_3 & g_3 & & \\ & & & & \\ & & & g_{N-1} & t_{N-1} \\ & & & s_N & g_N \end{bmatrix} \qquad (3.3.2\text{-}9)$$

if N is odd.

Evidently, G_1 and G_2 are unsymmetric and consists of (2X2) block submatrices of the form

$$G^* = \begin{bmatrix} g_1 & t_1 \\ s_{1+1} & g_{1+1} \end{bmatrix}. \qquad (3.3.2\text{-}10)$$

It can be shown for the unsymmetric matrices, the AGE method is convergent provided that all the eigenvalues of the matrices are positive.

The eigenvalues of G_1 and G_2 are $\lambda_1 = g_1$ and which are given by the determinantal equation derived from (3.3.2-10), i.e.,

$$det \begin{vmatrix} \lambda - g_1 & t_1 \\ s_{1+1} & \lambda - g_{1+1} \end{vmatrix} = 0$$

which simplifies to

$$\lambda^2 - \lambda(g_1 + g_{1+1}) + g_1 g_{1+1} - s_{1+1} t_1 = 0,$$

which has the roots

$$\lambda_2 = \frac{1}{2}(g_1 + g_{1+1}) - \frac{1}{2}\sqrt{(g_1 - g_{1+1})^2 + 4 s_{1+1} t_1}$$

and $$\lambda_3 = \frac{1}{2}(g_1 + g_{1+1}) + \frac{1}{2}\sqrt{(g_1 - g_{1+1})^2 + 4 s_{1+1} t_1}\ .$$

Both λ_1 and λ_3 are positive and we will show that λ_2 is also positive. This can be shown as follows.

$$\frac{1}{2}(g_1 + g_{1+1}) = \frac{1}{2}(1 + \frac{5}{12}h^2 q_1 + 1 + \frac{5}{12}h^2 q_{1+1}) = 1 + \frac{5}{24}h^2(q_1 + q_{1+1}).$$

$$(g_1 - g_{1+1})^2 = [\frac{5}{12}h^2(q_1 - q_{1+1})]^2 = \frac{25}{144}h^4(q_1 - q_{1+1})^2.$$

$$4 s_{1+1} t_1 = 4(-1 + \frac{1}{12}h^2 q_{1+1})(-1 + \frac{1}{12}h^2 q_1)$$

$$= 4 - \frac{1}{3}h^2(q_1 + q_{1+1}) + \frac{1}{36}h^4 q_1 q_{1+1}.$$

By neglecting all the terms which contain h^4, then we have

$$\frac{1}{2}\sqrt{(g_1 - g_{1+1})^2 + 4 s_{1+1} t_1} = \frac{1}{2}\sqrt{4 - \frac{1}{3}h^2(q_1 + q_{1+1})} < 1.$$

by (3.3.1-20) and (3.3.1-21)

Hence, λ_2 is positive. Hence, the AGE method is convergent.

Now, by using the AGE method, we determine $\mathbf{u}^{(k+1)}$ in two steps from the explicit form of (3.3.1-16) and (3.3.1-17).

Let us consider N even. For any $r > 0$, the (2X2) block submatrices $(rI + G_1)$, $(rI + G_2)$, $(rI - G_1)$ and $(rI - G_2)$ have the form of

$$\hat{G} = \begin{bmatrix} \alpha_1 & t_1 \\ s_{1+1} & \alpha_{1+1} \end{bmatrix} \text{ and } \tilde{G} = \begin{bmatrix} \beta_1 & -t_1 \\ -s_{1+1} & \beta_{1+1} \end{bmatrix}$$

where $\alpha_1 = r + g_1$ and $\beta_1 = r - g_1$.

The inverse of $\hat{G}$ is given by

$$\hat{G}^{-1} = d_i \begin{bmatrix} \alpha_{i+1} & -t_i \\ -s_{i+1} & \alpha_i \end{bmatrix}, \text{ where } d_i = \frac{1}{\alpha_i \alpha_{i+1} - s_{i+1} t_i}, \quad i = 1, \ldots, N/2.$$

In programming the AGE method, we need to store the arrays g_i, α_i, β_i, t_i and s_i, but not d_i as this value may be assigned as a variable.

We now write the algorithm for the Numerov-AGE formula.

Algorithm 3.3.2-1: The Numerov-AGE method the model problem (3.3.2-1).

Set $u_i^{(k)} = 0$, $i = 0, \ldots, N+1$, $s_1 = 0$, $t_N = 0$.

Step 1. To compute $\mathbf{u}^{(k+1/2)}$. Set $i = 1$.

while $i \le N-1$, compute

$$r_1 = b_i - s_i u_{i-1}^{(k)} + \beta_i u_i^{(k)}$$
$$r_2 = b_{i+1} + \beta_{i+1} u_{i+1}^{(k)} - t_{i+1} u_{i+2}^{(k)}$$
$$d = 1/(\alpha_i \alpha_{i+1} - s_{i+1} t_i)$$
$$u_i^{(k+1/2)} = (\alpha_{i+1} r_1 - t_i r_2) d$$
$$u_{i+1}^{(k+1/2)} = (-s_{i+1} r_1 + \alpha_i r_2) d$$
$$i = i + 2.$$

Step 2. To compute $\mathbf{u}^{(k+1)}$. Set $i = 2$.

$$u_1^{(k+1)} = (b_1 + \beta_1 u_1^{(k+1/2)} - t_1 u_2^{(k+1/2)})/\alpha_1$$

while $i \le N-2$, compute

$$r_1 = b_i - s_i u_{i-1}^{(k+1/2)} + \beta_i u_i^{(k+1/2)}$$
$$r_2 = b_{i+1} + \beta_{i+1} u_{i+1}^{(k+1/2)} - t_{i+1} u_{i+2}^{(k+1/2)}$$
$$d = 1/(\alpha_i \alpha_{i+1} - s_{i+1} t_i)$$
$$u_i^{(k+1)} = (\alpha_{i+1} r_1 - t_i r_2) d$$
$$u_{i+1}^{(k+1)} = (-s_{i+1} r_1 + \alpha_i r_2) d$$
$$i = i + 2.$$

$$u_N^{(k+1)} = (b_N - s_N u_{N-1}^{(k+1/2)} + \beta_N u_N^{(k+1/2)})/\alpha_N$$

Step 3. Repeat *Step* 1 and *Step* 2 until convergence is achieved.

3.3.3 Experimental results

Two problems were considered, the first problem is linear while the second is non-linear. The linear problem yields a symmetric matrix, but the non-linear problem gives an unsymmetric matrix. Both problems converge to the true solution, but the main concern in this section is achieving the gain in accuracy of the solution. To compare the accuracy of the solution, the principal truncation errors, E_{pte} is considered. In addition, the absolute error, E_{abs} is also included in the results.

The respective E_{pte} for the finite difference (FD) formula and the Numerov (NU) formula is given by

$$E_{pte}^{FD} = \frac{h^2}{12}\frac{d^4U}{dx^4} \qquad \text{and} \qquad E_{pte}^{NU} = \frac{h^4}{240}\frac{d^6U}{dx^6}.$$

Problem 1 - A Linear Problem.

$$-U'' + U = -x, \qquad 0 \le x \le 1,$$

$$U(0) = 0, \; U(1) = 1.$$

The exact solution to this problem is given by

$$U(x) = \frac{2e}{e^2 - 1}(e^x - e^{-x}) - x.$$

The respective matrix A derived from the finite difference approximation and Numerov's formula are given by

$$A_{FD} = \begin{bmatrix} 2g & -1 & & & 0 \\ -1 & 2g & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2g & -1 \\ 0 & & & -1 & 2g \end{bmatrix} \quad \text{and} \quad A_{NU} = \begin{bmatrix} 2g & -c & & & 0 \\ -c & 2g & -c & & \\ & \ddots & \ddots & \ddots & \\ & & -c & 2g & -c \\ 0 & & & -c & 2g \end{bmatrix}$$

where $g = 1 + \frac{1}{2}h^2$.

The respective E_{pte} for Problem 1 with FD approximation and NU method are as follows.

$$E_{pte}^{FD} = \frac{h^2}{12}(x + U) \qquad \text{and} \qquad E_{pte}^{NU} = \frac{h^4}{240}(x + U).$$

The results at the respective optimal iteration parameter, r_{opt}, are tabulated in Table 3.3.3-1, 3.3.3-2, 3.3.3-3 and (3.3.3-4).

x	Exact	Computed	E_{abs}	E_{pte}
.09	.64016497E-01	.64033024E-01	.16526873E-04	.10670945E-03
.18	.12931425E+00	.12934656E+00	.32303413E-04	.21430080E-03
.27	.19718512E+00	.19723170E+00	.46578798E-04	.32366323E-03
.36	.26894224E+00	.26900080E+00	.58560328E-04	.43570056E-03
.45	.34593088E+00	.34599831E+00	.67425282E-04	.55133872E-03
.55	.42953959E+00	.42961188E+00	.72292630E-04	.67153341E-03
.64	.52121165E+00	.52128386E+00	.72203583E-04	.79727795E-03
.73	.62245705E+00	.62252318E+00	.66128958E-04	.92961157E-03
.82	.73486491E+00	.73491783E+00	.52911860E-04	.10696279E-02
.91	.86011672E+00	.86014801E+00	.31286319E-04	.12184842E-02

Table 3.3.3-1: Problem 1 for N = 10 with FD formula, r_{opt} = 0.5

x	Exact	Computed	E_{abs}	E_{pte}
.09	.64016497E-01	.64016481E-01	.16115950E-07	.44090110E-07
.18	.12931425E+00	.12931422E+00	.36943794E-07	.88544851E-07
.27	.19718512E+00	.19718507E+00	.54031069E-07	.13373187E-06
.36	.26894224E+00	.26894217E+00	.73961545E-07	.18002488E-06
.45	.34593088E+00	.34593079E+00	.92035359E-07	.22780672E-06
.55	.42953959E+00	.42953949E+00	.10652684E-06	.27747256E-06
.64	.52121165E+00	.52121153E+00	.12246147E-06	.32943313E-06
.73	.62245705E+00	.62245692E+00	.12855725E-06	.38411818E-06
.82	.73486491E+00	.73486478E+00	.13710193E-06	.44197994E-06
.91	.86011672E+00	.86011658E+00	.14666171E-06	.50349695E-06

Table 3.3.3-2: Problem 1 for N = 10 with NU formula, r_{opt} = 0.5

x	Exact	Computed	E_{abs}	E_{pte}
.05	.33451410E-01	.33453792E-01	.23820599E-05	.15319887E-04
.14	.10109012E+00	.10109715E+00	.70332191E-05	.46098697E-04
.24	.17094317E+00	.17095450E+00	.11329837E-04	.77295873E-04
.33	.24450913E+00	.24452416E+00	.15027712E-04	.10919454E-03
.43	.32332026E+00	.32333813E+00	.17871012E-04	.14208420E-03
.52	.40895642E+00	.40897601E+00	.19587364E-04	.17626334E-03
.62	.50305944E+00	.50307932E+00	.19882605E-04	.21204214E-03
.71	.60734799E+00	.60736643E+00	.18435007E-04	.24974531E-03
.81	.72363322E+00	.72364811E+00	.14888737E-04	.28971504E-03
.90	.85383515E+00	.85384400E+00	.88452199E-05	.33231406E-03

Table 3.3.3-3: Problem 1 for N = 20 with FD formula, r_{opt} = 0.29

x	Exact	Computed	E_{abs}	E_{pte}
.05	.33451410E-01	.33451400E-01	.96064188E-08	.17368975E-08
.14	.10109012E+00	.10109009E+00	.24247434E-07	.52264587E-08
.24	.17094317E+00	.17094314E+00	.38791919E-07	.87634612E-08
.33	.24450913E+00	.24450908E+00	.53225150E-07	.12380011E-07
.43	.32332026E+00	.32332019E+00	.67358982E-07	.16108936E-07
.52	.40895642E+00	.40895634E+00	.80788720E-07	.19984085E-07
.62	.50305944E+00	.50305934E+00	.92956667E-07	.24040632E-07
.71	.60734799E+00	.60734789E+00	.10343102E-06	.28315399E-07
.81	.72363322E+00	.72363311E+00	.11256922E-06	.32847190E-07
.90	.85383515E+00	.85383503E+00	.12381932E-06	.37677140E-07

Table 3.3.3-4: Problem 1 for $N = 20$ with NU formula, $r_{opt} = 0.29$

Problem 2 - A Mildly Non-linear Problem.

$$-U'' + \frac{3}{2}U^2 = 0, \qquad 0 \le x \le 1,$$

$$U(0) = 4, \; U(1) = 1.$$

The exact solution to this problem is given by

$$U(x) = \frac{4}{(1+x)^2}.$$

The respective matrix A derived from the finite difference approximation and Numerov's formula are given by

$$A_{FD} = \begin{bmatrix} 2g_1 & -1 & & & \\ -1 & 2g_2 & -1 & & 0 \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2g_{N-1} & -1 \\ & 0 & & -1 & 2g_N \end{bmatrix} \text{ and } A_{NU} = \begin{bmatrix} 2w_1 & t_1 & & & \\ s_2 & 2w_2 & t_1 & & 0 \\ & \ddots & \ddots & \ddots & \\ & & s_{N-1} & 2w_{N-1} & t_{N-1} \\ & 0 & & s_N & 2w_N \end{bmatrix}$$

where $g_i = 1 + \frac{3}{4}h^2u_i$, $1 \le i \le N$, $\qquad s_i = \frac{1}{8}h^2u_{i-1} - 1$, $2 \le i \le N$,

$w_i = 1 + \frac{5}{8}h^2u_i$, $1 \le i \le N$, $\qquad$ and $t_i = \frac{1}{8}h^2u_{i+1} - 1$, $1 \le i \le N-1$.

The respective E_{pte} for Problem 2 with FD approximation and NU method are as follows.

$$E^{FD}_{pte} = \frac{h^2}{12}\left[\frac{15}{2}U^3\right] = \frac{5}{8}h^2U^3 \quad \text{and} \quad E^{NU}_{pte} = \frac{h^4}{240}\left[\frac{315}{4}U^4\right] = \frac{21}{64}h^4U^4.$$

The results for the respective optimal iteration parameter, r_{opt}, are tabulated in Tables 3.3.3-5, 3.3.3-6, 3.3.3-7 and 3.3.3-8.

x	Exact	Computed	E_{abs}	E_{pte}
.09	.33611111E+01	.33636121E+01	.25009683E-02	.19656780E+00
.18	.28639053E+01	.28674789E+01	.35736302E-02	.12178544E+00
.27	.24693877E+01	.24732768E+01	.38890889E-02	.78147097E-01
.36	.21511111E+01	.21549065E+01	.37954349E-02	.51686839E-01
.45	.18906250E+01	.18941018E+01	.34768061E-02	.35099797E-01
.55	.16747404E+01	.16777717E+01	.30312285E-02	.24394575E-01
.64	.14938271E+01	.14963373E+01	.25101404E-02	.17305460E-01
.73	.13407202E+01	.13426594E+01	.19391802E-02	.12502366E-01
.82	.12100000E+01	.12113294E+01	.13294563E-02	.91808200E-02
.91	.10975056E+01	.10981894E+01	.68373753E-03	.68411070E-02

Table 3.3.3-5: Problem 2 for N = 10 with FD formula, r_{opt} = 0.59

x	Exact	Computed	E_{abs}	E_{pte}
.09	.33611111E+01	.33610787E+01	.32413569E-04	.28601140E-02
.18	.28639053E+01	.28638621E+01	.43177241E-04	.15075685E-02
.27	.24693877E+01	.24693435E+01	.44248244E-04	.83328707E-03
.36	.21511111E+01	.21510701E+01	.41006200E-04	.47982941E-03
.45	.18906250E+01	.18905890E+01	.35934998E-04	.28632361E-03
.55	.16747404E+01	.16747103E+01	.30169626E-04	.17628988E-03
.64	.14938271E+01	.14938029E+01	.24187112E-04	.11159426E-03
.73	.13407202E+01	.13407020E+01	.18167801E-04	.72409857E-04
.82	.12100000E+01	.12099878E+01	.12156174E-04	.48038851E-04
.91	.10975056E+01	.10974995E+01	.61126772E-05	.32515168E-04

Table 3.3.3-6: Problem 2 for N = 10 with NU formula, r_{opt} = 0.57

x	Exact	Computed	E_{abs}	E_{pte}
.05	.36446281E+01	.36450590E+01	.43093159E-03	.68636502E-01
.14	.30625000E+01	.30633989E+01	.89887826E-03	.40742914E-01
.24	.26094674E+01	.26105304E+01	.10629317E-02	.25213184E-01
.33	.22500000E+01	.22510694E+01	.10693774E-02	.16166205E-01
.43	.19600000E+01	.19609925E+01	.99250881E-03	.10687330E-01
.52	.17226562E+01	.17235268E+01	.87053843E-03	.72559710E-02
.62	.15259515E+01	.15266745E+01	.72300267E-03	.50429055E-02
.71	.13611111E+01	.13616705E+01	.55940723E-03	.35781416E-02
.81	.12216066E+01	.12219906E+01	.38394051E-03	.25860982E-02
.90	.11025000E+01	.11026976E+01	.19766021E-03	.19002502E-02

Table 3.3.3-7: Problem 2 for N = 20 with FD formula, r_{opt} = 0.32

x	Exact	Computed	E_{abs}	E_{pte}
.05	.36446281E+01	.36446265E+01	.15918710E-05	.29769688E-03
.14	.30625000E+01	.30624969E+01	.30689776E-05	.14841056E-03
.24	.26094674E+01	.26094640E+01	.33958169E-05	.78228939E-04
.33	.22500000E+01	.22499967E+01	.32330233E-05	.43240403E-04
.43	.19600000E+01	.19599971E+01	.28590099E-05	.24899115E-04
.52	.17226562E+01	.17226538E+01	.24075751E-05	.14857790E-04
.62	.15259515E+01	.15259496E+01	.19174682E-05	.91479325E-05
.71	.13611111E+01	.13611096E+01	.14536154E-05	.57907472E-05
.81	.12216066E+01	.12216057E+01	.95769346E-06	.37573905E-05
.90	.11025000E+01	.11024995E+01	.47687908E-06	.24927331E-05

Table 3.3.3-8: Problem 2 for N = 20 with NU formula, r_{opt} = 0.32

The numerical results are carried out to a tolerance $\varepsilon = 10^{-8}$. The results clearly show that the greater accuracy is achieved when using the Numerov formula. The gains in accuracy in terms of E_{abs} and E_{pte} can be viewed as follows.

Table 3.3.3-1 (FD) versus *Table* 3.3.3-2 (NU) for 10 points, and

Table 3.3.3-3 (FD) versus *Table* 3.3.3-4 (NU) for 20 points.

Table 3.3.3-5 (FD) versus *Table* 3.3.3-6 (NU) for 10 points, and

Table 3.3.3-7 (FD) versus *Table* 3.3.3-8 (NU) for 20 points.

The results agree to the order of accuracy for the respective formula. However, more work is needed for both problems if the Numerov formula is considered. By considering the second problem which satisfies the algorithms 3.3.1-1 and 3.3.2-1, then for large N, we have

Formula	**The sweep**	**Operations**	**Extra work**
FD	First Second	$4N$ M + $3.5N$ A $4N$ M + $3.5N$ A Total = $8N$ M + $7N$ A = $15N$ operations	–
Numerov	First Second	$6.5N$ M + $3.5N$ A $6.5N$ M + $3.5N$ A Total = $13N$ M + $7N$ A = $20N$ operations	2 storage + 4 M + 3 A for the formula

Table 3.3.3-9 : The computational complexity

where M stands for multiplication and A for addition. Thus, from *Table* 3.3.3-9, it can be deduced that for large N, 33% more work is needed plus extra storage, i.e., the array s_i and t_i when using the Numerov formula.

3.4 Summary

This chapter begins with some discussion on the existing and well known iterative methods which are frequently used for solving the ODEs and PDEs. The SOR method has been shown to be more competitive than all the others. The AGE method based on the Alternating Direction Implicit (ADI) method as introduced by Evans [1984], is then given in full detail, to demonstrate that this method is competitive with the SOR method.

The experimental results in Sections 3.1.8 and 3.2.4 show how the AGE method competes with the SOR method. Some results show that the AGE method is superior than the SOR method especially for solving linear problems, when the number of iterations are compared. The only setback is perhaps due to the simplicity of the SOR method compared to the AGE method. However, since the matrices involved are purely small (2x2) block submatrices and these matrices are invertible, then, the AGE method seems fairly easy to implement. Also, with the capability of solving the problems governed by different boundary conditions, the AGE method may well be taken into consideration.

In actual fact, the algorithmic procedure given in this chapter can be written in a more compact form, which will avoid unnecessary computational effort. We will discuss these forms later in Chapter 4.

The AGE method is also proved to be numerically stable when using the Numerov formula in order to attain a greater accuracy. This is a very significant point for the AGE method as it is not only capable of solving a problem that yields a symmetric matrix A, but also the unsymmetric matrix A which is derived from the Numerov formula for nonlinear problems.

Finally, the experimental results indicate that the optimal value of the acceleration parameter r adheres closely to the theoretical analysis given when the matrix is diagonally dominant. For other matrices, it appears that the optimal value of r lies in the aritho-geometric mean interval of the eigenvalue bounds of the matrices involved. This interval will be discussed in greater detail in Chapter 5.

CHAPTER 4

THE OPTIMAL SINGLE AGE PARAMETER FOR ODE's

4.1 Determination of optimal AGE acceleration parameter

It has been shown in Chapter 3 that the AGE iterative method written in Peaceman-Rachford form is competitive compared to other existing iterative methods when solving the two-point boundary-value problem subject to Dirichlet boundary condition. The method is also stable when the problem is subjected to other boundary conditions. It has also been shown that the AGE method is stable when a more accurate formula, i.e, Numerov's method, is used to achieve a greater accuracy.

Since the AGE method in Peaceman-Rachford form will only serve to solve the one dimensional problem, then it is necessary to investigate alternative forms that allow solving the solution of a multi-dimensional problem. In this chapter, we will introduce the forms suggested by Douglas-Rachford, Douglas and Guittet. However, for comparison, we summarise briefly the AGE method in Peaceman-Rachford form.

Consider the linear system

$$\mathbf{A}\mathbf{u} = \mathbf{b} \tag{4.1-1}$$

where $\mathbf{u}$ and $\mathbf{b}$ are N-dimensional vectors and A is given as

$$A = \begin{bmatrix} 2g_1 & c_1 & & & \\ a_2 & 2g_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & a_{N-1} & 2g_{N-1} & c_{N-1} \\ & & & a_N & 2g_N \end{bmatrix}. \tag{4.1-2}$$

The AGE iterative method consists of splitting the matrix A into the form

$$A = G_1 + G_2 \tag{4.1-3}$$

where

$$G_1 = \begin{bmatrix} g_1 & c_1 & & & & & \\ a_2 & g_2 & & & & & \\ & & g_3 & c_3 & & & \\ & & a_4 & g_4 & & & \\ & & & & \ddots & & \\ & & & & & g_{N-1} & c_{N-1} \\ & & & & & a_N & g_N \end{bmatrix} \tag{4.1-4}$$

$$G_2 = \begin{bmatrix} g_1 & & & & & \\ & g_2 & c_2 & & & \\ & a_3 & g_3 & & & \\ & & & \ddots & & \\ & & & & g_{N-2} & c_{N-2} \\ & & & & a_{N-1} & g_{N-1} \\ & & & & & g_N \end{bmatrix} \tag{4.1-5}$$

for N is even. Let us assume that all the eigenvalues of G_1 and G_2 are real and positive, i.e., $g_i > \frac{1}{2}(a_i + c_i)$, $i = 1, 2, \ldots, N$.

4.1.1 The generalised AGE method in Peaceman-Rachford form

By applying the Peaceman-Rachford form, then for any iteration parameter $r > 0$, the AGE-PR(1) iterative scheme can be written as

$$(rI + G_1)\mathbf{u}^{(k+1/2)} = \mathbf{b} + (rI - G_2)\mathbf{u}^{(k)} \tag{4.1.1-1}$$

$$(rI + G_2)\mathbf{u}^{(k+1)} = \mathbf{b} + (rI - G_1)\mathbf{u}^{(k+1/2)} \tag{4.1.1-2}$$

or explicitly as

$$\mathbf{u}^{(k+1/2)} = (rI + G_1)^{-1}[\mathbf{b} + (rI - G_2)\mathbf{u}^{(k)}] \qquad (4.1.1\text{-}3)$$

$$\mathbf{u}^{(k+1)} = (rI + G_2)^{-1}[\mathbf{b} + (rI - G_1)\mathbf{u}^{(k+1/2)}]. \qquad (4.1.1\text{-}4)$$

It has been shown earlier that the AGE-PR(1) scheme is convergent. Now, let us consider the modification of equation (4.1.1-2) of the scheme. Then, for any $r > 0$, the AGE-PR(2) scheme can be written as

$$(rI + G_1)\mathbf{u}^{(k+1/2)} = \mathbf{b} + (rI - G_2)\mathbf{u}^{(k)} \qquad (4.1.1\text{-}5)$$

$$(rI + G_2)\mathbf{u}^{(k+1)} = 2r\mathbf{u}^{(k+1/2)} - (rI - G_2)\mathbf{u}^{(k)} \qquad (4.1.1\text{-}6)$$

after using equation (4.1.1-1) to express $G_1\mathbf{u}^{(k+1/2)}$ in terms of $G_2\mathbf{u}^{(k)}$, thereby saving on the evaluation of the right hand side vectors. The matrices G_1 and G_2 are as in (4.1-4) - (4.1-5). In explicit form, the AGE-PR(2) scheme can be written as

$$\mathbf{u}^{(k+1/2)} = (rI + G_1)^{-1}[\mathbf{b} + (rI - G_2)\mathbf{u}^{(k)}] \qquad (4.1.1\text{-}7)$$

$$\mathbf{u}^{(k+1)} = (rI + G_2)^{-1}[2r\mathbf{u}^{(k+1/2)} - (rI - G_2)\mathbf{u}^{(k)}] \qquad (4.1.1\text{-}8)$$

where the iteration matrix, T_r is given by

$$T_r = (rI + G_2)^{-1}[2r(rI + G_1)^{-1} - I](rI - G_2). \qquad (4.1.1\text{-}9)$$

It is obvious that, after dropping all the superscripts, the AGE-PR(2) scheme is consistent. Since G_1 and G_2 are unsymmetric, for the convergence, we need to show that $\|T_r\|_2 < 1$.

For this scheme,

$$\|T_r\|_2 = \|(rI + G_2)^{-1}[2r(rI + G_1)^{-1} - I](rI - G_2)\|_2 \qquad (4.1.1\text{-}10)$$

Let μ and ν be the respective eigenvalues of G_1 and G_2. Since all the eigenvalues of G_1 and G_2 are positive, then from (4.1.1-10)

$$\|T_r\|_2 = \|(rI + G_2)^{-1}[2r(rI + G_1)^{-1} - I](rI - G_2)\|_2$$

$$= \left|\frac{1}{(r + \nu)}\left(\frac{2r}{(r + \mu)} - 1\right)(r - \nu)\right|$$

$$= \max_{\mu,\nu}\frac{|(r - \mu)(r - \nu)|}{|(r + \mu)(r + \nu)|} < 1. \qquad (4.1.1\text{-}11)$$

Thus, the AGE-PR(2) scheme is convergent, and its rate of convergence is similar to the AGE-PR(1) scheme.

Now, let us introduce a parameter ω into equation (4.1.1-5). Then, the new set of equations become,

$$(rI + G_1)u^{(k+1/2)} = b + (rI - G_2)u^{(k)} \qquad (4.1.1\text{-}12)$$

$$(rI + G_2)u^{(k+1)} = (2-\omega)ru^{(k+1/2)} - [r(1-\omega)I - G_2]u^{(k)} \qquad (4.1.1\text{-}13)$$

or in explicit form

$$u^{(k+1/2)} = (rI + G_1)^{-1}[b + (rI - G_2)u^{(k)}] \qquad (4.1.1\text{-}14)$$

$$u^{(k+1)} = (rI + G_2)^{-1}[(2-\omega)ru^{(k+1/2)} - [r(1-\omega)I - G_2]u^{(k)}] \qquad (4.1.1\text{-}15)$$

resulting in the generalised AGE scheme.

Putting $\omega = 0$, we then have the AGE scheme (4.1.1-5) - (4.1.1-6). For $\omega = 1$, the scheme is analogous to one given by Douglas and Rachford [1956]. We call this scheme as the AGE method in Douglas-Rachford form (AGE-DR(1)). The AGE-DR(1) scheme can be written as

$$u^{(k+1/2)} = (rI + G_1)^{-1}[b + (rI - G_2)u^{(k)}] \qquad (4.1.1\text{-}16)$$

$$u^{(k+1)} = (rI + G_2)^{-1}[ru^{(k+1/2)} + G_2u^{(k)}]. \qquad (4.1.1\text{-}17)$$

The important feature for the new generalised scheme, $\omega = 1$, is that it can be applied to solve the boundary value problems with two or more variables, i.e., the two and three dimensional problems. This will be derived later in Chapter 6.

We now show that the AGE-DR(1) scheme is convergent.

The iteration matrix, T_r of (4.1.1-16) - (4.1.1-17) is given by

$$\begin{aligned} T_r &= (rI + G_2)^{-1}[r(rI + G_1)^{-1}(rI - G_2) + G_2] \\ &= (rI + G_2)^{-1}(rI + G_1)^{-1}[r(rI - G_2) + (rI + G_1)G_2] \\ &= (rI + G_2)^{-1}(rI + G_1)^{-1}[r^2I + G_1G_2]. \end{aligned} \qquad (4.1.1\text{-}18)$$

Since all the eigenvalues of G_1 and G_2 are positive, then

$$\|T_r\|_2 = \|(rI + G_2)^{-1}(rI + G_1)^{-1}[r^2 I + G_1 G_2]\|_2$$

$$= \frac{|\quad r^2 + \mu\nu \quad|}{|(r + \mu)(r + \nu)|} = \frac{|\quad r^2 + \mu\nu \quad|}{|r^2 + r(\mu + \nu) + \mu\nu|} < 1 \qquad (4.1.1\text{-}19)$$

where μ and ν are the maximum eigenvalues of G_1 and G_2. Thus, the AGE-DR(1) scheme is convergent. If $\gamma = \max(\mu, \nu)$, then (4.1.1-19) can be simplified to

$$\|T_r\|_2 \leqslant \frac{r^2 + \gamma^2}{(r + \gamma)^2} < 1. \qquad (4.1.1\text{-}20)$$

For the AGE-PR(2) scheme, $\|T_r\|_2$ is given by

$$\|T_r\|_2 = \frac{|(r - \mu)(r - \nu)|}{|(r + \mu)(r + \nu)|} < 1.$$

Thus,

$$-1 < \left(\frac{r - \mu}{r + \mu}\right)\left(\frac{r - \nu}{r + \nu}\right) < 1$$

$$-1 < \frac{(r - \mu)(r - \nu)}{(r + \mu)(r + \nu)} < 1$$

$$-1 < 1 - \frac{2r(\mu + \nu)}{(r + \mu)(r + \nu)} < 1. \qquad (4.1.1\text{-}21)$$

Now, from (4.1.1-19)

$$\|T_r\|_2 = \max_{\mu,\nu} \frac{|\quad r^2 + \mu\nu \quad|}{|(r + \mu)(r + \nu)|} < 1$$

which gives

$$-1 < \frac{r^2 + \mu\nu}{(r + \mu)(r + \nu)} < 1$$

$$-1 < 1 - \frac{r(\mu + \nu)}{(r + \mu)(r + \nu)} < 1. \qquad (4.1.1\text{-}22)$$

Generally, for $0 \leq \omega \leq 1$, we then have

$$-1 < 1 - \frac{(2-\omega)r(\mu + \nu)}{(r + \mu)(r + \nu)} < 1. \qquad (4.1.1\text{-}23)$$

It is clear that the inequalities (4.1.1-21) and (4.1.1-22) differ by a factor 2. Hence, it can be concluded that the rate of convergence of the AGE-PR(2) scheme is twice the rate of convergence of the AGE-DR(1) Scheme.

Now, by using the matrices G_1 and G_2 in (4.1-4) - (4.1-5), then the algorithm for a generalised scheme (4.1.1-14) - (4.1.1-15) may be written as follows.

Algorithm 4.1.1-1: The generalised AGE scheme, (4.1.1-14) - (4.1.1-15).

Set $u_i^{(k)} = 0$, i, $= 0, \ldots, N+1$, $a_1 = 0$, $c_N = 0$, $s = (2-\omega)r$.

Step 1. To compute $\mathbf{u}^{(k+1/2)}$. Set $i = 1$.

while $i \leq N-1$, compute

$$r_1 = b_i - a_i u_{i-1}^{(k)} + \beta_i u_i^{(k)}$$

$$r_2 = b_{i+1} + \beta_{i+1} u_{i+1}^{(k)} - c_{i+1} u_{i+2}^{(k)}$$

$$d = 1/(\alpha_i \alpha_{i+1} - a_{i+1} c_i)$$

$$u_i^{(k+1/2)} = (\alpha_{i+1} r_1 - c_i r_2) d$$

$$u_{i+1}^{(k+1/2)} = (-a_{i+1} r_1 + \alpha_i r_2) d$$

$$i = i + 2.$$

Step 2. To compute $\mathbf{u}^{(k+1)}$. Set $i = 2$.

$$u_1^{(k+1)} = (s u_1^{(k+1/2)} - \gamma_1 u_1^{(k)})/\alpha_1$$

while $i \leq N-2$, compute

$$r_1 = s u_i^{(k+1/2)} - \gamma_i u_i^{(k)} + c_i u_{i+1}^{(k)}$$

$$r_2 = s u_{i+1}^{(k+1/2)} + a_{i+1} u_i^{(k)} - \gamma_{i+1} u_{i+1}^{(k)}$$

$$d = 1/(\alpha_i \alpha_{i+1} - a_{i+1} c_i)$$

$$u_i^{(k+1)} = (\alpha_{i+1} r_1 - c_i r_2) d$$

$$u_{i+1}^{(k+1)} = (-a_{i+1} r_1 + \alpha_i r_2) d$$

$$i = i + 2.$$

$$u_N^{(k+1)} = (s u_N^{(k+1/2)} - \gamma_N u_N^{(k)})/\alpha_N$$

Step 3. Repeat *Step* 1 and *Step* 2 until convergence is achieved.

Here $\alpha_i = r + g_i$, $\beta_i = r - g_i$, $\gamma_i = (1-\omega)r - g_i$, $i = 1, 2, \ldots, N$.

Since this scheme is not competitive, i.e., when $\omega = 1$, we now investigate another variant of the AGE method.

4.1.2 The generalised AGE method in Douglas form

Let us consider the modification of equation (4.1.1-16) of the AGE-DR(1) scheme as follows. Then, for any $r > 0$, the new scheme can be written in explicit form as

$$u^{(k+1/2)} = (rI + G_1)^{-1}[b - Au^{(k)} + (rI + G_1)u^{(k)}] \qquad (4.1.2\text{-}1)$$

$$u^{(k+1)} = (rI + G_2)^{-1}[ru^{(k+1/2)} + G_2u^{(k)}] \qquad (4.1.2\text{-}2)$$

or

$$u^{(k+1/2)} = (rI + G_1)^{-1}[b + \{(rI + G_1) - A\}u^{(k)}] \qquad (4.1.2\text{-}3)$$

$$u^{(k+1)} = (rI + G_2)^{-1}[ru^{(k+1/2)} + G_2u^{(k)}] \qquad (4.1.2\text{-}4)$$

with the iteration matrix is given by

$$\begin{aligned} T_r &= (rI + G_2)^{-1}[r\{I - (rI + G_1)^{-1}A\} + G_2] \\ &= I - r(rI + G_2)^{-1}(rI + G_1)^{-1}A \end{aligned}$$

which simplifies to

$$T_r = I - r\prod_{i=2}^{1}(rI + G_i)^{-1}A. \qquad (4.1.2\text{-}5)$$

It is obvious that the scheme (4.1.2-3) - (4.1.2-4) is consistent. We now seek to analyse its convergence. Since all the eigenvalues G_1 and G_2 are positive, then

$$\begin{aligned} \|T_r\|_2 &= \|I - r\prod_{i=2}^{1}(rI + G_i)^{-1}A\|_2 \\ &= \left|1 - r\left(\frac{1}{r + \nu}\right)\left(\frac{1}{r + \mu}\right)(\mu + \nu)\right| \\ &= \left|1 - \frac{r\mu + r\nu}{(r + \mu)(r + \nu)}\right| \\ &= \left|\frac{r^2 + \mu\nu}{(r + \nu)(r + \mu)}\right| < 1 \end{aligned}$$

where μ and ν are the maximum eigenvalues of G_1 and G_2 respectively. Thus, the scheme is convergent and similar to the AGE-DR(1) scheme.

We now introduce a parameter ω in equation (4.1.2-3). The new set of equations then becomes

$$u^{(k+1/2)} = (rI + G_1)^{-1}[\omega b + \{(rI + G_1) - \omega A\}u^{(k)}] \qquad (4.1.2\text{-}6)$$

$$u^{(k+1)} = (rI + G_2)^{-1}[ru^{(k+1/2)} + G_2u^{(k)}]. \qquad (4.1.2\text{-}7)$$

This is another important feature, since the generalised AGE method (4.1.2-6) - (4.1.2-7) is applicable to solve problems with higher dimensions. This will be shown later in Chapter 6.

Putting $\omega = 1$, we have the scheme which is similar to AGE-DR(1) and denote this scheme as AGE-DR(2). For $\omega = 2$, the scheme is analogous to one given by Douglas [1956]. We denote this scheme as the AGE method in Douglas form (AGE-DG). Hence, The AGE-DG scheme is given by

$$u^{(k+1/2)} = (rI + G_1)^{-1}[2b + \{(rI + G_1) - 2A\}u^{(k)}] \qquad (4.1.2\text{-}8)$$

$$u^{(k+1)} = (rI + G_2)^{-1}[ru^{(k+1/2)} + G_2u^{(k)}] \qquad (4.1.2\text{-}9)$$

with the iteration matrix is given by

$$T_r = I - 2r\prod_{i=2}^{1}(rI + G_i)^{-1}A. \qquad (4.1.2\text{-}10)$$

Obviously, The AGE-DG scheme is consistent. We now show that this scheme is convergent. Since all the eigenvalues of G_1 and G_2 are positive, then

$$\|T_r\|_2 = \|I - 2r\prod_{i=2}^{1}(rI + G_i)^{-1}A\|_2$$

$$= \left|1 - r\left(\frac{1}{r+\nu}\right)\left(\frac{1}{r+\mu}\right)2(\mu+\nu)\right|$$

$$= \left|1 - \frac{2r\mu + 2r\nu}{(r+\mu)(r+\nu)}\right|$$

$$= \left|\frac{r^2 - r\mu - r\nu + \mu\nu}{(r+\mu)(r+\nu)}\right|$$

$$= \left|\frac{(r-\nu)(r-\mu)}{(r+\nu)(r+\mu)}\right| < 1$$

where μ and ν are the maximum eigenvalues of G_1 and G_2 respectively. Thus, the AGE-DG scheme is convergent and similar to the AGE-PR(2) scheme.

In general, the generalised AGE scheme (4.1.2-6) - (4.1.2-7) with the values of ω in [1,2] will have the iteration matrix as

$$T_r = I - \omega r \prod_{i=2}^{1} (rI + G_i)^{-1} A \qquad (4.1.2\text{-}11)$$

and the scheme is convergent.

It is clear that the generalised AGE scheme (4.1.2-6) - (4.1.2-7) will give the AGE-DG scheme when $\omega = 2$ and the AGE-DR(2) when $\omega = 1$. The AGE-DG scheme has also been shown to achieve a similar rate convergence as the AGE-PR(2) scheme. To prove the arguments, we present the algorithm in 4.1.2-1 which can then be transformed into a program and tested on the computer.

Let us recall the scheme (4.1.2-6) - (4.1.2-7), and write

$$\mathbf{u}^{(k+1/2)} = (rI + G_1)^{-1}[\omega \mathbf{b} + P\mathbf{u}^{(k)}] \qquad (4.1.2\text{-}12)$$

$$\mathbf{u}^{(k+1)} = (rI + G_2)^{-1}[r\mathbf{u}^{(k+1/2)} + G_2\mathbf{u}^{(k)}] \qquad (4.1.2\text{-}13)$$

where

$$P = (rI + G_1) - \omega A.$$

By using A in (4.1-2), G_1 in (4.1-4) and G_2 in (4.1-5), we have

$$P = \begin{bmatrix} p_1 & t_1 & & & \\ s_2 & p_2 & t_2 & & 0 \\ & \ddots & \ddots & \ddots & \\ & & s_{N-1} & p_{N-1} & t_{N-1} \\ 0 & & & s_N & p_N \end{bmatrix}. \qquad (4.1.2\text{-}14)$$

where $p_i = \alpha_i - 2g_i\omega$, $i = 1, 2, \ldots, N$,

$s_i = -a_i\omega$, $t_i = c_i(1 - \omega)$, $i = 1, 3, 5, \ldots, N\text{-}1$,

and $s_j = -a_j(1 - \omega)$, $t_j = -c_j\omega$, $j = 2, 4, 6, \ldots, N$,

with $\alpha_i = r + g_i$, $a_1 = 0$ and $c_N = 0$.

In programming, the arrays s_i, t_i, s_j and t_j may be assigned as variables, which will then consume less computer storage.

We now, present the algorithm in detail.

Algorithm 4.1.2-1: The generalised AGE scheme, (4.1.2-6) - (4.1.2-7).

Set $u_i^{(k)} = 0$, i, $= 0, \ldots, N+1$, $a_1 = 0$, $c_N = 0$.

Step 1. To compute $u^{(k+1/2)}$. Set $i = 1$.

while $i \le N-1$, compute

$$s = -a_i\omega,\ p_1 = \alpha_i - 2g_i\omega,\ t = c_i(1-\omega),$$

$$v = a_{i+1}(1-\omega),\ p_2 = \alpha_{i+1} - 2g_{i+1}\omega,\ q = -c_{i+1}\omega$$

$$r_1 = su_{i-1}^{(k)} + p_1u_i^{(k)} + tu_{i+1}^{(k)} + \omega b_i$$

$$r_2 = vu_i^{(k)} + p_2u_{i+1}^{(k)} + qu_{i+2}^{(k)} + \omega b_{i+1}$$

$$d = 1/(\alpha_i\alpha_{i+1} - a_{i+1}c_i)$$

$$u_i^{(k+1/2)} = (\alpha_{i+1}r_1 - c_ir_2)d$$

$$u_{i+1}^{(k+1/2)} = (-a_{i+1}r_1 + \alpha_ir_2)d$$

$$i = i + 2.$$

Step 2. To compute $u^{(k+1)}$. Set $i = 2$.

$$u_1^{(k+1)} = (ru_1^{(k+1/2)} + g_1u_1^{(k)})/\alpha_1$$

while $i \le N-2$, compute

$$r_1 = ru_i^{(k+1/2)} + g_iu_i^{(k)} + c_iu_{i+1}^{(k)}$$

$$r_2 = ru_{i+1}^{(k+1/2)} + a_{i+1}u_i^{(k)} + g_{i+1}u_{i+1}^{(k)}$$

$$d = 1/(\alpha_i\alpha_{i+1} - a_{i+1}c_i)$$

$$u_i^{(k+1)} = (\alpha_{i+1}r_1 - c_ir_2)d$$

$$u_{i+1}^{(k+1)} = (-a_{i+1}r_1 + \alpha_ir_2)d$$

$$i = i + 2.$$

$$u_N^{(k+1)} = (ru_N^{(k+1/2)} + g_Nu_N^{(k)})/\alpha_N$$

Step 3. Repeat *Step* 1 and *Step* 2 until convergence is achieved.

4.1.3 The generalised AGE method in Guittet's form

Guittet [1967] has considered another generalised form to solve the PDE problems with higher dimensions. Analogous to the one dimensional problem, the AGE method in Guittet's form (AGE-GT) may be written as

$$(rI + G_1)\mathbf{u}^{(k+1/2)} = \omega r[\mathbf{b} - A\mathbf{u}^{(k)}] + \prod_{i=1}^{2} (rI + G_i)\mathbf{u}^{(k)} \qquad (4.1.3\text{-}1)$$

$$(rI + G_2)\mathbf{u}^{(k+1)} = \mathbf{u}^{(k+1/2)} \qquad (4.1.3\text{-}2)$$

where the iteration matrix can be shown to have the form of (4.1.2-11). Thus, the scheme is also convergent and can be shown to be consistent.

Guittet has also considered the value of ω which will give the iterative method (4.1.3-1) and (4.1.3-2) better convergence. But, it is obvious that by putting $\omega = 1$, we have the AGE-DR(2) in Guittet's form, (AGE-DRGT), whilst for $\omega = 2$, yields the AGE-DG in Guittet's form (AGE-DGGT). This shows that the rate of convergence of the new scheme, at best, is similar to the AGE-DG scheme. The advantage of this form is that it also can be applied to solve problems with higher dimensions. This will be shown later in Chapter 6.

Let us rewrite the AGE-GT scheme (4.1.3-1) - (4.1.3-2) as

$$(rI + G_1)\mathbf{u}^{(k+1/2)} = [\prod_{i=1}^{2} (rI + G_i) - \omega rA]\mathbf{u}^{(k)} + \omega r\mathbf{b} \qquad (4.1.3\text{-}3)$$

$$(rI + G_2)\mathbf{u}^{(k+1)} = \mathbf{u}^{(k+1/2)} \qquad (4.1.3\text{-}4)$$

or in explicit form

$$\mathbf{u}^{(k+1/2)} = (rI + G_1)^{-1}[\{\prod_{i=1}^{2} (rI + G_i) - \omega rA\}\mathbf{u}^{(k)} + \omega r\mathbf{b}] \qquad (4.1.3\text{-}5)$$

$$\mathbf{u}^{(k+1)} = (rI + G_2)^{-1}\mathbf{u}^{(k+1/2)}. \qquad (4.1.3\text{-}6)$$

The disadvantage of the AGE-GT is in the evaluation of the matrix

$$P = \prod_{i=1}^{2} (rI + G_i) - \omega rA. \qquad (4.1.3\text{-}7)$$

which takes more computational effort.

The evaluation of P becomes more difficult when solving the problem with higher dimension as the multiplication of $(rI + G_i)$ becomes more complex. However, this is compensated by the simple calculation of the second equation.

Now, by considering the matrices A, G_1 and G_2 given in (4.1-2), (4.1-4) and (4.1.5), we write the algorithm for the AGE-GT scheme.

First, to evaluate the matrix P in (4.1.3-7). Let $\alpha_1 = r + g_1$.

$$P = \begin{bmatrix} \alpha_1 & c_1 & & & & \\ a_2 & \alpha_2 & & & & \\ & & \alpha_3 & c_3 & & \\ & & a_4 & \alpha_4 & & \\ & & & & \ddots & \\ & & & & \alpha_{N-1} & c_{N-1} \\ & & & & a_N & \alpha_N \end{bmatrix} \begin{bmatrix} \alpha_1 & & & & \\ & \alpha_2 & c_2 & & \\ & a_3 & \alpha_3 & & \\ & & & \ddots & \\ & & & \alpha_{N-2} & c_{N-2} \\ & & & a_{N-1} & \alpha_{N-1} \\ & & & & \alpha_N \end{bmatrix}$$

$$- \omega r \begin{bmatrix} 2g_1 & c_1 & & & 0 \\ a_2 & 2g_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & a_{N-1} & 2g_{N-1} & c_{N-1} \\ 0 & & & a_N & 2g_N \end{bmatrix}$$

$$= \begin{bmatrix} p_1 & r_1 & s_1 & & & & \\ w_1 & p_2 & q_2 & & & 0 & \\ & v_3 & p_3 & r_3 & s_3 & & \\ & t_3 & w_3 & p_4 & q_4 & \ddots & \\ & & \ddots & \ddots & \ddots & & \ddots \\ & & & v_{N-3} & p_{N-3} & r_{N-3} & s_{N-3} \\ & & & t_{N-3} & w_{N-3} & p_{N-2} & q_{N-2} \\ & 0 & & & & v_{N-1} & p_{N-1} & r_{N-1} \\ & & & & & t_{N-1} & w_{N-1} & p_N \end{bmatrix} \qquad (4.1.3\text{-}8)$$

where

$$p_i = \alpha_i^2 - 2\omega r g_i,\ 1 \le i \le N, \qquad s_i = c_i c_{i+1},\ i = 1, 3, \ldots, N-3,$$

$$t_i = a_i a_{i+1},\ v_i = a_i(\alpha_i - \omega r),\ i = 3, 5, \ldots, N-1,$$

$$z_i = c_i(\alpha_{i+1} - \omega r),\ w_i = a_{i+1}(\alpha_i - \omega r),\ i = 1, 3, \ldots, N-1,$$

and $q_{i+1} = c_{i+1}(\alpha_{i+1} - \omega r),\ i = 1, 3, \ldots, N-1.$

It should be noticed that since $c_N = 0$, then $s_{N-1} = 0$ and $q_N = 0$, and since $a_1 = 0$, then $t_1 = v_1 = 0$. In programming, these arrays may be assigned as variables, and hence conserve a lot of storage.

We now write the algorithm for the AGE-GT as follows.

Algorithm 4.1.3-1: The AGE-GT scheme, equations (4.1.3-5) - (4.1.3-6).

Set $u_i^{(k)} = 0,\ i = 0, \ldots, N+1,\ a_1 = 0,\ c_N = 0.$

Step 1. To compute $\mathbf{u}^{(k+1/2)}$ from equation (4.1.3-5). Set $i = 1$.

while $i \le N-1$, compute

$$v = a_i(\alpha_i - \omega r),\ p_1 = \alpha_i^2 - 2\omega r g_i,\ z = c_i(\alpha_{i+1} - \omega r),$$

$$s = c_i c_{i+1},\ t = a_i a_{i+1},\ w = a_{i+1}(\alpha_i - \omega r),$$

$$p_2 = \alpha_{i+1}^2 - 2\omega r g_{i+1},\ q = c_{i+1}(\alpha_{i+1} - \omega r),$$

$$r_1 = v u_{i-1}^{(k)} + p_1 u_i^{(k)} + z u_{i+1}^{(k)} + s u_{i+2}^{(k)} + \omega r b_i$$

$$r_2 = t u_{i-1}^{(k)} + w u_i^{(k)} + p_2 u_{i+1}^{(k)} + q u_{i+2}^{(k)} + \omega r b_{i+1}$$

$$d = 1/(\alpha_i \alpha_{i+1} - a_{i+1} c_i)$$

$$u_i^{(k+1/2)} = (\alpha_{i+1} r_1 - c_i r_2) d$$

$$u_{i+1}^{(k+1/2)} = (-a_{i+1} r_1 + \alpha_i r_2) d$$

$$i = i + 2.$$

Step 2. To compute $\mathbf{u}^{(k+1)}$ from equation (4.1.3-6). Set $i = 2$.

$$u_1^{(k+1)} = u_1^{(k+1/2)}/\alpha_1$$

while $i \le N-2$, compute

$$d = 1/(\alpha_i \alpha_{i+1} - a_{i+1} c_i)$$

$$u_i^{(k+1)} = (\alpha_{i+1} u_i^{(k+1/2)} - c_i u_{i+1}^{(k+1/2)}) d$$

$$u_{i+1}^{(k+1)} = (-a_{i+1}u_{i}^{(k+1/2)} + \alpha_{i}u_{i+1}^{(k+1/2)})d$$

$$i = i + 2.$$

$$u_{N}^{(k+1)} = u_{N}^{(k+1/2)}/\alpha_{N}.$$

Step 3. Repeat *Step* 1 and *Step* 2 until convergence is achieved.

4.1.4 The computational complexity

It has been shown that the AGE-PR, AGE-DG and AGE-DGGT schemes have a similar rate of convergence, whilst the AGE-DR method converges slower than the three schemes. To determine which scheme is the best, we now analyse the arithmetic operations involved in the respective schemes. For each scheme, we will estimate the amount of operations needed in each iteration that is required to solve the model problem.

In this section, the comparison of the computational complexity for the three schemes will be based on the algorithms 4.1.1-1, 4.1.2-1 and 4.1.3-1 respectively. *Table* 4.1.4-1 summarises the amount of operations required for each algorithm when N is large.

Algorithm	Multiplication	Addition	Overall
4.1.1-1	14N	7N	21N
4.1.2-1	18N	10N	28N
4.1.3-1	19N	10N	29N

Table 4.1.4-1: The amount of operations per iteration.

The amount of computational work in presented *Table* 4.1.4-1 serves as a general comparison for the three schemes. Moreover, the amount of work shown satisfies a problem that yield the matrix A in (4.1-2) where g_i is not a function of u_i. However, faster scheme can be derived if $g_i = g$, a_i and/or $c_i = -1$, $\forall$ i, which reduces the computational work accordingly.

4.1.5 Experimental results

Three linear and one non-linear problems were tested to compare the results for the schemes that have been discussed. Problem 1 is run with several values of constant ρ to see how the schemes compare. The matrix A derived from each problem is also presented and is used as a guide to compute the amount of computational work needed by the problem.

Problem 1 - A Linear Problem

$$-U'' + \rho U = (\rho+1)(\sin x + \cos x), \qquad 0 \le x \le \frac{\pi}{2},$$

$$U(0) = 1,\ U(\frac{\pi}{2}) = 1. \qquad h = \pi/2(N+1).$$

The exact solution is $U(x) = \sin x + \cos x$.

The matrix A is

$$A = \begin{bmatrix} 2g & -1 & & & \\ -1 & 2g & -1 & & 0 \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2g & -1 \\ & 0 & & -1 & 2g \end{bmatrix}$$

where $g = 1 + 0.5\rho h^2$.

Algorithm - Scheme	Multiplication	Addition	Overall
4.1.1-1 - AGE-DR(1) (ω = 1)	7N	6N	13N
4.1.1-1 - AGE-PR(2) (ω = 0)	7N	6N	13N
4.1.2-1 - AGE-DR(2) (ω = 1)	7N	6N	13N
4.1.2-1 - AGE-DG (ω = 2)	8N	7N	15N
4.1.3-1 - AGE-DRGT (ω = 1)	7N	6N	13N
4.1.3-1 - AGE-DGGT (ω = 2)	7N	6N	13N

Table 4.1.5-1 : Problem 1, The amount of computation per iteration

The results for various ρ are tabulated as follows:

	AGE-PR(2), $\omega = 0$ AGE-DG, $\omega = 2$ AGE-DGGT $\omega = 2$		AGE-DR(1), $\omega = 1$ AGE-DR(2), $\omega = 1$ AGE-DRGT $\omega = 1$	
N	optimum *r*	iter	*r*	iter
10	0.490	17	0.490	38
20	0.277 - 0.280	38	0.277 - 0.280	70-73
40	0.150 - 0.151	77	0.150 - 0.151	139-141
80	0.080 - 0.081	156	0.080 - 0.081	278-286
160	0.043	312	0.043	565

Table 4.1.5-2 : Problem 1, Number of Iterations when $\rho = 0$

	AGE-PR(2), $\omega = 0$ AGE-DG, $\omega = 2$ AGE-DGGT $\omega = 2$		AGE-DR(1), $\omega = 1$ AGE-DR(2), $\omega = 1$ AGE-DRGT $\omega = 1$	
N	optimum *r*	iter	*r*	iter
10	0.510 - 0.529	17	0.510 - 0.529	33-37
20	0.290 - 0.291	34	0.290 - 0.291	61-62
40	0.155 - 0.158	69	0.155 - 0.158	116-123
80	0.082 - 0.084	142	0.082 - 0.084	231-244
160	0.044 - 0.045	289	0.044 - 0.045	474-493

Table 4.1.5-3 : Problem 1, Number of Iterations when $\rho = 1$

	AGE-PR(2), $\omega = 0$ AGE-DG, $\omega = 2$ AGE-DGGT $\omega = 2$		AGE-DR(1), $\omega = 1$ AGE-DR(2), $\omega = 1$ AGE-DRGT $\omega = 1$	
N	optimum *r*	iter	*r*	iter
10	0.705 - 1.193	7	0.705 - 1.193	20-21
20	0.556 - 0.608	12	0.556 - 0.608	27-29
40	0.303 - 0.322	23	0.303 - 0.322	44-48
80	0.158 - 0.176	47	0.158 - 0.176	79-91
160	0.087 - 0.097	96	0.087 - 0.097	159-179

Table 4.1.5-4 : Problem 1, Number of Iterations when $\rho = 40$

	AGE-PR(2), $\omega = 0$ AGE-DG, $\omega = 2$ AGE-DGGT $\omega = 2$		AGE-DR(1), $\omega = 1$ AGE-DR(2), $\omega = 1$ AGE-DRGT $\omega = 1$	
N	optimum *r*	iter	*r*	iter
10	1.044 - 1.619	6	1.044 - 1.619	18-19
20	0.495 - 0.781	10	0.495 - 0.781	24-25
40	0.219 - 0.414	19	0.219 - 0.414	40-41
80	0.102 - 0.109	36	0.102 - 0.109	69-68
160	0.044 - 0.047	71	0.044 - 0.047	129-125

Table 4.1.5-5 : Problem 1, Number of Iterations when $\rho = 70$

	AGE-PR(2), $\omega = 0$	AGE-DG, $\omega = 2$	AGE-DGGT $\omega = 2$	AGE-DR(1), $\omega = 1$	AGE-DR(2), $\omega = 1$	AGE-DRGT $\omega = 1$
N	optimum r		iter	r		iter
10	1.555 - 1.741		5	1.555 - 1.741		17-18
20	0.571 - 0.950		9	0.571 - 0.950		22-24
40	0.270 - 0.460		16	0.270 - 0.460		35-35
80	0.117 - 0.163		31	0.117 - 0.163		59-54
160	0.051 - 0.063		61	0.051 - 0.063		110-100

Table 4.1.5-6 : Problem 1, Number of Iterations when $\rho = 100$

	AGE-PR(2), $\omega = 0$	AGE-DG, $\omega = 2$	AGE-DGGT $\omega = 2$	AGE-DR(1), $\omega = 1$	AGE-DR(2), $\omega = 1$	AGE-DRGT $\omega = 1$
N	optimum r		iter	r		iter
10	2.675 - 2.739		4	2.675 - 2.739		17
20	0.851 - 1.369		7	0.851 - 1.369		19-20
40	0.569 - 0.577		11	0.569 - 0.577		26
80	0.169 - 0.308		22	0.169 - 0.308		44
160	0.077 - 0.090		42	0.077 - 0.090		79-73

Table 4.1.5-7 : Problem 1, Number of Iterations when $\rho = 200$

Problem 2 - A Linear Problem

$$U'' - \left(\frac{1 + x}{2 + x}\right)U = 0, \qquad 0 \le x \le 1,$$

$$U(0) = 1,\ U(1) = 0. \qquad h = 1/(N+1).$$

The exact solution is not known, but the computed solution can be obtained when $\| \mathbf{r} \| = \|\mathbf{b} - A\mathbf{u}^{(k+1)}\| < \varepsilon$. The matrix A for this problem is given by

$$A = \begin{bmatrix} 2g_1 & -1 & & & \\ -1 & 2g_2 & -1 & & 0 \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2g_{N-1} & -1 \\ 0 & & & -1 & 2g_N \end{bmatrix}.$$

where $g_i = 1 + 0.5h^2\left(\frac{1 + x}{2 + x}_i\right)$, $1 \le i \le N$.

The results are tabulated as follows:

Algorithm - Scheme	Multiplication	Addition	Overall
4.1.1-1 - AGE-DR(1) (ω = 1)	9N	7N	16N
4.1.1-1 - AGE-PR(2) (ω = 0)	9N	7N	16N
4.1.2-1 - AGE-DR(2) (ω = 1)	10N	7N	17N
4.1.2-1 - AGE-DG (ω = 2)	10N	8N	18N
4.1.3-1 - AGE-DRGT (ω = 1)	10N	7N	17N
4.1.3-1 - AGE-DGGT (ω = 2)	10N	7N	17N

Table 4.1.5-8 : Problem 2, The amount of computation per iteration

	AGE-PR(2), ω = 0	AGE-DG, ω = 2	AGE-DGGT ω = 2	AGE-DR(1), ω = 1	AGE-DR(2), ω = 1	AGE-DRGT ω = 1
N	optimum r		iter	r		iter
10	0.488 - 0.520		19	0.488 - 0.520		37-45
20	0.277 - 0.294		39	0.277 - 0.294		69-82
40	0.153 - 0.156		78	0.153 - 0.156		143-149
80	0.083 - 0.085		159	0.083 - 0.085		289-300
160	0.046		314	0.046		592

Table 4.1.5-9 : Problem 2, Number of Iterations

Problem 3 - A Linear Problem

$$U'' - (1 + x^2)U = -1, \qquad -1 \le x \le 1,$$

$$U(-1) = 0, \; U(1) = 0. \qquad h = 2/(N+1).$$

The computed solution is obtained when $\| \mathbf{r} \| = \| \mathbf{b} - A\mathbf{u}^{(k+1)} \| < \varepsilon$.

The matrix A for this problem is given by

$$A = \begin{bmatrix} 2g_1 & -1 & & & \\ -1 & 2g_2 & -1 & & 0 \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2g_{N-1} & -1 \\ 0 & & & -1 & 2g_N \end{bmatrix}$$

where $g_i = 1 + 0.5h^2(1 + x_i^2)$, $1 \le i \le N$.

The results are tabulated as follows:

Algorithm - Scheme	Multiplication	Addition	Overall
4.1.1-1 - AGE-DR(1) ($\omega = 1$)	$9N$	$7N$	$16N$
4.1.1-1 - AGE-PR(2) ($\omega = 0$)	$9N$	$7N$	$16N$
4.1.2-1 - AGE-DR(2) ($\omega = 1$)	$10N$	$7N$	$17N$
4.1.2-1 - AGE-DG ($\omega = 2$)	$10N$	$8N$	$18N$
4.1.3-1 - AGE-DRGT ($\omega = 1$)	$10N$	$7N$	$17N$
4.1.3-1 - AGE-DGGT ($\omega = 2$)	$10N$	$7N$	$17N$

Table 4.1.5-10 : Problem 3, The amount of computation per iteration

	AGE-PR(2), AGE-DG, AGE-DGGT $\omega = 0$ $\omega = 2$ $\omega = 2$		AGE-DR(1), AGE-DR(2), AGE-DRGT $\omega = 1$ $\omega = 1$ $\omega = 1$	
N	optimum r	iter	r	iter
10	0.560 - 0.580	12	0.560 - 0.580	30-32
20	0.303 - 0.305	20	0.303 - 0.305	46-47
40	0.160 - 0.161	39	0.160 - 0.161	82-83
80	0.080	70	0.080	144
160	0.040	132	0.040	267

Table 4.1.5-11 : Problem 3, Number of Iterations

Problem 4 - A Mildly Nonlinear Problem.

$$-U'' + \frac{3}{2}U^2 = 0.$$

$U(0) = 4$, $U(1) = 1$. $\quad h = 1/(N+1)$.

The exact solution to this problem is given by $U(x) = 4/(1+x)^2$.

The matrix A is given by

$$A = \begin{bmatrix} 2g_1 & -1 & & & \\ -1 & 2g_2 & -1 & & 0 \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2g_{N-1} & -1 \\ & 0 & & -1 & 2g_N \end{bmatrix}$$

where $g_i = 1 + 0.75h^2u_i$, $1 \le i \le N$.

The result is tabulated as follows:

Algorithm - Scheme	Multiplication	Addition	Overall
4.1.1-1 - AGE-DR(1) (ω = 1)	10*N*	10*N*	20*N*
4.1.1-1 - AGE-PR(2) (ω = 0)	10*N*	11*N*	21*N*
4.1.2-1 - AGE-DR(2) (ω = 1)	11*N*	10*N*	21*N*
4.1.2-1 - AGE-DG (ω = 2)	13*N*	10*N*	23*N*
4.1.3-1 - AGE-DRGT (ω = 1)	14*N*	14*N*	28*N*
4.1.3-1 - AGE-DGGT (ω = 2)	14*N*	14*N*	28*N*

Table 4.1.5-12 : Problem 4, The amount of computation per iteration

	AGE-PR(2), ω = 0 AGE-DG, ω = 2 AGE-DGGT ω = 2		AGE-DR(1), ω = 1 AGE-DR(2), ω = 1 AGE-DRGT ω = 1	
N	optimum *r*	iter	*r*	iter
10	0.572 - 0.579	17	0.572 - 0.579	35-36
20	0.317 - 0.323	35	0.317 - 0.323	63-66
40	0.164 - 0.179	72	0.164 - 0.179	109-135
80	0.089 - 0.093	143	0.089 - 0.093	233-253
160	0.048 - 0.049	285	0.048 - 0.049	478-493

Table 4.1.5-13 : Problem 4, Number of Iterations

The results also show that, for each problem, the amount of work needed per iteration is almost equal regardless of the algorithms. However, in terms of the amount of computational work needed for each iteration, for large *N*, the AGE-PR(2) scheme requires less work than the other two schemes. For Problem 4, Algorithm 4.1.3-1 shows a little more extra work since g_i depends on $u_i^{(k)}$.

The results show that the number of iterations for each problem determined by the above algorithms are in qualitative agreement with the theory presented in Sections 4.1.1 - 4.1.3. This can be seen from the tables, that the AGE-PR(2), AGE-DG and AGE-DGGT schemes yield similar results in terms of optimal *r* and number of iterations.

This shows that AGE-PR(2), AGE-DG, AGE-DGGT schemes are better than the scheme based on AGE-DR. The results for the AGE-DR(1), AGE-DR(2) and AGE-DRGT schemes are presented in the form of ranges to show the variation in iterations over the range of r. For example, in Problem 4, for N = 160, r ranges from 0.048 to 0.049 which gives the number of iterations from 478 to 493. The other ranges can be described in the same way.

Now, by careful consideration of the simplicity of the algorithms and the possible need to extend it to solve problems with higher dimensions, then, at this stage, we may say that Algorithm 4.1.2-1 for ω = 2, i.e., the AGE-DG scheme is the best choice.

4.1.6 The AGE-DR method with optimal single parameter - experimental results

Since the optimal single parameter for the AGE-DR method is based on the closely related AGE-PR(1) form, no theoretical analysis has been given. So, the optimal single parameter for the AGE-DR(1), AGE-DR(2) and AGE-DRGT schemes are now determined experimentally. Obviously we cannot expect that, the optimal single parameter to be the same since the iteration matrices are no longer of symmetric and regular form. The results for each problem are presented below.

	AGE-DR(1), ω = 1	AGE-DR(2), ω = 1	AGE-DRGT ω = 1
N	optimum r		iter
10	0.474		33
20	0.262		54
40	0.138		97
80	0.069		161
160	0.035		317

Table 4.1.5-14 : Problem 1, Number of Iterations when ρ = 0

	AGE-DR(1), ω = 1	AGE-DR(2), ω = 1	AGE-DRGT ω = 1
N	optimum *r*		iter
10	0.498 - 0.500		31
20	0.277 - 0.278		51
40	0.144		84
80	0.074		161
160	0.037		316

Table 4.1.5-15 : Problem 1, Number of Iterations when ρ = 1

	AGE-DR(1), ω = 1	AGE-DR(2), ω = 1	AGE-DRGT ω = 1
N	optimum *r*		iter
10	0.788 - 1.023		19
20	0.484 - 0.551		26
40	0.269 - 0.277		40
80	0.137 - 0.139		69
160	0.069		127

Table 4.1.5-16 : Problem 1, Number of Iterations when ρ = 40

	AGE-DR(1), ω = 1	AGE-DR(2), ω = 1	AGE-DRGT ω = 1
N	optimum *r*		iter
10	1.012 - 1.501		18
20	0.514 - 0.713		23
40	0.318 - 0.340		33
80	0.169		54
160	0.080 - 0.081		86

Table 4.1.5-17 : Problem 1, Number of Iterations when ρ = 70

	AGE-DR(1), ω = 1	AGE-DR(2), ω = 1	AGE-DRGT ω = 1
N	optimum *r*		iter
10	1.453 - 1.691		17
20	0.617 - 0.824		21
40	0.351 - 0.392		29
80	0.184 - 0.190		46
160	0.092		83

Table 4.1.5-18 : Problem 1, Number of Iterations when ρ = 100

	AGE-DR(1), $\omega = 1$	AGE-DR(2), $\omega = 1$	AGE-DRGT $\omega = 1$
N	optimum *r*		iter
10	1.800 - 3.332		17
20	0.816 - 1.249		19
40	0.477 - 0.520		24
80	0.245 - 0.250		37
160	0.125 - 0.126		64

Table 4.1.5-19 : Problem 1, Number of Iterations when $\rho = 200$

	AGE-DR(1), $\omega = 1$	AGE-DR(2), $\omega = 1$	AGE-DRGT $\omega = 1$
N	optimum *r*		iter
10	0.473 - 0.474		32
20	0.266		59
40	0.141		115
80	0.073		224
160	0.037		444

Table 4.1.5-20 : Problem 2, Number of Iterations

	AGE-DR(1), $\omega = 1$	AGE-DR(2), $\omega = 1$	AGE-DRGT $\omega = 1$
N	optimum *r*		iter
10	0.520 - 0.526		26
20	0.290 - 0.291		42
40	0.150		72
80	0.076		133
160	0.038		249

Table 4.1.5-21 : Problem 3, Number of Iterations

	AGE-DR(1), $\omega = 1$	AGE-DR(2), $\omega = 1$	AGE-DRGT $\omega = 1$
N	optimum *r*		iter
10	0.539 - 0.541		30
20	0.296		50
40	0.154		88
80	0.078		163
160	0.039		317

Table 4.1.5-22 : Problem 4, Number of Iterations

These results show that the range of optimum r is slightly different from that given by the earlier AGE-PR(2) form. Although, there are some small gains in the number of iterations, these improvements do not show any sign of superiority of the AGE-PR(2), AGE-DG and AGE-DGGT schemes. Thus, again we may consider that, at this stage, the AGE-DG scheme is a better choice.

4.2 The solution with alternative computational forms

Now, it is obvious that the form and rearrangement of the equations to be solved is important in order to save time in the computation. In this section, we will consider alternative forms for the schemes in solving the two-point boundary-value problem in relation to the time taken for each iteration.

Let us recall the three generalised schemes given in Section 4.1. Then, the respective explicit formula for these schemes can be written as follows:

The AGE-PR(2) Scheme

$$\mathbf{u}^{(k+1/2)} = (rI + G_1)^{-1}[\mathbf{b} + (rI - G_2)\mathbf{u}^{(k)}] \qquad (4.2.1\text{-}1)$$

$$\mathbf{u}^{(k+1)} = (rI + G_2)^{-1}[2r\mathbf{u}^{(k+1/2)} - (rI - G_2)\mathbf{u}^{(k)}]. \qquad (4.2.1\text{-}2)$$

The AGE-DG Scheme

$$\mathbf{u}^{(k+1/2)} = (rI + G_1)^{-1}[2\mathbf{b} + \{(rI + G_1) - 2A\}\mathbf{u}^{(k)}] \qquad (4.2.1\text{-}3)$$

$$\mathbf{u}^{(k+1)} = (rI + G_2)^{-1}[r\mathbf{u}^{(k+1/2)} + G_2\mathbf{u}^{(k)}]. \qquad (4.2.1\text{-}4)$$

The AGE-DGGT Scheme

$$\mathbf{u}^{(k+1/2)} = (rI + G_1)^{-1}[\{\prod_{i=1}^{2}(rI + G_i) - 2rA\}\mathbf{u}^{(k)} + 2r\mathbf{b}] \qquad (4.2.1\text{-}5)$$

$$\mathbf{u}^{(k+1)} = (rI + G_2)^{-1}\mathbf{u}^{(k+1/2)}. \qquad (4.2.1\text{-}6)$$

We will also consider the AGE-PR(1) Scheme

$$\mathbf{u}^{(k+1/2)} = (rI + G_1)^{-1}[\mathbf{b} + (rI - G_2)\mathbf{u}^{(k)}] \qquad (4.2.1\text{-}7)$$

$$\mathbf{u}^{(k+1)} = (rI + G_2)^{-1}[\mathbf{b} + (rI - G_1)\mathbf{u}^{(k+1/2)}]. \qquad (4.2.1\text{-}8)$$

In this section, we will use the matrices A, G_1 and G_2 as given in (4.1-2), (4.1-3) and (4.1-4).

4.2.1 The computational forms of AGE-PR, AGE-DG and AGE-DGGT schemes

The Algorithms 4.1.2-1, 4.1.2-2 and 4.1.2-3 show that the computation of $\mathbf{u}^{(k+1/2)}$ and $\mathbf{u}^{(k+1)}$ are made up of terms involving intermediate values r_1 and r_2. Now, instead of computing r_1 and r_2 separately, we will consider the substitution of these values into the solutions of $\mathbf{u}^{(k+1/2)}$ and $\mathbf{u}^{(k+1)}$. By doing so, we will then have the solutions of $\mathbf{u}^{(k+1/2)}$ and $\mathbf{u}^{(k+1)}$ in terms of a computational molecule which may lead to saving in computational work.

In algorithmic form, the respective molecular forms of the three schemes can be presented as follows. We commence with the AGE-PR(2) scheme. Let $\alpha_i = r + g_i$, $\beta_i = r - g_i$, $i = 1, 2, \ldots, N$.

Algorithm 4.2.1-1: The Computational form of the AGE-PR(2) Scheme, (COMP-AGE-PR(2)), equations (4.2.1-1) - (4.2.1-2).

Set $u_i^{(k)} = 0$, $i = 0, \ldots, N+1$, $a_1 = 0$, $c_N = 0$, $s = 2r$.

Step 1. To compute $\mathbf{u}^{(k+1/2)}$. Set $i = 1$.

while $i \le N-1$, compute

$$d = 1/(\alpha_i\alpha_{i+1} - a_{i+1}c_i),\ A = -d\alpha_{i+1}a_i,\ B = d\alpha_{i+1}\beta_i,$$

$$C = -dc_i\beta_{i+1},\ D = dc_ic_{i+1},\ E = d(\alpha_{i+1}b_i - c_ib_{i+1}),$$

$$P = da_ia_{i+1},\ Q = -da_{i+1}\beta_i,\ R = d\alpha_i\beta_{i+1},$$

$$S = -d\alpha_ic_{i+1},\ T = d(\alpha_ib_{i+1} - a_{i+1}b_i),$$

$$u_i^{(k+1/2)} = Au_{i-1}^{(k)} + Bu_i^{(k)} + Cu_{i+1}^{(k)} + Du_{i+2}^{(k)} + E$$

$$u_{i+1}^{(k+1/2)} = Pu_{i-1}^{(k)} + Qu_i^{(k)} + Ru_{i+1}^{(k)} + Su_{i+2}^{(k)} + T$$

$$i = i + 2.$$

Step 2. To compute $\mathbf{u}^{(k+1)}$. Set $i = 2$.

$$u_1^{(k+1)} = (su_1^{(k+1/2)} - \beta_1 u_1^{(k)})/\alpha_1$$

while $i \le N-2$, compute

$$d = 1/(\alpha_i\alpha_{i+1} - a_{i+1}c_i),\ A = ds\alpha_{i+1},\ B = -dsc_i,$$

$$C = -d(\alpha_{i+1}\beta_i + a_{i+1}c_i),\ D = dc_i(\alpha_{i+1} + \beta_{i+1}),$$

$$P = -dsa_{i+1},\ Q = ds\alpha_i,\ R = da_{i+1}(\alpha_i + \beta_i),$$

$$S = -d(\alpha_i\beta_{i+1} + a_{i+1}c_i).$$

$$u_i^{(k+1)} = Au_i^{(k+1/2)} + Bu_{i+1}^{(k+1/2)} + Cu_i^{(k)} + Du_{i+1}^{(k)}$$

$$u_{i+1}^{(k+1)} = Pu_i^{(k+1/2)} + Qu_{i+1}^{(k+1/2)} + Ru_i^{(k)} + Su_{i+1}^{(k)}$$

$$i = i + 2.$$

$$u_N^{(k+1)} = (su_N^{(k+1/2)} - \beta_N u_N^{(k)})/\alpha_N.$$

Step 3. Repeat *Step* 1 and *Step* 2 until convergence is achieved.

The computational molecules for *Step* 1, i.e., to compute $u_i^{(k+1/2)}$, $i = 1, 3, \ldots, N-1$, are given by *Figure* 4.2.1-1.

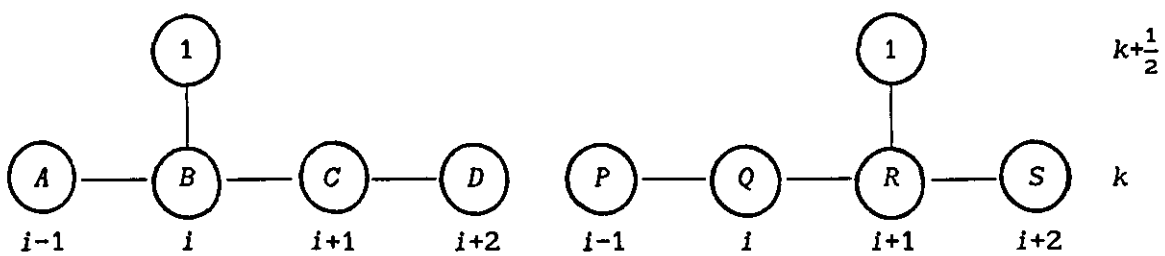


Figure 4.2.1-1 : The computational molecule for $\mathbf{u}^{(k+1/2)}$, AGE-PR(2).

For the computation of $u_i^{(k+1)}$, $i = 2, 4, \ldots, N$, the computational molecules can be presented as in *Figure* 4.2.1-2.

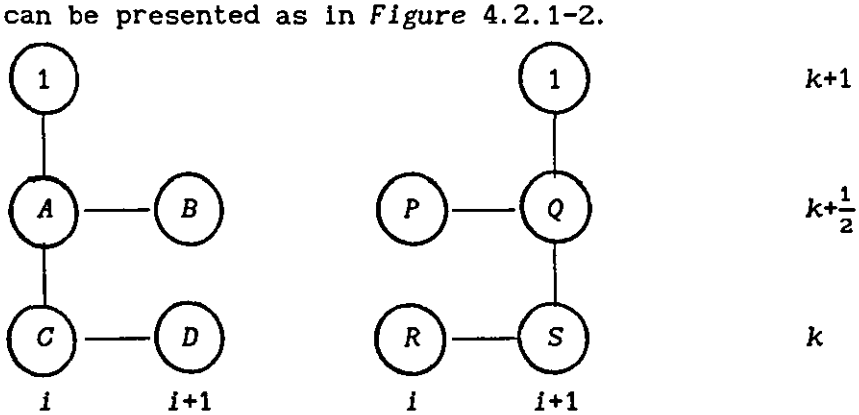


Figure 4.2.1-2 : The computational molecules for $\mathbf{u}^{(k+1)}$, AGE-PR(2)

Now, we present the algorithm for the AGE-DG scheme in the computational by using equations (4.2.1-3) - (4.2.1-4).

Algorithm 4.2.1-2: The Computational form of the AGE-DG scheme, (COMP-AGE-DG), Equations (4.2.1-3) - (4.2.1-4).

Set $u_i^{(k)} = 0,\ i = 0,\ \ldots,\ N+1,\ a_1 = 0,\ c_N = 0.$

Step 1. To compute $\mathbf{u}^{(k+1/2)}$. Set $i = 1$.

while $i \le N-1$, compute

$$d = 1/(\alpha_i\alpha_{i+1} - a_{i+1}c_i),\ A = -2da_i\alpha_{i+1},\ D = 2dc_ic_{i+1}$$

$$B = 1 + 2d(a_{i+1}c_i - 2g_i\alpha_{i+1}),\ P = 2da_ia_{i+1},$$

$$C = 2dc_i(2g_{i+1} - \alpha_{i+1}),\ E = 2d(\alpha_{i+1}b_i - c_ib_{i+1}),$$

$$Q = 2da_{i+1}(2g_i - \alpha_i),\ T = 2d(\alpha_ib_{i+1} - a_{i+1}b_i),$$

$$R = 1 + 2d(a_{i+1}c_i - 2g_{i+1}\alpha_i),\ S = -2dc_{i+1}\alpha_i$$

$$u_i^{(k+1/2)} = Au_{i-1}^{(k)} + Bu_i^{(k)} + Cu_{i+1}^{(k)} + Du_{i+2}^{(k)} + E$$

$$u_{i+1}^{(k+1/2)} = Pu_{i-1}^{(k)} + Qu_i^{(k)} + Ru_{i+1}^{(k)} + Su_{i+2}^{(k)} + T$$

$$i = i + 2.$$

Step 2. To compute $\mathbf{u}^{(k+1)}$. Set $i = 2$.

$$u_1^{(k+1)} = (ru_1^{(k+1/2)} + g_1u_1^{(k)})/\alpha_1$$

while $i \le N-2$, compute

$$d = 1/(\alpha_i\alpha_{i+1} - a_{i+1}c_i),\ A = dr\alpha_{i+1},\ B = -drc_i,$$

$$C = d(\alpha_{i+1}g_i - a_{i+1}c_i),\ D = dc_i(\alpha_{i+1} - g_{i+1}),$$

$$P = -dra_{i+1},\ Q = dr\alpha_i,\ R = da_{i+1}(\alpha_i - g_i),$$

$$S = d(\alpha_ig_{i+1} - a_{i+1}c_i).$$

$$u_i^{(k+1)} = Au_i^{(k+1/2)} + Bu_{i+1}^{(k+1/2)} + Cu_i^{(k)} + Du_{i+1}^{(k)}$$

$$u_{i+1}^{(k+1)} = Pu_i^{(k+1/2)} + Qu_{i+1}^{(k+1/2)} + Ru_i^{(k)} + Su_{i+1}^{(k)}$$

$$i = i + 2.$$

$$u_N^{(k+1)} = (ru_N^{(k+1/2)} + g_Nu_N^{(k)})/\alpha_N.$$

Step 3. Repeat *Step* 1 and *Step* 2 until convergence is achieved.

The computational molecules for the evaluation of $u_i^{(k+1/2)}$, $i = 1, 3, \ldots, N-1$, are given by *Figure* 4.2.1-3.

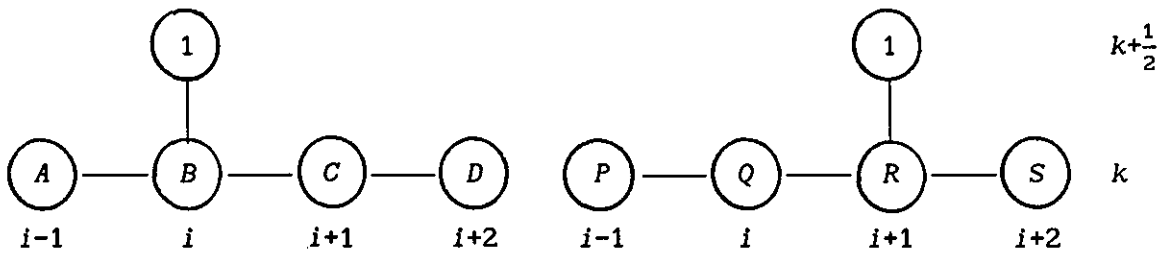


Figure 4.2.1-3 : The computational molecule for $\mathbf{u}^{(k+1/2)}$, AGE-DG.

For the evaluation of $u_i^{(k+1)}$, $i = 2, 4, \ldots, N$, the computational molecules can be presented as in *Figure* 4.2.1-4.

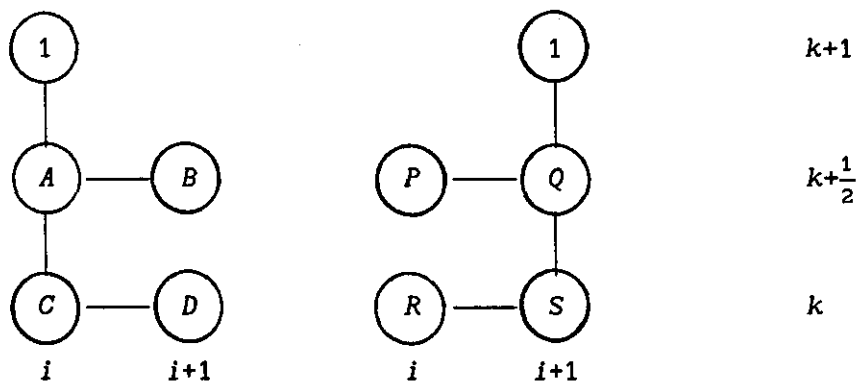


Figure 4.2.1-4 : The computational molecules for $\mathbf{u}^{(k+1)}$, AGE-DG.

By using equations (4.2.1-5) - (4.2.1-6), we next present the algorithm for the AGE-DGGT in computational form as follows.

Algorithm 4.2.1-3: The Computational form of the AGE-DGGT scheme, (COMP-AGE-DGGT), equations (4.2.1-5) - (4.2.1-6).

Set $u_i^{(k)} = 0$, $i = 0, \ldots, N+1$, $a_1 = 0$, $c_N = 0$.

Step 1. To compute $\mathbf{u}^{(k+1/2)}$. Set $i = 1$.

while $i \le N-1$, compute

$$d = 1/(\alpha_i\alpha_{i+1} - a_{i+1}c_i),\ w = 2rd,\ A = a_i(1 - w\alpha_{i+1}),$$

$$B = \alpha_i + w(a_{i+1}c_i - 2\alpha_{i+1}g_i),\ C = wc_i(2g_{i+1} - \alpha_{i+1}),$$

$$D = wc_ic_{i+1},\ E = w(\alpha_{i+1}b_i - c_ib_{i+1}),$$

$$Q = wa_{i+1}(2g_i - \alpha_i),\quad S = c_{i+1}(1 - w\alpha_i),$$

$$R = \alpha_{i+1} + w(a_{i+1}c_i - 2g_{i+1}\alpha_i),\quad P = wa_ia_{i+1},$$

$$T = w(\alpha_i b_{i+1} - a_{i+1}b_i).$$

$$u_i^{(k+1/2)} = Au_{i-1}^{(k)} + Bu_i^{(k)} + Cu_{i+1}^{(k)} + Du_{i+2}^{(k)} + E$$

$$u_{i+1}^{(k+1/2)} = Pu_{i-1}^{(k)} + Qu_i^{(k)} + Ru_{i+1}^{(k)} + Su_{i+2}^{(k)} + T$$

$$i = i + 2.$$

Step 2. To compute $\mathbf{u}^{(k+1)}$ from equation (4.1.3-6). Set $i = 2$.

$$u_1^{(k+1)} = u_1^{(k+1/2)}/\alpha_1$$

while $i \le N-2$, compute

$$d = 1/(\alpha_i\alpha_{i+1} - a_{i+1}c_i)$$

$$A = d\alpha_{i+1},\quad B = -dc_i,\quad C = -da_{i+1},\quad D = d\alpha_i$$

$$u_i^{(k+1)} = Au_i^{(k+1/2)} + Bu_{i+1}^{(k+1/2)}$$

$$u_{i+1}^{(k+1)} = Cu_i^{(k+1/2)} + Du_{i+1}^{(k+1/2)}$$

$$i = i + 2.$$

$$u_N^{(k+1)} = u_N^{(k+1/2)}/\alpha_N.$$

Step 3. Repeat *Step* 1 and *Step* 2 until convergence is achieved.

The computational molecule for this scheme for evaluating $u_i^{(k+1/2)}$, $i = 1, 3, \ldots, N-1$, are given by *Figure* 4.2.1-5.

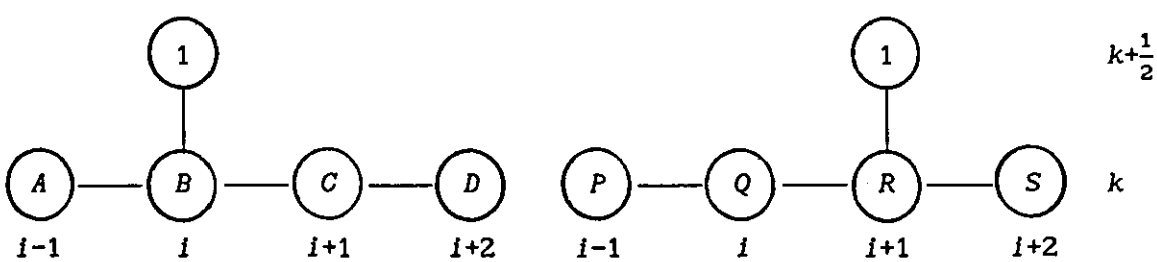


Figure 4.2.1-5 : The computational molecules for $\mathbf{u}^{(k+1/2)}$, AGE-DGGT.

For the evaluation of $u_i^{(k+1)}$, $i = 2, 4, \ldots, N$, the computational molecules can be presented as in *Figure* 4.2.1-6.

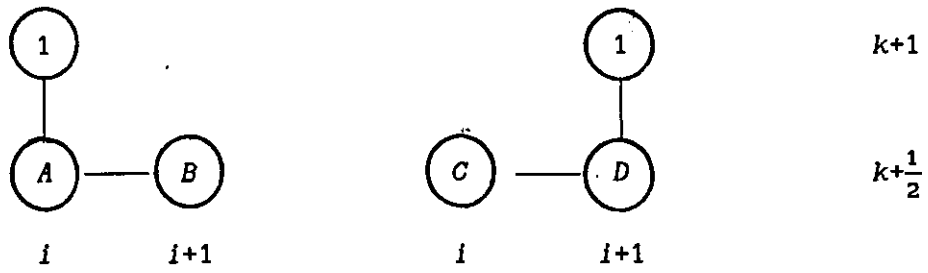


Figure 4.2.1-6 : The computational molecules for $\mathbf{u}^{(k+1)}$, AGE-DGGT.

Finally, we present the computational form of the AGE-PR(1) scheme (COMP-AGE-PR(1)), derived from equations (4.2.1-7) - (4.2.1-8).

Since equation (4.2.1-7) is similar to equation (4.2.1-1), we then have the computational molecules for computing $\mathbf{u}^{(k+1/2)}$ via this scheme similar to the one given in *Figure* 4.2.1-1. We will now derive the computational molecules for computing $\mathbf{u}^{(k+1)}$ from equation (4.2.1-8).

In algorithmic form, equation (4.2.1-8) can be presented as follows.

To compute $\mathbf{u}^{(k+1)}$. Set $i = 2$.

$$u_1^{(k+1)} = (b_1 + \beta_1 u_1^{(k+1/2)} - c_1 u_2^{(k+1/2)})/\alpha_1$$

while $i \le N-2$, compute

$$d = 1/(\alpha_i \alpha_{i+1} - a_{i+1} c_i),\quad A = -da_i \alpha_{i+1},\quad B = d\alpha_{i+1}\beta_i,$$

$$C = -dc_i \beta_{i+1},\quad D = dc_i c_{i+1},\quad E = d(\alpha_{i+1} b_i - c_i b_{i+1}),$$

$$P = da_i a_{i+1},\quad Q = -da_{i+1}\beta_i,\quad R = d\alpha_i \beta_{i+1},$$

$$S = -d\alpha_i c_{i+1},\quad T = d(b_{i+1}\alpha_i - a_{i+1} b_i)$$

$$u_i^{(k+1)} = Au_{i-1}^{(k+1/2)} + Bu_i^{(k+1/2)} + Cu_{i+1}^{(k+1/2)} + Du_{i+2}^{(k+1/2)} + E$$

$$u_{i+1}^{(k+1)} = Pu_{i-1}^{(k+1/2)} + Qu_i^{(k+1/2)} + Ru_{i+1}^{(k+1/2)} + Su_{i+2}^{(k+1/2)} + T$$

$$i = i + 2.$$

$$u_N^{(k+1)} = (b_N - a_N u_{N-1}^{(k+1/2)} + \beta_N u_N^{(k+1/2)})/\alpha_N.$$

The computational molecules for computing $u_i^{(k+1)}$, $i = 1, 3, \ldots, N-1$, via the COMP-AGE-PR(1) can be derived as in *Figure* 4.2.1-7.

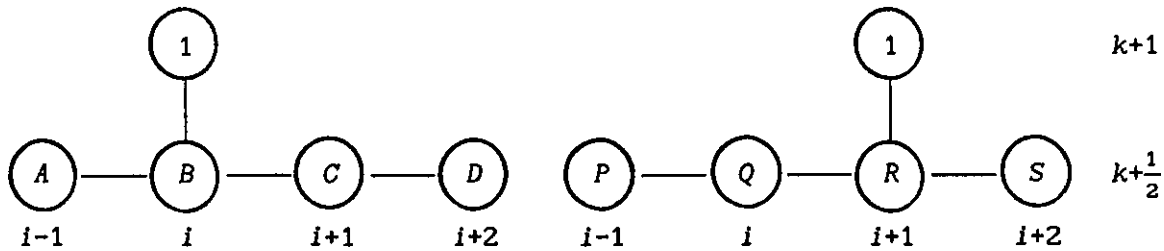


Figure 4.2.1-7 : The computational molecule for $u^{(k+1)}$, AGE-PR(1).

From the figures, we can summarise the total number of nodes in the computational molecules (Comp. Molec.) at each point u_i for each scheme as follows.

	No of Nodes in Comp. Molec.		
The Scheme	Step 1	Step 2	Total
COMP-AGE-PR(2)	4	4	8
COMP-AGE-DG	4	4	8
COMP-AGE-DGGT	4	2	6
COMP-AGE-PR(1)	4	4	8

Table 4.2.1-1 : The number of nodes in the computational molecules at each point u_i

Table 4.2.1-1 clearly shows that the COMP-AGE-DGGT scheme requires a smaller number of nodes in the computational molecules. Since it is obvious that the smaller the number of nodes, then a smaller amount of computational work is needed in each iteration. This scheme, however, uses many intermediate values prior to determining the computational molecules which will require extra computational work. Thus, at this stage, no scheme seems superior over the others. We continue our investigation into a further arrangement of the schemes. This arrangement is called the Coupled AGE (CAGE) scheme, in which both computational sweeps are compressed into a single stage AGE.

4.2.2 Coupled AGE (CAGE) form of PR, Douglas and Guittet

Until now, all the schemes discussed previously have used two stages, i.e., the first to solve for $u^{(k+1/2)}$ followed by the solution for $u^{(k+1)}$. In this section, we will show that the two stage schemes can be combined into a coupled single stage AGE (CAGE) method.

Let us recall the four schemes that have been discussed in Section 4.2.1. The respective CAGE for these schemes can be written as follows. Let μ and ν be the respective eigenvalues of G_1 and G_2.

The CAGE-PR(2) Scheme

$$u^{(k+1)} = (rI + G_2)^{-1}[2r(rI + G_1)^{-1} - I](rI - G_2)u^{(k)} + 2r(rI + G_2)^{-1}(rI + G_1)^{-1}b. \qquad (4.2.2\text{-}1)$$

The iteration matrix, T_r is given by

$$T_r = (rI + G_2)^{-1}[2r(rI + G_1)^{-1} - I](rI - G_2).$$

We now show that the scheme is convergent. Since

$$\|T_r\|_2 = \|(rI + G_2)^{-1}[2r(rI + G_1)^{-1} - I](rI - G_2)\|_2$$

$$= \left|\frac{1}{r + \nu}\left[\frac{2r}{r + \mu} - 1\right](r - \nu)\right|$$

$$\leq \max_{\mu,\nu}\left|\frac{r - \nu}{r + \nu}\right|\left|\frac{r - \mu}{r + \mu}\right| < 1.$$

Hence the scheme is convergent.

The CAGE-DG Scheme

$$u^{(k+1)} = (rI + G_2)^{-1}[r\{I - 2(rI + G_1)^{-1}A\} + G_2]u^{(k)} + 2r(rI + G_2)^{-1}(rI + G_1)^{-1}b$$

$$= (rI + G_2)^{-1}[(rI + G_2) - 2r(rI + G_1)^{-1}A]u^{(k)} + 2r(rI + G_2)^{-1}(rI + G_1)^{-1}b$$

$$= [I - 2r(rI + G_2)^{-1}(rI + G_1)^{-1}A]u^{(k)} + 2r(rI + G_2)^{-1}(rI + G_1)^{-1}b. \qquad (4.2.2\text{-}2)$$

The iteration matrix, T_r is given by

$$T_r = I - 2r(rI + G_2)^{-1}(rI + G_1)^{-1}A.$$

For convergence, we need $\|T_r\|_2 < 1$. Since

$$\begin{aligned}\|T_r\|_2 &= \|I - 2r(rI + G_2)^{-1}(rI + G_1)^{-1}A\|_2 \\ &= \left|1 - \frac{2r}{(r+\mu)(r+\nu)}(\mu + \nu)\right| \\ &= \left|\frac{r^2 - r(\mu+\nu) + \mu\nu}{(r+\mu)(r+\nu)}\right| \\ &= \left|\frac{r-\nu}{r+\nu}\right|\left|\frac{r-\mu}{r+\mu}\right| < 1.\end{aligned}$$

Hence the scheme is convergent.

The CAGE-DGGT Scheme

$$\begin{aligned}\mathbf{u}^{(k+1)} &= [I - 2r(rI + G_2)^{-1}(rI + G_1)^{-1}A]\mathbf{u}^{(k)} \\ &\quad + 2r(rI + G_2)^{-1}(rI + G_1)^{-1}\mathbf{b}. \qquad (4.2.2\text{-}3)\end{aligned}$$

The iteration matrix, T_r is given by

$$T_r = I - 2r(rI + G_2)^{-1}(rI + G_1)^{-1}A.$$

Since the iteration matrix of the CAGE-DGGT scheme is similar to the iteration matrix of the CAGE-DG scheme, then the scheme converges.

The CAGE-PR(1) Scheme

$$\begin{aligned}\mathbf{u}^{(k+1)} &= (rI + G_2)^{-1}(rI - G_1)(rI + G_1)^{-1}(rI - G_2)\mathbf{u}^{(k)} \\ &\quad + (rI + G_2)^{-1}[I + (rI - G_1)(rI + G_1)^{-1}]\mathbf{b}. \qquad (4.2.2\text{-}4)\end{aligned}$$

The iteration matrix, T_r is given by

$$T_r = (rI + G_2)^{-1}(rI - G_1)(rI + G_1)^{-1}(rI - G_2).$$

We now show that the scheme converges. Since

$$\begin{aligned}\|T_r\|_2 &= \|(rI + G_2)^{-1}(rI - G_1)(rI + G_1)^{-1}(rI - G_2)\|_2 \\ &= \left|\frac{1}{r+\nu}(r-\mu)\frac{1}{r+\mu}(r-\nu)\right| \\ &\le \max_{\mu,\nu}\left|\frac{r-\nu}{r+\nu}\right|\left|\frac{r-\mu}{r+\mu}\right| < 1.\end{aligned}$$

Hence the scheme is convergent.

In general, the CAGE method is of the form

$$u^{(k+1)} = T_r u^{(k)} + Cb \tag{4.2.2-5}$$

with T_r, a matrix of the form given by

$$T_r = \begin{bmatrix} X & X & X & & & & & & \\ X & X & X & X & X & & & 0 & \\ X & X & X & X & X & & & & \\ & X & X & X & X & X & X & & \\ & X & X & X & X & X & X & & \\ & \diagdown & \diagdown & \diagdown & \diagdown & \diagdown & \diagdown & \diagdown & \\ & & X & X & X & X & X & X & \\ & & X & X & X & X & X & X & \\ & 0 & & & X & X & X & X & X \\ & & & & X & X & X & X & X \\ & & & & & & X & X & X \end{bmatrix} \tag{4.2.2-6}$$

and C is given as

$$C = \begin{bmatrix} X & X & & & & & & & \\ X & X & X & X & & & & & \\ X & X & X & X & & & 0 & & \\ & X & X & X & X & & & & \\ & X & X & X & X & & & & \\ & \diagdown & \diagdown & \diagdown & \diagdown & \diagdown & & & \\ & & X & X & X & X & & & \\ & & X & X & X & X & & & \\ & 0 & & & X & X & X & X & \\ & & & & X & X & X & X & \\ & & & & & & X & X & \end{bmatrix} \tag{4.2.2-7}$$

where X represents a non-zero element.

The coefficient for each non-zero element can be determined by solving the iteration matrix, T_r and matrix C from each scheme of the CAGE method.

From (4.2.2-6) and (4.2.2-7), we now present the algorithm for the CAGE method, i.e, to compute $u_i^{(k+1)}$ as follows. We commence with the algorithm for the CAGE-PR(2) scheme, i.e., equation (4.2.2-1).

Algorithm 4.2.2-1: The CAGE-PR(2) scheme, equation (4.2.2-1).

Set $u_i^{(k)} = 0$, $i = 0, \ldots, N+1$.

Step 1. To compute $\mathbf{u}^{(k+1)}$. Set $i = 2$.

$$d = 1/(\alpha_1\alpha_2 - a_2c_1),\ w_1 = 2rd,\ E = -w_1c_1/\alpha_1,$$

$$A = \beta_1(w_1\alpha_2 - 1)/\alpha_1,\ B = E\beta_2,\ C = -Ec_2,\ D = w_1\alpha_2/\alpha_1$$

$$u_1^{(k+1)} = Au_1^{(k)} + Bu_2^{(k)} + Cu_3^{(k)} + Db_1 + Eb_2$$

while $i \le N-2$, compute

$$d_1 = 1/(\alpha_{i-1}\alpha_i - a_ic_{i-1}),\ d_2 = 1/(\alpha_i\alpha_{i+1} - a_{i+1}c_i),$$

$$d_3 = 1/(\alpha_{i+1}\alpha_{i+2} - a_{i+2}c_{i+1}),\ w_1 = 2rd_1d_2,\ w_2 = 2rd_2d_3,$$

$$A_1 = w_1a_i\alpha_{i+1},\ A = A_1a_{i-1},\ B = -A_1\beta_{i-1},$$

$$C = \alpha_{i+1}\beta_i(w_1\alpha_{i-1} - d_2) + a_{i+1}c_i(w_2\alpha_{i+2} - d_2),$$

$$D = c_i[d_2(\alpha_{i+1} + \beta_{i+1}) - w_1\alpha_{i-1}\alpha_{i+1} - w_2\alpha_{i+2}\beta_{i+1}],$$

$$K = w_2c_ic_{i+1},\ E = K\beta_{i+2},\ F = -Kc_{i+2},$$

$$G = -w_1a_i\alpha_{i+1},\quad H = w_1\alpha_{i-1}\alpha_{i+1},\ J = -w_2c_i\alpha_{i+2},$$

$$P_1 = w_1a_ia_{i+1},\ P = -P_1a_{i-1},\ Q = P_1\beta_{i-1},$$

$$R = a_{i+1}[d_2(\alpha_i + \beta_i) - w_1\alpha_{i-1}\beta_i - w_2\alpha_i\alpha_{i+2}],$$

$$S = a_{i+1}c_i(w_1\alpha_{i-1} - d_2) + \alpha_i\beta_{i+1}(w_2\alpha_{i+2} - d_2),$$

$$Y = -w_2\alpha_ic_{i+1},\ T = Y\beta_{i+2},\ U = -Yc_{i+2},$$

$$V = w_1a_ia_{i+1},\ W = -w_1a_{i+1}\alpha_{i-1},\ X = w_2\alpha_i\alpha_{i+2},$$

$$u_i^{(k+1)} = Au_{i-2}^{(k)} + Bu_{i-1}^{(k)} + Cu_i^{(k)} + Du_{i+1}^{(k)} + Eu_{i+2}^{(k)} + Fu_{i+3}^{(k)} + Gb_{i-1} + Hb_i + Jb_{i+1} + Kb_{i+2}$$

$$u_{i+1}^{(k+1)} = Pu_{i-2}^{(k)} + Qu_{i-1}^{(k)} + Ru_i^{(k)} + Su_{i+1}^{(k)} + Tu_{i+2}^{(k)} + Uu_{i+3}^{(k)} + Vb_{i-1} + Wb_i + Xb_{i+1} + Yb_{i+2}$$

$$i = i + 2.$$

$$d = 1/(\alpha_{N-1}\alpha_N - a_N c_{N-1}),\ w_1 = 2rd,\ D = -w_1 a_N/\alpha_N,$$

$$A = -Da_{N-1},\ B = D\beta_{N-1},\ C = \beta_N(w_1\alpha_{N-1} - 1)/\alpha_N,\ E = w_1\alpha_{N-1}/\alpha_N$$

$$u_N^{(k+1)} = Au_{N-2}^{(k)} + Bu_{N-1}^{(k)} + Cu_N^{(k)} + Db_{N-1} + Eb_N.$$

Step 2. Repeat *Step* 1 until convergence is achieved.

The following algorithm is for the CAGE-DG and CAGE-GT schemes, since both schemes have similar iteration matrix, T_r and matrix C.

Algorithm 4.2.2-2: The CAGE-DG scheme, equation (4.2.2-2) and the CAGE-DGGT scheme, equation (4.2.2-3).

Set $u_i^{(k)} = 0,\ i = 0, \ldots, N+1.$

Step 1. To compute $\mathbf{u}^{(k+1)}$. Set $i = 2$.

$$d = 1/(\alpha_1\alpha_2 - a_2c_1),\ w = 2rd/\alpha_1\ E = -wc_1,\ C = -Ec_2,$$

$$A = 1 + w(a_2c_1 - 2g_1\alpha_2),\ B = -E(2g_2 - \alpha_2),\ D = w\alpha_2.$$

$$u_1^{(k+1)} = Au_1^{(k)} + Bu_2^{(k)} + Cu_3^{(k)} + Db_1 + Eb_2$$

while $i \le N-2$, compute

$$d_1 = 1/(\alpha_{i-1}\alpha_i - a_i c_{i-1}),\ d_2 = 1/(\alpha_i\alpha_{i+1} - a_{i+1}c_i),$$

$$d_3 = 1/(\alpha_{i+1}\alpha_{i+2} - a_{i+2}c_{i+1}),\ w_1 = 2rd_1d_2,$$

$$w_2 = 2rd_2d_3,\ A_1 = w_1a_i\alpha_{i+1},\ K = w_2c_ic_{i+1},$$

$$A = A_1a_{i-1},\ B = A_1(2g_{i-1} - \alpha_{i-1}),$$

$$C = 1 + w_2a_{i+1}c_i\alpha_{i+2} + w_1\alpha_{i+1}(a_ic_{i-1} - 2\alpha_{i-1}g_i)$$

$$D = c_i[w_2(2\alpha_{i+2}g_{i+1} - a_{i+2}c_{i+1}) - w_1\alpha_{i-1}\alpha_{i+1}],$$

$$E = K(\alpha_{i+2} - 2g_{i+2}),\ F = -Kc_{i+2},\ G = -w_1a_i\alpha_{i+1},$$

$$H = w_1\alpha_{i-1}\alpha_{i+1},\ J = -w_2c_i\alpha_{i+2},\ P_1 = w_1a_ia_{i+1}$$

$$P = -P_1a_{i-1},\ Q = P_1(\alpha_{i-1} - 2g_{i-1}),$$

$$R = a_{i+1}[w_1(2\alpha_{i-1}g_i - a_ic_{i-1}) - w_2\alpha_i\alpha_{i+2}],$$

$$S = 1 + w_1c_ia_{i+1}\alpha_{i-1} + w_2\alpha_i(a_{i+2}c_{i+1} - 2g_{i+1}\alpha_{i+2}),$$

$$Y = -w_2c_{i+1}\alpha_i,\quad T = -Y(2g_{i+2} - \alpha_{i+2}),\ U = -Yc_{i+2},$$

$$V = w_1a_ia_{i+1},\ W = -w_1a_{i+1}\alpha_{i-1},\ X = w_2\alpha_i\alpha_{i+2},$$

$$u_i^{(k+1)} = Au_{i-2}^{(k)} + Bu_{i-1}^{(k)} + Cu_i^{(k)} + Du_{i+1}^{(k)} + Eu_{i+2}^{(k)} + Fu_{i+3}^{(k)}$$
$$+ Gb_{i-1} + Hb_i + Jb_{i+1} + Kb_{i+2}$$
$$u_{i+1}^{(k+1)} = Pu_{i-2}^{(k)} + Qu_{i-1}^{(k)} + Ru_i^{(k)} + Su_{i+1}^{(k)} + Tu_{i+2}^{(k)} + Uu_{i+3}^{(k)}$$
$$+ Vb_{i-1} + Wb_i + Xb_{i+1} + Yb_{i+2}$$

$i = i + 2.$

$d_1 = 1/(\alpha_{N-1}\alpha_N - a_N c_{N-1})$, $w = 2rd_1/\alpha_N$, $D = -wa_N$

$A = -Da_{N-1}$, $B = -D(2g_{N-1} - \alpha_{N-1})$,

$C = 1 + w(a_N c_{N-1} - 2g_N\alpha_{N-1})$, $E = w\alpha_{N-1}$

$u_N^{(k+1)} = Au_{N-2}^{(k)} + Bu_{N-1}^{(k)} + Cu_N^{(k)} + Db_{N-1} + Eb_N.$

Step 2. Repeat *Step* 1 until convergence is achieved.

Finally, we present the algorithm for the CAGE-PR(1) scheme.

Algorithm 4.2.2-3: The CAGE-PR(1) scheme, equation (4.2.2-4).

Set $u_i^{(k)} = 0$, $i = 0, \ldots, N+1$.

Step 1. To compute $u^{(k+1)}$. Set $i = 2$.

$d = 1/(\alpha_1\alpha_2 - a_2c_1)$, $w = d/\alpha_1$, $w_1 = wc_1(\alpha_1 + \beta_1)$,

$A = w\beta_1(\alpha_2\beta_1 + a_2c_1)$, $B = -w_1\beta_2$, $C = w_1c_2$,

$D = (1 + da_2c_1 + d\alpha_2\beta_1)/\alpha_1$, $E = -dc_1(\alpha_1 + \beta_1)/\alpha_1$

$u_1^{(k+1)} = Au_1^{(k)} + Bu_2^{(k)} + Cu_3^{(k)} + Db_1 + Eb_2$

while $i \le N-2$, compute

$d_1 = 1/(\alpha_{i-1}\alpha_i - a_ic_{i-1})$, $d_2 = 1/(\alpha_i\alpha_{i+1} - a_{i+1}c_i)$,

$d_3 = 1/(\alpha_{i+1}\alpha_{i+2} - a_{i+2}c_{i+1})$, $w_1 = d_1d_2$, $w_2 = d_2d_3$,

$A_1 = w_1a_i\alpha_{i+1}$, $A_2 = A_1(\alpha_i + \beta_i)$, $A = A_2a_{i-1}$,

$B = -A_2\beta_{i-1}$, $C_1 = \alpha_{i-1}\beta_i + a_ic_{i-1}$,

$D_1 = \alpha_{i+2}\beta_{i+1} + a_{i+2}c_{i+1}$, $C = w_1\alpha_{i+1}\beta_iC_1 + w_2a_{i+1}c_iD_1$,

$D = -c_i(w_1\alpha_{i+1}C_1 + w_2\beta_{i+1}D_1)$, $E_1 = w_2c_ic_{i+1}$,

$E_2 = E_1(\alpha_{i+1} + \beta_{i+1})$, $E = E_2\beta_{i+2}$, $F = -E_2c_{i+2}$,

$P = -A_2$, $Q = d_2\alpha_{i+1}(1 + d_1a_ic_{i-1} + d_1\alpha_{i-1}\beta_i)$

$$R = -d_2c_i(1 + d_3a_{i+2}c_{i+1} + d_3\alpha_{i+2}\beta_{i+1}),\ S = E_2$$

$$u_i^{(k+1)} = Au_{i-2}^{(k)} + Bu_{i-1}^{(k)} + Cu_i^{(k)} + Du_{i+1}^{(k)} + Eu_{i+2}^{(k)} + Fu_{i+3}^{(k)} + Pb_{i-1} + Qb_i + Rb_{i+1} + Sb_{i+2}$$

$$A_1 = w_1a_ia_{i+1},\ A_2 = A_1(\alpha_i + \beta_i),\ A = -A_2a_{i-1},$$

$$B = A_2\beta_{i-1},\ C = -a_{i+1}(w_1\beta_iC_1 + w_2\alpha_iD_1),$$

$$D = w_1a_{i+1}c_iC_1 + w_2\alpha_i\beta_{i+1}D_1,\ E_1 = w_2\alpha_ic_{i+1},$$

$$E_2 = E_1(\alpha_{i+1} + \beta_{i+1}),\ E = -E_2\beta_{i+2},\ F = E_2c_{i+2},$$

$$P = A_2,\ Q = -d_2a_{i+1}(1 + d_1a_ic_{i-1} + d_1\alpha_{i-1}\beta_i)$$

$$R = d_2\alpha_i(1 + d_3a_{i+2}c_{i+1} + d_3\alpha_{i+2}\beta_{i+1}),\ S = -E_2$$

$$u_{i+1}^{(k+1)} = Au_{i-2}^{(k)} + Bu_{i-1}^{(k)} + Cu_i^{(k)} + Du_{i+1}^{(k)} + Eu_{i+2}^{(k)} + Fu_{i+3}^{(k)} + Pb_{i-1} + Qb_i + Rb_{i+1} + Sb_{i+2}$$

$$i = i + 2.$$

$$d = 1/(\alpha_{N-1}\alpha_N - a_Nc_{N-1}),\ w = d/\alpha_N,\ w_1 = wa_N(\alpha_N + \beta_N),$$

$$A = w_1a_{N-1},\ B = -w_1\beta_{N-1},\ C = w\beta_N(a_Nc_{N-1} + \alpha_{N-1}\beta_N),$$

$$D = -da_N(\alpha_N + \beta_N)/\alpha_N,\ E = (1 + da_Nc_{N-1} + d\alpha_N\beta_{N-1})/\alpha_N$$

$$u_N^{(k+1)} = Au_{N-2}^{(k)} + Bu_{N-1}^{(k)} + Cu_N^{(k)} + Db_{N-1} + Eb_N.$$

Step 2. Repeat *Step* 1 until convergence is achieved.

The algorithms 4.2.2-1, 4.2.2-2 and 4.2.2-3 show that, in order to determine the coefficient for each node and element b_i, we need to compute many intermediate values. Also, extra work is needed in each iteration if g_i (for a given problem) depends on the solution vector u_i. However, in the case where g_i is independent of u_i, all the intermediate values can be computed outside the loop and thereby save time in each iteration. Moreover, from these algorithms, it can be deduced that the computational molecule for the CAGE method for large N, is given by the 6 nodal formulae, i.e,

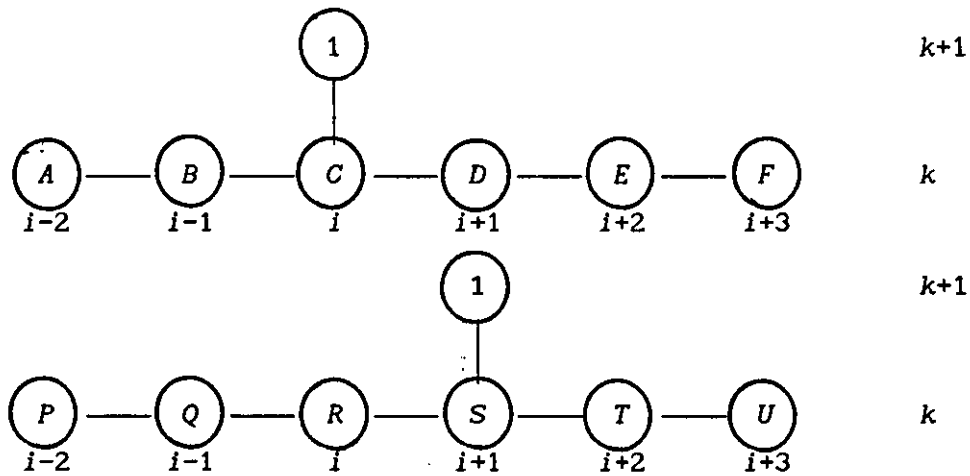


Figure 4.2.2-1: Computational molecule for the one step CAGE method.

Table 4.2.2-1 below shows the number of nodes for each scheme presented in Section 4.2.1 compared to the scheme derived in the form of the CAGE method above.

Comp. Molec. Scheme	No. of Nodes	The CAGE method	No. of Nodes
COMP-AGE-PR(2)	8	CAGE-PR(2)	6
COMP-AGE-DG	8	CAGE-DG	6
COMP-AGE-DGGT	6	CAGE-DGGT	6
COMP-AGE-PR(1)	8	CAGE-PR(1)	6

Table 4.2.2-1 : The number of nodes in the computational molecules

Table 4.2.2-1 indicates that there is a 25% saving in computational work in the CAGE method over the COMP-AGE schemes, except the case for CAGE-DGGT over COMP-AGE-DGGT. Although, many intermediate values are needed prior to calculating the coefficients for the solution vector **u**, these gains make the CAGE method better than the two step COMP-AGE schemes.

Although *Table* 4.2.2-1 does not include the computation of the vector **b**, it is obvious that the CAGE-PR(1) scheme requires more work compared to the CAGE-PR(2), CAGE-DG and CAGE-DGGT schemes.

In the CAGE-PR(2) scheme, the evaluation of the coefficients A, B, C, ... etc of the matrix T_r is more difficult than in the CAGE-DG scheme. Thus, at this stage, we may consider the CAGE-DG or CAGE-DGGT schemes to be the best choice.

In the next section, we will consider another possible time saving form of presenting the AGE method, which is called the smart AGE (SMAGE) in PR form.

4.2.3 Smart AGE (SMAGE) in PR form

In this section, we will investigate an alternative approach of evaluating the AGE method based on the AGE-PR(2) scheme. This new scheme is called Smart AGE (SMAGE), and is predicted to save time as the idea involved is to eliminate evaluating two similar terms on the right hand sides of the AGE-PR(2) scheme.

If we recall the AGE-PR(2) scheme in explicit form, we have, i.e.,

$$\mathbf{u}^{(k+1/2)} = (rI + G_1)^{-1}[\mathbf{b} + (rI - G_2)\mathbf{u}^{(k)}] \qquad (4.2.4\text{-}1)$$

$$\mathbf{u}^{(k+1)} = (rI + G_2)^{-1}[2r\mathbf{u}^{(k+1/2)} - (rI - G_2)\mathbf{u}^{(k)}]. \qquad (4.2.4\text{-}2)$$

The two similar terms in these equations is $(rI - G_2)\mathbf{u}^{(k)}$ and we let this term be φ. The evaluation and saving of φ depend upon the problems, i.e., either linear or nonlinear. The matrix A derived from the linear problems give either a constant or variable diagonal element, whereas from the nonlinear problems, this element is variable.

The SMAGE-LINEAR scheme for the linear problems is envisaged to save 2 multiplications and 1 addition for every iteration. Whilst, the SMAGE-NONLINEAR scheme for the nonlinear problems is expected to save 1 multiplication and 1 addition for every iteration.

The algorithms for these schemes are presented as follows:

Algorithm 4.2.3-1: The SMAGE-LINEAR scheme.

Set $u_i^{(k)} = 0$, $i = 0, \ldots, N+1$, $a_1 = 0$, $c_N = 0$,

$\alpha_i = r + g_i$, $\beta_i = r - g_i$, $i = 1, 2, \ldots, N$.

Step 1. To compute $\varphi = (rI - G_2)u^{(0)}$. set $i = 1$.

while $i \le N-1$, compute

$$\varphi_i = - c_i u_{i-1}^{(0)} + \beta_i u_i^{(0)}$$

$$\varphi_{i+1} = \beta_{i+1} u_{i+1}^{(0)} - a_{i+1} u_{i+2}^{(0)}$$

$$i = i + 2.$$

Step 2. To compute $u^{(k+1/2)}$. Set $i = 1$.

while $i \le N-1$, compute

$$r_1 = b_i + \varphi_i, \quad r_2 = b_{i+1} + \varphi_{i+1}$$

$$d = 1/(\alpha_i \alpha_{i+1} - a_{i+1} c_i)$$

$$u_i^{(k+1/2)} = (\alpha_{i+1} r_1 - c_i r_2) d$$

$$u_{i+1}^{(k+1/2)} = (-a_{i+1} r_1 + \alpha_i r_2) d$$

$$i = i + 2.$$

Step 3. For $i = 1, 2, \ldots, N$, compute $\phi_i = - \varphi_i + 2r u_i^{(k+1/2)}$.

Step 4. To compute $u^{(k+1)}$.

$$u_1^{(k+1)} = \phi_1 / \alpha_1$$

while $i \le N-2$, compute

$$d = 1/(\alpha_i \alpha_{i+1} - a_{i+1} c_i)$$

$$u_i^{(k+1)} = (\alpha_{i+1} \phi_i - c_i \phi_{i+1}) d$$

$$u_{i+1}^{(k+1)} = (- a_{i+1} \phi_i + \alpha_i \phi_{i+1}) d$$

$$i = i + 2.$$

$$u_N^{(k+1)} = \phi_N / \alpha_N.$$

Step 5. For $i = 1, 2, \ldots, N$, compute $\varphi_i = - \phi_i + 2r u_i^{(k+1)}$.

Step 6. Repeat *Step* 2 to *Step* 5 until convergence is achieved.

Algorithm 4.2.3-2: The SMAGE-NONLINEAR scheme.

Set $u_i^{(k)} = 0,\ i = 0, \ldots, N+1,\ a_1 = 0,\ c_N = 0,$

Step 1. To compute g_i, α_i and β_i.

for $i = 1$ to N

compute g_i, $\alpha_i = r + g_i$, $\beta_i = r - g_i$,

Step 2. To compute $\varphi = (rI - G_2)u^{(k)}$. set $i = 1$.

while $i \le N-1$, compute

$$\varphi_i = - c_i u_{i-1}^{(0)} + \beta_i u_i^{(0)}$$

$$\varphi_{i+1} = \beta_{i+1} u_{i+1}^{(0)} - a_{i+1} u_{i+2}^{(0)}$$

$$i = i + 2.$$

Step 3. To compute $u^{(k+1/2)}$. Set $i = 1$.

while $i \le N-1$, compute

$$r_1 = b_i + \varphi_i,\ r_2 = b_{i+1} + \varphi_{i+1}$$

$$d = 1/(\alpha_i \alpha_{i+1} - a_{i+1} c_i)$$

$$u_i^{(k+1/2)} = (\alpha_{i+1} r_1 - c_i r_2)d$$

$$u_{i+1}^{(k+1/2)} = (-a_{i+1} r_1 + \alpha_i r_2)d$$

$$i = i + 2.$$

Step 4. For $i = 1, 2, \ldots, N$, compute $\phi_i = - \varphi_i + 2r u_i^{(k+1/2)}$.

Step 5. To compute $u^{(k+1)}$.

$$u_1^{(k+1)} = \phi_1/\alpha_1$$

while $i \le N-2$, compute

$$d = 1/(\alpha_i \alpha_{i+1} - a_{i+1} c_i)$$

$$u_i^{(k+1)} = (\alpha_{i+1} \phi_i - c_i \phi_{i+1})d$$

$$u_{i+1}^{(k+1)} = (- a_{i+1} \phi_i + \alpha_i \phi_{i+1})d$$

$$i = i + 2.$$

$$u_N^{(k+1)} = \phi_N/\alpha_N.$$

Step 6. Repeat *Step* 1 to *Step* 5 until convergence is achieved.

It should be noticed that the Algorithms 4.2.3-1 and 4.2.3-2 differ in some aspects. For the linear problems, since the values of g_i, α_i and β_i are unchanged for every iteration, these values can be computed outside the iteration loop, i.e, in the SMAGE-LINEAR scheme. As a result, the first set of φ_i is also computed the loop. Thus, the routine work will start from *Step* 2.

For the nonlinear problems, the values of g_i, α_i and β_i varies at every iteration. Thus, the computations for these values must be kept within the iteration loop. Consequently, the computation of φ_i must also be carried out within the loop. The next calculation of φ_i is performed by a simple operation as outlined in the SMAGE-NONLINEAR scheme.

4.2.4 Experimental results

The COMP-AGE, CAGE and SMAGE schemes of Sections 4.2.1, 4.2.2 and 4.2.3 were investigated experimentally on three problems, two linear and one mildly nonlinear and the results obtained are presented as follows. Each problem will be concerned with the computational complexity and the speed (CPU time) for the schemes discussed in this chapter. The time is measured initially from the initialisation of $\mathbf{u}^0$ until the solution converges to $\mathbf{u}^{(k)}$, where k is the number of iterations.

Problem 1 - A Linear Problem

$$-U'' + \rho U = (\rho+1)(\sin x + \cos x), \qquad 0 \le x \le \frac{\pi}{2},$$

$$U(0) = 1,\ U(\frac{\pi}{2}) = 1. \qquad h = \pi/2(N+1).$$

The exact solution is

$$U(x) = \sin x + \cos x.$$

The matrix A is given by

$$A = \begin{bmatrix} 2g & -1 & & & \\ -1 & 2g & -1 & & 0 \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2g & -1 \\ 0 & & & -1 & 2g \end{bmatrix}$$

where $g = 1 + 0.5\rho h^2$.

1) The computational complexity.

Since g is independent of the solution vector **u** and is a constant, all intermediate calculations may be computed outside the iteration loop. The evaluation of vector **b** at each point, can be assigned as an array so that it will save 1 addition and 2 multiplications per iteration. Thus, the work involves only the addition and multiplication of each node and the addition of an element of an array.

It should be noticed that for the three CAGE schemes, the amount of computational work is the same, i.e., 6 multiplications and 6 additions. Thus, it is sufficient to tabulate the amount of work for the CAGE-PR(2) scheme in *Table* 4.2.4-1. The total operations for other schemes may be derived in the same way.

The results for large N are tabulated in *Table* 4.2.4-1.

The Scheme	Multiplication	Addition	Overall
COMP-AGE-PR(2)	8*N*	7*N*	15*N*
COMP-AGE-DG	8*N*	7*N*	15*N*
COMP-AGE-DGGT	6*N*	5*N*	11*N*
COMP-AGE-PR(1)	8*N*	8*N*	16*N*
CAGE-PR(2)	6*N*	6*N*	12*N*
SMAGE-LINEAR	7*N*	6*N*	13*N*

Table 4.2.4-1: Problem 1, the amount of work per iteration

2) The CPU Time.

Table 4.2.4-2 shows the times taken for each prescribed scheme for solving Problem 1.

		ρ=70	The schemes [ ] with times taken in sec.					
N	r	iter	[1]	[2]	[3]	[4]	[5]	[6]
20	0.60	10	0.04	0.04	0.03	0.04	0.03	0.04
40	0.30	19	0.15	0.15	0.11	0.16	0.12	0.14
80	0.11	37	0.59	0.57	0.44	0.61	0.45	0.56
160	0.05	73	2.30	2.24	1.75	2.43	1.78	2.15
320	0.02	140	8.74	8.63	6.69	9.21	6.80	8.30

Table 4.2.4-2: Problem 1, the CPU time taken for each scheme

Notation: [1]: COMP-AGE-PR(2), [2]:COMP-AGE-DG, [3]: COMP-AGE-DGGT, [4]: COMP-AGE-PR(1), [5]: CAGE-PR(2), [6]: SMAGE-LINEAR.

Problem 2 - A Linear Problem

$$U'' - (1 + x^2)U = -1, \qquad -1 \le x \le 1,$$

$$U(-1) = 0, \; U(1) = 0. \qquad h = 2/(N+1).$$

The computed solution is obtained when

$$\| \mathbf{r} \| = \| \mathbf{b} - A\mathbf{u}^{(k+1)} \| < \varepsilon.$$

The matrix A for this problem is given by

$$A = \begin{bmatrix} 2g_1 & -1 & & & \\ -1 & 2g_2 & -1 & & 0 \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2g_{N-1} & -1 \\ & 0 & & -1 & 2g_N \end{bmatrix} \qquad (4.2.4\text{-}1)$$

where $g_i = 1 + 0.5h^2(1 + x_i^2)$, $1 \le i \le N$.

1) The computational complexity.

Now g_i is independent of the solution vector **u** but varies as x_i takes different values. The value g_1, however, can be computed outside the iteration loop to save time. Hence, the values α_i and β_i may also be computed outside the loop. All other intermediate calculations may be kept within the loop to avoid the substantial use of arrays. Thus, the total work is the amount of work in *Table* 4.2.4-1 together with the calculations for the intermediate values. The amount of work at each iteration for large N is tabulated in *Table* 4.2.4-3.

For these particular problems, the CAGE-PR(2) is found to give the best time compared to other CAGE schemes. We would expect a similar amount of work for every CAGE scheme, however, due the different style of presentation for these scheme, the results are different. Thus, it is sufficient to tabulate the results for the CAGE-PR(2) scheme.

Problem 2 is linear with g_i variable. Hence, for comparison, we use the SMAGE-LINEAR scheme.

The Scheme	Multiplication	Addition	Overall
COMP-AGE-PR(2)	19.5N	11N	30.5N
COMP-AGE-DG	21N	14N	35N
COMP-AGE-DGGT	16.5N	11N	27.5N
COMP-AGE-PR(1)	20N	11N	31N
CAGE-PR(2)	16.5N	24.5N	41N
SMAGE-LINEAR	9N	7N	16N

Table 4.2.4-3: Problem 2, the amount of work per iteration

2) The CPU Time.

Table 4.2.4-4 shows the times taken for each prescribed scheme for solving Problem 2. The notation for the schemes as in *Table* 4.2.4-2.

			The Schemes [ ] with times taken in Sec.					
N	r	iter	[1]	[2]	[3]	[4]	[5]	[6]
20	0.31	22	0.23	0.25	0.21	0.22	0.25	0.13
40	0.16	39	0.79	0.89	0.73	0.78	0.92	0.46
80	0.08	70	2.88	3.19	2.61	2.85	3.34	1.65
160	0.04	132	10.77	12.06	9.85	10.67	12.76	6.22
320	0.02	248	40.69	45.59	37.15	40.32	47.63	23.58

Table 4.2.4-4: Problem 2, the CPU time taken for each scheme

Problem 3 - A Mildly Nonlinear Problem.

$$-U'' + \frac{3}{2}U^2 = 0, \qquad 0 \le x \le 1,$$

$$U(0) = 4, \; U(1) = 1. \qquad h = 1/(N+1).$$

The exact solution to this problem is given by

$$U(x) = 4/(1+x)^2.$$

The matrix A is given as in (4.2.4-1), where

$$g_i = 1 + 0.75h^2u_i, \; 1 \le i \le N.$$

1) The computational complexity.

Since g_i depends on the solution u_i, all intermediate calculations must be kept within the iteration loop. Consequently, a little more extra work is needed compared to the amount of work for Problem 2. This work comes from the computation of g_i, α_i and β_i. For this problem, a critical assessment of the computational work required is necessary to determine which of the schemes is the most efficient.

The CAGE-PR(2) scheme again is found to have the fastest CPU time among the CAGE schemes. Hence, the results for this scheme is displayed in *Tables* 4.2.4-5 and 4.2.4-6. Since problem 3 is nonlinear, we then use the SMAGE-NONLINEAR scheme for the amount of work and time.

Table 4.2.4-5 shows the amount of work needed for each iteration for solving Problem 3.

The Scheme	Multiplication	Addition	Overall
COMP-AGE-PR(2)	20.5*N*	14*N*	34.5*N*
COMP-AGE-DG	22*N*	16*N*	38*N*
COMP-AGE-DGGT	17.5*N*	13*N*	30.5*N*
COMP-AGE-PR(1)	21*N*	14*N*	35*N*
CAGE-PR(2)	13.5*N*	26.5*N*	40*N*
SMAGE-NONLINEAR	9*N*	9*N*	18*N*

Table 4.2.4-5: Problem 3, the amount of work per iteration

2) The CPU Time.

Table 4.2.4-6 shows the CPU time taken for each prescribed scheme for solving Problem 3. For the notation of the schemes, please refer to *Table* 4.2.4-2.

			The Schemes [ ] with times taken in Sec.					
N	*r*	iter	[1]	[2]	[3]	[4]	[5]	[6]
20	0.32	35	0.36	0.39	0.31	0.34	0.43	0.21
40	0.17	72	1.46	1.59	1.28	1.43	1.76	0.87
80	0.09	143	5.80	6.37	5.12	5.69	7.02	3.43
160	0.05	286	23.10	25.29	20.38	22.93	28.12	13.58
320	0.03	636	103.54	114.50	90.50	102.33	125.50	60.62

Table 4.2.4-6: Problem 3, the CPU time taken for each scheme

The results show the agreement between the computational work and the CPU time for each scheme. First, we outline the results for the CAGE schemes.

By inspections, the CAGE-PR(2) scheme gives a better performance in terms of computations and CPU time compared with the other CAGE schemes. However, the best CPU time is only shown for the linear problem where g is a constant. As many intermediate variables involved prior to the computation the solution vector **u**, the scheme needs more CPU time and becomes less competitive compared with other scheme. However, the CAGE-DG and CAGE-DGGT schemes seem more suitable to be used with the Richardson and Chebyshev semi-iterative methods. Thus, for the CAGE method, we may consider either the CAGE-DG or CAGE-DGGT scheme for the multi-parameter case which will be discussed later in Chapter 5.

The COMP-AGE-DGGT scheme gives the best CPU time for solving linear problems. However, its computation is quite cumbersome since we need many intermediate variables prior to the use of the scheme. Hence, we might discard this scheme for its laborious computational effort.

The SMAGE scheme is competitive as it is simple to implement and just falls slightly behind the COMP-AGE-DGGT scheme in terms of CPU time when solving linear problems with g, a constant. However, the scheme takes less work and gives the best CPU time when g is variable, as shown in Problems 2 and 3. The setback for the SMAGE scheme is that it is not suitable for the solution with multi-parameters, i.e., to be used with the Richardson or Chebyshev semi-iterative methods. Thus, the scheme is highly recommended for the solution with a single parameter.

4.3 Summary

The aim of this chapter is to find alternative forms of presenting the AGE method so that it can be used to solve the multi-dimensional problems.

The standard form of PR, i.e., the AGE-PR(1) scheme is known to be limited as it will only serve to solve the one dimensional problem. This equation, when rewritten in generalised form results in the AGE-PR(2) scheme.

This approach, together with the strategy suggested by Douglas and Guittet, present new schemes which are analysed theoretically and tested for convergence. The results show that all the schemes give a similar rate of convergence when $\omega = 2$, i.e., all the schemes are identical.

By comparing the computational complexity for each scheme, the AGE-PR(1) scheme has less computations although as stated previously, it is limited in application. The AGE-DG scheme can be extended to solve the two and three dimensional problems. The scheme has been shown to have slightly less computational work over the AGE-DGGT scheme and it is easier to implement. Also, all these formulae can be written in a compact form, i.e., in the form of a computational molecule. Thus, we might consider the AGE-DG scheme is the best choice.

Since we have discarded the PR form, our main interest now lies only in the COMP-AGE-DG and COMP-AGE-DGGT schemes. It is found that the COMP-AGE-DGGT scheme gives a smaller number of nodes and thus better CPU time but with more complex computational work. Hence, we again might consider the AGE-DG scheme as being more competitive over the other schemes.

The formulae above, when rewritten in a single stage or coupled form, produces the CAGE formula. Based on the fact that the CAGE method for the schemes give a similar matrix, one would expect that the CPU times would be the same. Experimentally, this is not true and the CAGE-PR(2) scheme is shown to give a better time.

The obvious reason is, more intermediate variables are needed in the other CAGE schemes compared to the CAGE-PR(2) scheme. The results conclude the CAGE method is not competitive to solve the one dimensional problem, by a single parameter, sequentially. However, the CAGE-DG and CAGE-GT schemes can easily be combined with other methods such as the Richardson method to form a good second order method.

Although the SMAGE scheme shows a significant improved CPU time, it can only be considered for solving the one dimensional problem. The scheme which is based on the AGE-PR(2) scheme cannot be extended to solve a problem with higher dimensions. Moreover, this form is not suitable to be used with a second order method.

Hence, for the recommendations, we may consider the AGE-DG or SMAGE scheme for solving the one dimensional problem with a single parameter. For the solution with multi-parameters, the CAGE-DG or CAGE-DGGT schemes are recommended to be used with the second order methods.

CHAPTER 5

MULTI-PARAMETER ITERATIVE METHODS FOR ODE's

5.1 The application of multi-parameters to iterative methods

The AGE method and the variation of schemes that have been discussed so far, were fully concerned with the determination of a single optimal iteration parameter, i.e, the stationary case. Young [1971] has shown that in the nonstationary case, i.e., the application of more than one iteration parameter, the Alternating Direction Implicit (ADI) in Peaceman-Rachford form to solve the two dimensional problem, yields a faster rate of convergence. Since the AGE method is analogous to the ADI method, then it may be worth while to investigate a similar strategy to achieve a better convergence.

In the non stationary case, one allows the iteration parameter to vary from one iteration to the next. However, a number of parameters are used in a cycle in order to determine the least number of iterations. For example, a cycle of four are made up of parameters r_1, r_2, r_3 and r_4. These parameters, will be used in turn in the first four iterations, the next four iterations and so on. The convergence may be achieved without having to use all the parameters in a cycle, and if this happens then there is no need to complete the cycle as the solution has converged.

We now seek to analyse the convergence properties for the application of multi-parameters in the AGE method.

Let us recall the AGE-PR(1) scheme given in Chapter 4, by the equations (4.1.1-1) - (4.1.1-2),

$$(rI + G_1)u^{(k+1/2)} = b + (rI - G_2)u^{(k)} \tag{5.1-1}$$

$$(rI + G_2)u^{(k+1)} = b + (rI - G_1)u^{(k+1/2)} \tag{5.1-2}$$

or explicitly as

$$u^{(k+1/2)} = (rI + G_1)^{-1}[b + (rI - G_2)u^{(k)}] \tag{5.1-3}$$

$$u^{(k+1)} = (rI + G_2)^{-1}[b + (rI - G_1)u^{(k+1/2)}]. \tag{5.1-4}$$

Now, let us introduce an iteration parameter, r_k. Then, for any $r_k > 0$, we have

$$(r_k I + G_1)u^{(k+1/2)} = b + (r_k I - G_2)u^{(k)} \tag{5.1-5}$$

$$(r_k I + G_2)u^{(k+1)} = b + (r_k I - G_1)u^{(k+1/2)} \tag{5.1-6}$$

or explicitly as

$$u^{(k+1/2)} = (r_k I + G_1)^{-1}[b + (r_k I - G_2)u^{(k)}] \tag{5.1-7}$$

$$u^{(k+1)} = (r_k I + G_2)^{-1}[b + (r_k I - G_1)u^{(k+1/2)}]. \tag{5.1-8}$$

with the iteration matrix, T_{r_k}, as

$$T_{r_k} = (r_k I + G_2)^{-1}(r_k I - G_1)(r_k I + G_1)^{-1}(r_k I - G_2). \tag{5.1-9}$$

Let μ and ν be the maximum eigenvalues of G_1 and G_2 respectively. Then, assuming commutativity, we have $(r_k I + G_1)^{-1}v = (r_k + \mu)^{-1}v$, $(r_k I + G_2)^{-1}v = (r_k + \nu)^{-1}v$, it follows that

$$T_{r_k} v = \left(\frac{(r_k - \mu)(r_k - \nu)}{(r_k + \mu)(r_k + \nu)} \right) v \tag{5.1-10}$$

where v is an eigenvector of $\|T_{r_k}\|$ for any r_k. Hence, for any set of parameters $r_1, r_2, \ldots, r_m$ we have

$$\left(\prod_{k=1}^{m} T_{r_k} \right) v = \left(\prod_{k=1}^{m} \frac{(r_k - \mu)(r_k - \nu)}{(r_k + \mu)(r_k + \nu)} \right) v. \tag{5.1-11}$$

Thus, v is an eigenvector of the matrix $\prod_{k=1}^{m} T_{r_k}$ which corresponds to the AGE method for m iterations using $r_1, r_2, \ldots, r_m$. If μ and ν lies in the ranges $0 < a \le \mu \le b$, $0 < a \le \nu \le b$, we have

$$\rho\left(\prod_{k=1}^{m} T_{r_k} \right) \leq \max_{\substack{a\leq\mu\leq b \\ a\leq\nu\leq b}} \left(\prod_{k=1}^{m} \left| \frac{(r_k - \mu)(r_k - \nu)}{(r_k + \mu)(r_k + \nu)} \right| \right)$$

$$= \max_{a\leq\gamma\leq b} \left(\prod_{k=1}^{m} \left| \frac{(r_k - \gamma)}{(r_k + \gamma)} \right| \right)^2$$

$$= \phi(a, b: r_1, r_2, \ldots, r_m). \qquad (5.1\text{-}12)$$

Since each factor of the product

$$Q_m(\gamma, r_k) = \prod_{k=1}^{m} \left| \frac{(r_k - \gamma)}{(r_k + \gamma)} \right| \qquad (5.1\text{-}13)$$

is less then unity, then

$$\rho\left(\prod_{k=1}^{m} T_{r_k} \right) < 1. \qquad (5.1\text{-}14)$$

Hence, the multi-parameter AGE method is convergent.

The analysis for the convergence of the AGE-PR(2), AGE-DG, AGE-DGGT and other schemes can be shown in a similar way.

The task now is to minimise the function $\phi(a, b: r_1, r_2, \ldots, r_m)$, i.e., to speed up the rate of convergence by a careful choice of r_k. In the next section, we will discuss the existing multi-parameter formulae that are used to minimise the analogous ADI method.

5.1.1 The existing multi-parameter ADI formula

The problem of minimising the function $\phi(a, b: r_1, r_2, \ldots, r_m)$ was initially solved analytically in terms of Fourier analysis by Peaceman-Rachford [1955]. These parameters were used in solving an elliptic equation in a square region bounded by Dirichlet boundary conditions via the ADI method. In the ADI method, the matrix A derived from the equation, is split into H and V, giving both H and V as tridiagonal matrices. The eigenvalues of H and V are found to be

$$\mu = 4\ sin^2\frac{\pi h}{2} \text{ and } \nu = 4\ cos^2\frac{\pi h}{2}. \tag{5.1.1-1}$$

The Peaceman-Rachford (PR) parameters are given by

$$r_k^{PR} = \nu\left(\frac{\mu}{\nu}\right)^{(2k-1)/(2m)}, \quad k = 1, 2, \ldots, m. \tag{5.1.1-2}$$

Evidently by (5.1.1-2), we have

$$\mu < r_m^{PR} < \ldots < r_2^{PR} < r_1^{PR} < \nu. \tag{5.1.1-3}$$

With this set of parameters we can show that

$$\rho\left(\prod_{k=1}^{m} T_{r_k^{PR}}\right) \le \left(\frac{1 - \left(\frac{\mu}{\nu}\right)^{1/(2m)}}{1 + \left(\frac{\mu}{\nu}\right)^{1/(2m)}}\right)^2. \tag{5.1.1-4}$$

We now determine the rate of convergence for the PR parameters. For μ and ν in (5.1.1-2), the PR parameters are given by

$$r_k^{PR} = 4\ cos^2\frac{\pi h}{2}\left(tan^2\frac{\pi h}{2}\right)^{(2k-1)/(2m)}, \quad k = 1, 2, \ldots, m. \tag{5.1.1-5}$$

Substituting r_k^{PR} in (5.1-12), gives

$$\begin{aligned}\rho\left(\prod_{k=1}^{m} T_{r_k^{PR}}\right) &\le \left(\frac{1 - \left(tan\frac{\pi h}{2}\right)^{1/m}}{1 + \left(tan\frac{\pi h}{2}\right)^{1/m}}\right)^2 \\ &\sim \left(\frac{1 - \left(\frac{\pi h}{2}\right)^{1/m}}{1 + \left(\frac{\pi h}{2}\right)^{1/m}}\right)^2 \\ &\sim 1 - 4\left(\frac{\pi h}{2}\right)^{1/m}.\end{aligned} \tag{5.1.1-6}$$

Therefore, for the rate of convergence we have

$$\mathcal{R}\left(\prod_{k=1}^{m} T_{r_k^{PR}}\right) = 4\left(\frac{\pi h}{2}\right)^{1/m}. \tag{5.1.1-7}$$

Thus, the rate of convergence is asymptotically proportional to $h^{1/m}$, and the number of iterations needed to reduce the error by a given tolerance is proportional to $h^{-1/m}$. For $m > 1$, this shows an order of magnitude improvement over the SOR method where its rate of convergence is proportional to h.

Young and Gregory [1973] has shown numerically that the parameters used by PR were nearly as good as the theoretical optimum parameters. Numerically, the bound on the average spectral radius

$$\left[\rho \left(\prod_{k=1}^{m} T_{r_k}^{PR} \right) \right]^{1/m} \tag{5.1.1-8}$$

can be shown to decrease at first, and then increases. For $h = 1/20$, it was found that the optimum number of parameters, $m = 3$ in the sense that the bound on (5.1.1-8) is minimised. Theoretically, this can be shown as follows. Let

$$z_m = \left(\frac{\mu}{\nu} \right)^{1/(2m)} \tag{5.1.1-9}$$

where μ and ν are as in (5.1.1-1). We now seek to determine m so that

$$\sigma_m = \left[\rho \left(\prod_{k=1}^{m} T_{r_k}^{PR} \right) \right]^{1/m} = \left(\frac{1 - z_m}{1 + z_m} \right)^{2/m}. \tag{5.1.1-10}$$

If we let

$$\delta = \frac{1 - z_m}{1 + z_m} \tag{5.1.1-11}$$

then

$$z_m = \frac{1 - \delta}{1 + \delta} \tag{5.1.1-12}$$

and, from (5.1.1-9)

$$m = \frac{log\ (a/b)}{2\ log\ [(1 - \delta)/(1 + \delta)]} \tag{5.1.1-13}$$

and from (5.1.1-10), we get

$$\begin{aligned} \zeta_m &= -\ log\ \sigma_m \\ &= -\frac{2}{m}\ log \left(\frac{1 - z_m}{1 + z_m} \right) \\ &= -\frac{4\ log\ [(1 - \delta)/(1 + \delta)]\ log\ \delta}{log\ (a/b)}\ . \end{aligned} \tag{5.1.1-14}$$

By equating to zero the first derivative of (5.1.1-14) with respect to δ, we get

$$\frac{1-\delta}{2}^{2} \log \frac{1-\delta}{1+\delta} = \delta \log \delta \qquad (5.1.1\text{-}15)$$

which has the solution

$$\bar{\delta} = \sqrt{2} - 1 \approx 0.414. \qquad (5.1.1\text{-}16)$$

Thus, the optimum value of m can be determined by finding the smallest integer such that

$$(0.414)^{2m} \leq \frac{a}{b} . \qquad (5.1.1\text{-}17)$$

For the model problem, $a/b = \tan^2(\pi h/2)$. Hence, for $h = 1/20$,

$$(0.414)^{2m} \leq \tan^2(\pi h/2) \approx 6.194 \times 10^{-3}$$

which gives $m = 3$.

The other parameter sequences are given as follows.

Wachspress Parameter Sequence

Wachspress [1968], has also solved the problem analytically in terms of elliptic functions and gives another set of parameters,

$$r_k^W = \nu \left(\frac{\mu}{\nu} \right)^{(k-1)/(m-1)}, \quad m \geq 2, \; k = 1, 2, \ldots, m. \qquad (5.1.1\text{-}18)$$

This set of parameters can be shown to have the similar rate of convergence as in (5.1.1-7). Evidently by (5.1.1-18), we have

$$\mu < r_{m-1}^W < \ldots < r_3^M < r_2^M < \nu \qquad (5.1.1\text{-}19)$$

with $r_1^W = \nu$ and $r_m^W = \mu$.

With this set of parameters we can show that

$$\varphi\left(\prod_{k=2}^{m-1} T_{r_k}^W \right) \leq \left(\frac{1 - \left(\frac{\mu}{\nu} \right)^{1/(m-1)}}{1 + \left(\frac{\mu}{\nu} \right)^{1/(m-1)}} \right)^2 . \qquad (5.1.1\text{-}20)$$

We now determine the rate of convergence for the Wachspress parameters. For μ and ν in (5.1.1-1), these parameters are given by

$$r_k^W = 4\cos^2\frac{\pi h}{2}\left(\tan^2\frac{\pi h}{2}\right)^{(k-1)/(m-1)}, \qquad (5.1.1\text{-}21)$$

$$k = 1,\ 2,\ \ldots,\ m,\ m \geq 2.$$

By substituting r_k^W in (5.1-12), we have

$$\mathcal{S}\left(\prod_{k=2}^{m-1} T_{r_k}^{PR}\right) \leq \left(\frac{1-\left(\tan^2\frac{\pi h}{2}\right)^{1/(m-1)}}{1+\left(\tan^2\frac{\pi h}{2}\right)^{1/(m-1)}}\right)^2$$

$$\sim \left(\frac{1-\left(\frac{\pi h}{2}\right)^{1/(m-1)}}{1+\left(\frac{\pi h}{2}\right)^{1/(m-1)}}\right)$$

$$\sim 1 - 4\left(\frac{\pi h}{2}\right)^{1/(m-1)} \qquad (5.1.1\text{-}22)$$

where $m \geq 2$. Therefore, for the rate of convergence we have

$$\mathcal{R}\left(\prod_{k=2}^{m-1} T_{r_k}^{W}\right) = 4\left(\frac{\pi h}{2}\right)^{1/(m-1)}. \qquad (5.1.1\text{-}23)$$

Hence, the rate of convergence is again asymptotically proportional to $O(h^{1/m})$.

Young and Ehrlich Parameter Sequence

Young and Ehrlich [1960], introduced another set of parameters given by

$$r_k^Y = \frac{1}{4}\left(\sin\frac{\pi h}{2}\right)^{(1-2k)/m}, \quad k = 1,\ 2,\ \ldots,\ m. \qquad (5.1.1\text{-}24)$$

Young has shown that these parameters also have a similar rate of convergence as in (5.1.1-7). Unfortunately, the application of the Young parameters can lead to a rounding error growth.

Wachspress and Jordan Parameter Sequence

Wachspress[1963], has obtained another set of m parameters when $m = 2^k$ but later generalised for all m. It was based on the hypothesis that these k optimum parameters which minimise the

function (5.1-12) lie on the interval governed by the aritho-geometric interval $[\sqrt{ab}, \left(\frac{a+b}{2}\right)]$. The two parameter formula given by Varga [1962], i.e.,

$$r_{1,2}^{J} = \left[\sqrt{ab}\left(\frac{a+b}{2}\right)\right]^{\frac{1}{2}} \pm \left[\sqrt{ab}\left(\frac{a+b}{2}\right) - ab\right]^{\frac{1}{2}} \qquad (5.1.1\text{-}25)$$

where $0 < a \leq b$, can be derived as follows.

It has been shown that for a single parameter, the optimum value is $r = \sqrt{ab}$ or $r_1 = ab/r_1$. Suppose now that we can algebraically determine the m parameters which minimise the function (5.1-12) for any interval. As shown by Wachspress [1963], this leads to a precise determination of the 2^k parameter problem. If the r_k are the parameters that minimise the function for the interval $0 < a \leq \gamma \leq b$, then

$$Q_m(\lambda, r_k) = \prod_{k=1}^{m} \left|\frac{(r_k - \gamma)}{(r_k + \gamma)}\right|$$

$$= \prod_{k=1}^{m} \left(\frac{r_k - \gamma}{r_k + \gamma}\right)\left(\frac{ab/r_k - \gamma}{ab/r_k + \gamma}\right). \qquad (5.1.1\text{-}26)$$

Thus, we can reduce the interval from $[a,b]$ to $[a, \sqrt{ab}]$. Hence,

$$\mathcal{S}\left(\prod_{k=1}^{m} T_{r_k}\right) \leq \max_{a \leq \gamma \leq \sqrt{ab}}\left(\prod_{k=1}^{m} \left(\frac{(r_k - \gamma)}{(r_k + \gamma)}\right)\left(\frac{(ab/r_k - \gamma)}{(ab/r_k + \gamma)}\right)\right)^2. \qquad (5.1.1\text{-}27)$$

Now we can divide each factor in the numerator and denominator of (5.1.1-26) by $\sqrt{ab}$. Then, Q_m can be expressed as a product of quadratics, i.e.,

$$Q_m(\lambda, r_k) = \prod_{k=1}^{m}\left[\frac{\left(\frac{\lambda}{\sqrt{ab}}\right)^2 + 1 - \left(\frac{\lambda}{\sqrt{ab}}\right)\left(\frac{r_k}{\sqrt{ab}} + \frac{\sqrt{ab}}{r_k}\right)}{\left(\frac{\lambda}{\sqrt{ab}}\right)^2 + 1 + \left(\frac{\lambda}{\sqrt{ab}}\right)\left(\frac{r_k}{\sqrt{ab}} + \frac{\sqrt{ab}}{r_k}\right)}\right]. \qquad (5.1.1\text{-}28)$$

By multiplying each quadratic by $ab/2\lambda$, we have

$$Q_m(\lambda, r_k) = \prod_{k=1}^{m} \left[\frac{\frac{\sqrt{ab}}{2}\left(\frac{\lambda}{\sqrt{ab}} + \frac{\sqrt{ab}}{\lambda}\right) - \frac{1}{2}\left(r_k + \frac{ab}{r_k} \right)}{\frac{\sqrt{ab}}{2}\left(\frac{\lambda}{\sqrt{ab}} + \frac{\sqrt{ab}}{\lambda}\right) + \frac{1}{2}\left(r_k + \frac{ab}{r_k} \right)} \right]. \qquad (5.1.1\text{-}29)$$

Then, by letting

$$\phi \equiv \frac{\sqrt{ab}}{2}\left(\frac{\lambda}{\sqrt{ab}} + \frac{\sqrt{ab}}{\lambda}\right), \quad s_k \equiv \frac{1}{2}\left(r_k + \frac{ab}{r_k} \right) \qquad (5.1.1\text{-}30)$$

we thus have

$$Q_m(\lambda, r_k) = \prod_{k=1}^{m} \left(\frac{s_k - \phi}{s_k + \phi} \right) \qquad (5.1.1\text{-}31)$$

where $0 < \sqrt{ab} \le \phi \le \frac{a+b}{2}$.

By using this hypothesis, one then may use (5.1.1-30) to determine the corresponding k parameters r_k for the interval $a \le \lambda \le b$ by solving the quadratic equations arising from (5.1.1-30). Now, let us define

$$a_0 = a, \qquad b_0 = b,$$
$$a_{k+1} = \sqrt{a_k b_k}, \qquad b_{k+1} = (a_k + b_k)/2, \qquad k \ge 0. \qquad (5.1.1\text{-}32)$$

Thus, the problem of finding the 2^k parameters for the interval $[a,b]$ is reduced to the problem of finding a single optimum parameter for the interval $[a_k, b_k]$, whose solution is obviously $\sqrt{a_k b_k}$. Hence, for two parameters with $\sqrt{ab} \le \phi \le \frac{a+b}{2}$, the problem is reduced to solving

$$\frac{1}{2}\left(r_k + \frac{ab}{r_k} \right) = a_2 \qquad (5.1.1\text{-}33)$$

where

$$a_2 = \sqrt{a_1 b_1} = [\sqrt{ab}(\tfrac{a+b}{2})]^{1/2}. \qquad (5.1.1\text{-}34)$$

The relation (5.1.1-33) can be solved quadratically, giving

$$r_{1,2}^J = \left[\sqrt{ab}\left(\frac{a+b}{2}\right)\right]^{\frac{1}{2}} \pm \left[\sqrt{ab}\left(\frac{a+b}{2}\right) - ab\right]^{\frac{1}{2}}.$$

Douglas Parameter Sequence

Douglas [1962], has also given a set of PR parameter formulation for the multi-parameters. It is called the *finite geometric parameter*

sequence and is given in the form

$$r_{k+1}^{D} = \frac{1}{2}\left(\frac{\mu}{\nu}\right)^{k} \frac{\mu}{\sin^2 \frac{\pi}{2N}} \tag{5.1.1-35}$$

or

$$r_{k+1}^{D} = \frac{1}{2}\left(\frac{\nu}{\mu}\right)^{k} \frac{\mu}{\cos^2 \frac{\pi}{2N}} \tag{5.1.1-36}$$

where $k = 0, 1, 2, \ldots, (k_0 - 1)$, with k_0 given approximately by

$$k_0 \approx \frac{2 \log \tan (\pi/2N)}{\log (\mu/\nu)} . \tag{5.1.1-37}$$

The formulation (5.1.1-35) can be derived as follows:

Let $\zeta_{k+1} = 2r_{k+1}$, $k = 0, 1, 2, \ldots$ (5.1.1-38)

and $\xi^{(s)} = \sin^2 \frac{s\pi}{2N}$, $s = 1, 2, \ldots, N-1$,

and put $\zeta_1 \xi^{(1)} = \mu$, $\zeta_1 \xi^{(2)} = \nu$, with $\xi^{(1)} = \sin^2 \frac{\pi}{2N}$. (5.1.1-39)

Then, from (5.1.1-38), it follows that

$$\zeta_1 = \frac{\mu}{\sin^2 \frac{\pi}{2N}}, \qquad \xi^{(2)} = \frac{\nu}{\mu} \sin^2 \frac{\pi}{2N} . \tag{5.1.1-40}$$

Similarly, if $\zeta_1 \xi^{(1)} = \mu$, $\zeta_1 \xi^{(2)} = \nu$, then

$$\zeta_2 = \frac{\mu}{\nu} \frac{\mu}{\sin^2 \frac{\pi}{2N}}, \quad \xi^{(3)} = \left(\frac{\nu}{\mu}\right)^2 \sin^2 \frac{\pi}{2N} . \tag{5.1.1-41}$$

By continuing this process, we have in general

$$\zeta_{k+1} = \left(\frac{\mu}{\nu}\right)^{k} \frac{\mu}{\sin^2 \frac{\pi}{2N}}, \quad k = 0, 1, 2, \ldots, (k_0 - 1) \tag{5.1.1-42}$$

and

$$\xi^{(k+1)} = \left(\frac{\nu}{\mu}\right)^{k} \sin^2 \frac{\pi}{2N}, \quad k = 0, 1, 2, \ldots, k_0 \tag{5.1.1-43}$$

are obtained where k_0 is the largest number such that

$$\left(\frac{\nu}{\mu}\right)^{k_0} \sin^2 \frac{\pi}{2N} \leq \cos^2 \frac{\pi}{2N} . \tag{5.1.1-44}$$

Thus, the finite geometric sequence is given by

$$r_{k+1}^{D} = \frac{1}{2}\left(\frac{\mu}{\nu}\right)^{k} \frac{\mu}{\sin^2 \frac{\pi}{2N}}, \quad k = 0, 1, 2, \ldots, (k_0 - 1).$$

Similarly, the formulation for (5.1.1-36) is as follows:

By putting $\zeta_1 \xi^{(1)} = \nu$, $\zeta_1 \xi^{(2)} = \mu$, with $\xi^{(1)} = \cos^2 \frac{\pi}{2N}$, the formulation can be derived in a similar way. These results are obtained when q is the largest integer such that

$$\left(\frac{\nu}{\mu} \right)^q \cos^2 \frac{\pi}{2N} \leq \sin^2 \frac{\pi}{2N} . \qquad (5.1.1\text{-}45)$$

From (5.1.1-44) and (5.1.1-45), it can be deduced that

$$q \approx \frac{2 \log \tan (\pi/2N)}{\log (\mu/\nu)} .$$

Douglas also gives another version using the Wachspress parameter sequence and it is given by

$$r_{k+1} = \frac{1}{2 \cos^2 \frac{\pi}{2N}} \left(\cot^2 \frac{\pi}{2N} \right)^{k/(q-1)} , \qquad (5.1.1\text{-}46)$$

$$k = 0, 1, 2, \ldots, q - 1$$

where $q \geq 2$ is the smallest integer given by

$$(\sqrt{2} - 1)^{(q-1)} \leq \tan \frac{\pi}{2N} . \qquad (5.1.1\text{-}47)$$

The details of the derivation of this parameter sequence, which is close to the optimum value for the model problem (elliptic 2D problem), are omitted. The derivation requires a commutativity property on H and V.

In the next section, we present an alternative approach, i.e., *The Heuristic Search*, to find a sequence of parameters for attaining an improved rate of convergence.

5.1.2 The heuristic search for multi-parameters

All the existing multi-parameters discussed in Section 5.1.1, are based on solving the two dimensional elliptic problem in a square region bounded by the Dirichlet boundary conditions using the ADI method in PR form.

Although the technique of splitting A into two submatrices via the AGE method is similar to the splitting in the ADI method, the resultant matrices G_1 and G_2 consists of only (2X2) block submatrices. Hence, it is a natural extension to apply the ADI parameters strategies to the AGE method. However, we must also bear in mind that in ADI, H and V are both tridiagonal matrices. Moreover, this shows that the eigenvalues of G_1 and G_2 are clustered whereas the eigenvalues of H and V are distinct and distributed. Hence, it is envisaged that it may only be necessary to consider a cycle of only 1, 2 or 3 parameters from the existing parameter sequence of Section 5.1.1 for the AGE method in order to show whether an improved rate of convergence is possible.

The alternative approach to the multi-parameter case is to perform a heuristic search for the parameters. The simplest heuristic search is a two parameter search. The greater the number of parameter to be found, the more laborious the search. In this section, we will only consider the heuristic search for parameters in a cycle of 2 and 3.

In a cycle of 2, one chooses two arbitrary values, say, r_1 and r_2 with r_1 to be near the optimal single parameter. The parameter r_1 is fixed whilst the other parameter r_2 is varied until the minimal number of iterations is reached. At this stage, the parameter r_2 can be taken as a local optimal parameter. Now, with the parameter r_2 fixed, we search for the new value of parameter r_1 that gives a minimal number of iterations. At this stage, the parameter r_1 can be considered as a local optimal parameter. This process must be repeated at least once. This is to ensure that r_1 and r_2 are the optimal parameters that yield the global minimal number of iterations.

In a cycle of 3, the search becomes more tedious. Now, one has to choose three arbitrary parameters, say, r_1, r_2 and r_3 with one of them, let r_1 to be near to the optimal single parameter. Now, with r_1 and r_2 fixed, the parameter r_3 is varied until the local minimal number of iterations is found. Then, at this stage, one can take r_3 as a local optimal parameter.

Now, one varies r_2 whilst the parameters r_1 and r_3 are fixed until one gets the local minimal number of iterations. At this stage, r_2 can be considered as a local optimal parameter. Finally, one searches for r_1 while having fixed the parameter r_2 and r_3. Then, r_1 can be taken as a local optimal parameter when the local minimal number of iterations is reached. As in a cycle of 2, this process must also be repeated as to confirm that these three parameters are optimal and giving the global minimal number of iterations.

5.1.3 The solution for two and three parameters

Consider a linear system derived from a two-point boundary-value problem

$$A\mathbf{u} = \mathbf{b} \tag{5.1.3-1}$$

where $\mathbf{u}$ and $\mathbf{b}$ are N-dimensional vectors and A is given as

$$A = \begin{bmatrix} 2g_1 & c_1 & & & \\ a_2 & 2g_2 & c_2 & & 0 \\ & \ddots & \ddots & \ddots & \\ & & a_{N-1} & 2g_{N-1} & c_{N-1} \\ 0 & & & a_N & 2g_N \end{bmatrix}. \tag{5.1.3-2}$$

The AGE iterative method consists of splitting the matrix A into the form

$$A = G_1 + G_2 \tag{5.1.3-3}$$

where

$$G_1 = \begin{bmatrix} g_1 & c_1 & & & & \\ a_2 & g_2 & & & & \\ & & g_3 & c_3 & & \\ & & a_4 & g_4 & & \\ & & & & \ddots & \\ & & & & g_{N-1} & c_{N-1} \\ & & & & a_N & g_N \end{bmatrix} \tag{5.1.3-4}$$

and

$$G_2 = \begin{bmatrix} g_1 & & & & & \\ & g_2 & c_2 & & & \\ & a_3 & g_3 & & & \\ & & & \ddots & & \\ & & & g_{N-2} & c_{N-2} & \\ & & & a_{N-1} & g_{N-1} & \\ & & & & & g_N \end{bmatrix} \tag{5.1.3-5}$$

for N is even. Let us assume that all the eigenvalues of G_1 and G_2 are real and positive, i.e., $g_i > \frac{1}{2}(a_i + c_i)$, $i = 1, 2, \ldots, N$.

It is obvious that G_1 and G_2 consist of (2X2) block submatrices and the eigenvalues of G_1 and G_2 are given by

$$det \begin{vmatrix} \lambda - g_i & c_i \\ a_{i+1} & \lambda - g_{i+1} \end{vmatrix} = 0 \tag{5.1.3-6}$$

i.e., $\lambda^2 - (g_i + g_{i+1})\lambda + g_i g_{i+1} - a_{i+1} c_i = 0$,

or $$\lambda_{i,i+1} = \frac{1}{2}(g_i + g_{i+1}) \pm \frac{1}{2}\sqrt{(g_i - g_{i+1})^2 + 4a_{i+1}c_i} \ . \qquad (5.1.3\text{-}7)$$

The matrix G_2 has two more eigenvalues and these are given by

$$\lambda_1 = g_1 \text{ and } \lambda_N = g_N. \qquad (5.1.3\text{-}8)$$

The bounds for these eigenvalues can be obtained by inspection for the different cases below. Let $a = max\ a_i$ and $c = max\ c_i$.

Case I. All g_i are equal, i.e., $g = g_i$.

Thus, the bounds are $\mu = g - \sqrt{ac}$ and $\nu = g + \sqrt{ac}$. (5.1.3-9)

Case II. All g_i are monotonically increasing, i.e., $g_i < g_{i+1}$, $\forall\ i$.

Thus, $$\mu = \frac{1}{2}(g_1 + g_2) - \frac{1}{2}\sqrt{(g_1 - g_2)^2 + 4ac}, \text{ and} \qquad (5.1.3\text{-}10)$$

$$\nu = \frac{1}{2}(g_{N-1} + g_N) + \frac{1}{2}\sqrt{(g_{N-1} - g_N)^2 + 4ac} \ . \qquad (5.1.3\text{-}11)$$

Case III. All g_i are monotonically decreasing, i.e., $g_i > g_{i+1}$, $\forall\ i$.

Thus, $$\mu = \frac{1}{2}(g_{N-1} + g_N) - \frac{1}{2}\sqrt{(g_{N-1} - g_N)^2 + 4ac}, \text{ and} \qquad (5.1.3\text{-}12)$$

$$\nu = \frac{1}{2}(g_1 + g_2) - \frac{1}{2}\sqrt{(g_1 - g_2)^2 + 4ac} \ . \qquad (5.1.3\text{-}13)$$

With this bounds, we may investigate the possibility of reducing the number of iterations for the case of 2 and 3 parameters, from the relation (5.1.1-17), i.e., $(0.414)^{2m} \leq \frac{\mu}{\nu}$. Since μ and ν differ from problem to problem, the analysis will be presented experimentally in greater detail. First, we present the sequence of 2 and 3 parameters from the existing parameter discussed in Section 5.1.1.

(1) *The PR Parameter sequence.*

(a) 2 parameters.

$$r_1^{PR} = \nu\left(\frac{\mu}{\nu}\right)^{1/4}, \ r_2^{PR} = \nu\left(\frac{\mu}{\nu}\right)^{3/4}. \qquad (5.1.3\text{-}14)$$

(b) 3 parameters.

$$r_1^{PR} = \nu\left(\frac{\mu}{\nu}\right)^{1/6},\quad r_2^{PR} = \nu\left(\frac{\mu}{\nu}\right)^{1/2},\quad r_3^{PR} = \nu\left(\frac{\mu}{\nu}\right)^{5/6}. \qquad (5.1.3\text{-}15)$$

(2) *The Wachspress Parameter sequence.*

(a) 2 parameters.

$$r_1^{W} = \nu,\quad r_2^{W} = \mu. \qquad (5.1.3\text{-}16)$$

(b) 3 parameters.

$$r_1^{W} = \nu,\quad r_2^{W} = \nu\left(\frac{\mu}{\nu}\right)^{1/2},\quad r_3^{W} = \mu. \qquad (5.1.3\text{-}17)$$

(3) *The Young Parameter sequence.*

(a) 2 parameters.

$$r_1^{Y} = \frac{1}{4}\left(sin\ \frac{\pi h}{2}\right)^{-1/2},\quad r_2^{Y} = \frac{1}{4}\left(sin\ \frac{\pi h}{2}\right)^{-3/2}. \qquad (5.1.3\text{-}18)$$

(b) 3 parameters.

$$r_1^{Y} = \frac{1}{4}\left(sin\ \frac{\pi h}{2}\right)^{-1/6},\quad r_2^{Y} = \frac{1}{4}\left(sin\ \frac{\pi h}{2}\right)^{-1/2},\quad r_3^{Y} = \frac{1}{4}\left(sin\ \frac{\pi h}{2}\right)^{-5/6}. \qquad (5.1.3\text{-}19)$$

(4) *The Jordan Parameters*

(a) 2 parameters.

$$r_{1,2}^{J} = \left[\sqrt{ab}\left(\frac{a+b}{2}\right)\right]^{\frac{1}{2}} \pm \left[\sqrt{ab}\left(\frac{a+b}{2}\right) - ab\right]^{\frac{1}{2}}. \qquad (5.1.3\text{-}20)$$

(5) *The Douglas Parameters - PR version.*

(a) 2 parameters.

$$r_1^{DPR} = \frac{\nu}{2\ sin^2\ \frac{\pi h}{2}},\qquad r_2^{DPR} = \frac{\mu}{2\ sin^2\ \frac{\pi h}{2}} \qquad (5.1.3\text{-}21)$$

or

$$r_1^{DPR} = \frac{\mu}{2\ cos^2\ \frac{\pi h}{2}},\qquad r_2^{DPR} = \frac{\nu}{2\ cos^2\ \frac{\pi h}{2}}. \qquad (5.1.3\text{-}22)$$

(b) 3 parameters.

r_1 and r_2 are as in (5.1.3-21) or (5.1.3-22) respectively,

and

$$r_3^{DPR} = \frac{\mu^2}{2\nu \sin^2 \frac{\pi h}{2}}, \quad \text{or} \quad r_3^{DPR} = \frac{\nu^2}{2\mu \cos^2 \frac{\pi h}{2}} . \qquad (5.1.3\text{-}23)$$

(6) *The Douglas Parameters - Wachspress version.*

(a) 2 parameters.

$$r_1^{DW} = \frac{1}{2 \cos^2 \frac{\pi h}{2}}, \qquad r_2^{DW} = \frac{1}{2 \cos^2 \frac{\pi h}{2}} \left(\cot^2 \frac{\pi h}{2} \right)^{1/2} . \qquad (5.1.3\text{-}24)$$

(b) 3 parameters.

r_1^{DW} and r_2^{DW} are as in (5.1.3-24), and $r_3^{DW} = \dfrac{1}{2 \sin^2 \frac{\pi h}{2}}$.

(5.1.3-25)

5.1.4 Other methods to determine the m (>1) parameters

Since no established theory is known to date, apart from using the existing ADI multi-parameter formula and by heuristic search, several methods to determine the m parameters have also been investigated. Firstly, based on the arithmetic mean principle. Here, we consider the parameters, r_1, r_2, ... r_m to be evenly distributed as in *Figure* 5.1.4-1

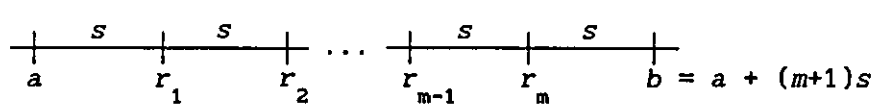


Figure 5.1.4-1: The arithmetic mean principle

where a and b are the bounds of the eigenvalues of G_1 and G_2. This will give

$$r_k = a + \frac{(b-a)k}{m+1}, \quad k = 1, 2, \ldots, m. \qquad (5.1.4\text{-}1)$$

Secondly, the geometric mean principle is used, i.e, by considering the parameters to be distributed as $r_1 = at$, $r_2 = at^2$, ..., $r_m = at^m$ and finally, $b = at^{m+1}$ as in *Figure* 5.1.4-2

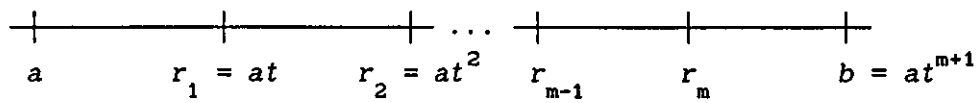


Figure 5.1.4-2: The geometric mean principle

where t is defined as the geometric ratio, i.e., $t = (b/a)^{1/(m+1)}$. This will give

$$r_k = a\left(\frac{b}{a}\right)^{k/(m+1)}, \quad k = 1, 2, \ldots, m. \tag{5.1.4-2}$$

Another hypothesis based on the inverse law is also considered. This is due to the nature of the submatrices G_1 and G_2 which consist of the (2X2) block of the matrix $\hat{G}$,

$$\hat{G} = \begin{bmatrix} g & -1 \\ -1 & g \end{bmatrix} \tag{5.1.4-3}$$

which has the eigenvalues g-1 and g+1. The third eigenvalue, i.e., for the odd case, (N is odd) is g.

Since the smallest eigenvalue is zero, then the parameter r_1 is also close to zero. Thus, for the method to converge, we must choose the parameter r_2 large enough in the relation $r_2 = \alpha/r_1$ where α is a certain number in the eigenvalue range. When $r_1 = r_2$ (in a single parameter case), we have the relation $r = \sqrt{\alpha}$ which guarantees that the solution converges.

However, this principal can lead to large parameter values and hence round off error growth as will be discussed later.

5.1.5 Experimental results

The application of the existing multi-parameters theoretical analysis (Section 5.1.1), the heuristic search for multi-parameters (Section 5.1.2) and other methods in Section 5.1.4 were all investigated on three problems, the linear and non-linear problems bounded by Dirichlet boundary conditions and a linear problem which has periodic boundary conditions. The results obtained were compared for the rate of convergence.

Problem 1 - A Linear Problem

$$U'' - (1 + x^2)U = -1, \qquad -1 \le x \le 1,$$

$$U(-1) = 0,\ U(1) = 0, \qquad h = 2/(N+1).$$

The computed solution is obtained when $\| \mathbf{r} \| = \| \mathbf{b} - A\mathbf{u}^{(k+1)} \| < \varepsilon$.

The matrix A for this problem is given by

$$A = \begin{bmatrix} 2g_1 & -1 & & & \\ -1 & 2g_2 & -1 & & 0 \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2g_{N-1} & -1 \\ & 0 & & -1 & 2g_N \end{bmatrix}$$

where $g_i = 1 + 0.5h^2(1 + x_i^2)$, $1 \le i \le N$.

The value g_i is monotonically increasing for i increasing, then by splitting the matrix A into G_1 and G_2, the bounds of the eigenvalues may be determined from the relation (5.1.3-10). These values are

$$\mu = \frac{1}{2}(g_1 + g_2) - \frac{1}{2}\sqrt{(g_1 - g_2)^2 + 4}$$

and

$$\nu = \frac{1}{2}(g_{N-1} + g_N) + \frac{1}{2}\sqrt{(g_{N-1} - g_N)^2 + 4}\ .$$

By inspection, it can be shown that the relation (5.1.1-10) is minimised for a specified value of m used in the relation (5.1.1-17). However, the experimental results obtained on using this sequence of parameters do not show any improvement concerning the rate of convergence. For this problem, although the eigenvalues of the iteration matrix are well separated, they are of high order in multiplicity. Thus, there is a possibility of rounding errors in the minimisation process which reduces the rate of convergence.

The heuristic 2 parameter search yields $r_1 = r_2$, which signifies that the optimal single parameter is the best. These results are shown in *Table* 5.1.5-1 below.

	AGE-DG Scheme		Heuristic Search		
N	Optimum r	Iter	r_1	r_2	Iter
10	0.560 - 0.580	12	0.560	0.560	12
20	0.303 - 0.305	20	0.303	0.303	20
40	0.160 - 0.161	39	0.160	0.160	39
80	0.080	70	0.080	0.080	70

Table 5.1.5-1: The number of iterations

Problem 2 - A Mildly Nonlinear Problem.

$$-U'' + \tfrac{3}{2}U^2 = 0, \qquad 0 \le x \le 1,$$

$$U(0) = 4, \; U(1) = 1, \qquad h = 1/(N+1).$$

The exact solution to this problem is given by $U(x) = 4/(1+x)^2$.

The matrix A is given by

$$A = \begin{bmatrix} 2g_1 & -1 & & & \\ -1 & 2g_2 & -1 & & 0 \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2g_{N-1} & -1 \\ & 0 & & -1 & 2g_N \end{bmatrix}$$

where $g_i = 1 + 0.75h^2u_i$, $1 \le i \le N$.

The value g_1 is now monotonically decreasing, thus we may use the relation (5.1.3-11) to determine the bounds of the eigenvalues. Having split the matrix A into G_1 and G_2, we then have

$$\mu = \frac{1}{2}(g_{N-1} + g_N) + \frac{1}{2}\sqrt{(g_{N-1} - g_N)^2 + 4}$$

and $$\nu = \frac{1}{2}(g_1 + g_2) - \frac{1}{2}\sqrt{(g_1 - g_2)^2 + 4}\ .$$

Experimentally, we found there is no gain in the number of iterations when solving the problem either by using the existing parameters or via the heuristic search. This is again due to the same reasons as given previously. Thus, the optimal single parameter is definitely the best choice. For the results, see *Table* 5.1.5-2.

	AGE-DG Scheme		Heuristic Search		
N	Optimum r	Iter	r_1	r_2	Iter
10	0.572 - 0.579	17	0.576	0.576	17
20	0.317 - 0.323	35	0.320	0.320	35
40	0.164 - 0.179	72	0.170	0.170	72
80	0.089 - 0.093	143	0.090	0.090	143

Table 5.1.5-2: The number of iterations

Problem 3 - A Linear Periodic Problem

$$-U'' + \rho U = (\rho+1)(\sin x + \cos x), \qquad 0 \le x \le 2\pi,$$

$$U(0) = U(2\pi),\ U'(0) = U'(2\pi), \qquad h = 2\pi/N.$$

The exact solution is $U(x) = \sin x + \cos x$. The matrix A is

$$A = \begin{bmatrix} 2g & -1 & & & -1 \\ -1 & 2g & -1 & 0 & \\ & \ddots & \ddots & \ddots & \\ 0 & -1 & 2g & & -1 \\ -1 & & -1 & & 2g \end{bmatrix}$$

where $g = 1 + 0.5\rho h^2$.

Since g is constant, then by splitting the matrix A into G_1 and G_2, we can use the relation (5.1.3-9) to the determine bounds of the eigenvalues and these are given by

$$\mu = g - 1 = 0.5\rho h^2 \text{and } \nu = g + 1 = 2 + 0.5\rho h^2.$$

The results upon using the PR, Wachspress and Jordan parameters, i.e., for 2 and 3 parameters derived in Section 5.1.3, are given in *Tables* 5.1.5-3 and 5.1.5-4. NC in *Table* 5.1.5-3 means no convergence.

ρ=70	AGE-DG Scheme		2 parameters		
N	Optimum *r*	Iter	PR	Wachspress	Jordan
20	3.652 - 4.908	4	4	4	4
40	1.283 - 1.807	6	5	5	5
80	0.686 - 0.708	9	7	6	7
160	0.271 - 0.342	16	11	6	9
320	0.138 - 0.154	26	NC	6	NC

Table 5.1.5-3: The number of iterations

ρ=70	AGE-DG Scheme		3 parameters	
N	Optimum *r*	Iter	PR	Wachspress
20	3.652 - 4.908	4	4	4
40	1.283 - 1.807	6	5	6
80	0.686 - 0.708	9	8	8
160	0.271 - 0.342	16	10	8
320	0.138 - 0.154	26	13	8

Table 5.1.5-4: The number of iterations

The results show a clear gain in the number of iterations when using the multi-parameter formula for solving the linear problem subject to periodic boundary conditions. It is clear that the Wachspress parameters are the best over all other parameters. Although the results given are only for the value ρ = 70, it can also be shown that for larger ρ, the gain in the number of iterations is also significant. The PR and Jordan parameters do not give convergence for these cases since one of the parameters is not large enough.

5.2 The multi-parameter case for semi-iterative method

In this section, we intend to make an improvement on the rate of convergence of the AGE method by applying the multi-parameter case for the semi-iterative method. Two methods will be discussed, i.e, the 1st order Richardson iterative method and the 2nd order Chebyshev semi-iterative method. Initially, we derive the general concept of the semi-iterative method.

For any stationary iterative process to solve $A\mathbf{u} = \mathbf{b}$, we can write

$$N\mathbf{u}^{(k+1)} = N\mathbf{u}^{(k)} - (A\mathbf{u}^{(k)} - \mathbf{b}), \quad N \neq 0 \tag{5.2-1}$$

or
$$\mathbf{u}^{(k+1)} = (I - N^{-1}A)\mathbf{u}^{(k)} - N^{-1}\mathbf{b} \tag{5.2-2}$$

$$= \mathcal{G}\mathbf{u}^{(k)} + \mathcal{C} \tag{5.2-3}$$

where N is easily inverted. $\mathcal{G}$ is the iteration matrix with $\rho(\mathcal{G}) < 1$. Thus, all the eigenvalues of $\mathcal{G}$, λ_m satisfy

$$-\rho(\mathcal{G}) < \lambda_m < \rho(\mathcal{G}) \tag{5.2-4}$$

where $|\mathcal{G} - \lambda_m I| = 0$ and A is Symmetric.

Now, consider the splitting of $A = D - L - U$, then we can identify the process as

Jacobi, if $N = D$, (5.2-5)

Gauss-Seidel, if $N = D - L$, (5.2-6)

and SOR, if $N = \frac{1}{\omega}(D - \omega L)$, $\quad 0 < \omega < 2$. (5.2-7)

We now, find the values of $\mathcal{G}$ and N for the AGE method.

By Combining (5.1-3) and (5.1-4) in order to eliminate $\mathbf{u}^{(k+1/2)}$, we then have

$$\mathbf{u}^{(k+1)} = (rI + G_2)^{-1}(rI - G_1)(rI + G_1)^{-1}(rI - G_2)\mathbf{u}^{(k)} + (rI + G_2)^{-1}\mathbf{b} + (rI + G_2)^{-1}(rI - G_1)(rI + G_1)^{-1}\mathbf{b}. \tag{5.2-8}$$

This form is similar to the previously discussed CAGE-PR(1) scheme.

It has been shown that the CAGE-PR(1) scheme has a similar rate of convergence compared to the CAGE-DG scheme. Thus, instead of using equation (5.2-8), we now apply the CAGE-DG scheme to find $\mathcal{G}$ and N. Now, if we recall the CAGE-DG scheme, in explicit form, we have

$$\mathbf{u}^{(k+1)} = [I - 2r(rI + G_2)^{-1}(rI + G_1)^{-1}A]\mathbf{u}^{(k)} + 2r(rI + G_2)^{-1}(rI + G_1)^{-1}\mathbf{b} \qquad (5.2\text{-}9)$$

which gives

$$\mathcal{G} = I - 2r(rI + G_2)^{-1}(rI + G_1)^{-1}A. \qquad (5.2\text{-}10)$$

Thus, $$N^{-1} = 2r(rI + G_2)^{-1}(rI + G_1)^{-1}. \qquad (5.2\text{-}11)$$

It can be shown further that after some manipulations,

$$\begin{aligned}\mathcal{G} &= I - 2r(rI + G_2)^{-1}(rI + G_1)^{-1}A \\ &= (rI + G_2)^{-1}(rI - G_1)(rI + G_1)^{-1}(rI - G_2)\end{aligned}$$

and

$$\begin{aligned}\mathcal{C} &= 2r(rI + G_2)^{-1}(rI + G_1)^{-1}\mathbf{b} \\ &= (rI + G_2)^{-1}\mathbf{b} + (rI + G_2)^{-1}(rI - G_1)(rI + G_1)^{-1}\mathbf{b}.\end{aligned}$$

5.2.1 The Richardson method

In an early paper, Richardson [1910] presented a method for solving the linear system $A\mathbf{u} = \mathbf{b}$. The method, known as the Richardson method, is defined as

$$N\mathbf{u}^{(k+1)} = N\mathbf{u}^{(k)} - \omega_{k+1}(A\mathbf{u}^{(k)} - \mathbf{b}), \quad N \neq 0 \qquad (5.2.1\text{-}1)$$

or after some manipulation, as

$$\mathbf{u}^{(k+1)} = (I - \omega_{k+1}N^{-1}A)\mathbf{u}^{(k)} + \omega_{k+1}N^{-1}\mathbf{b}, \qquad (5.2.1\text{-}2)$$

where $\omega_1, \omega_2, \ldots, \omega_m$ are the iteration parameters.

It is obvious that the method (5.2.1-2) defines a nonstationary linear iterative method. Moreover, the method is consistent for any $\omega_k \neq 0$.

Evidently if A is nonsingular, then the method is convergent if and only if

$$\lim_{k\to\infty} \prod_{k=1}^{m} (I - \omega_{k+1} N^{-1} A) = 0. \qquad (5.2.1\text{-}3)$$

It is obvious that if we choose $\omega_{k+1} = 1$, we would get the form of (5.1-1). Now, for $\omega_{k+1} = \omega$, we have the Jacobi overrelaxation method. In fact, if $N^{-1}A$ has the property that all the eigenvalues of $N^{-1}A$ are positive and real, and the maximum and minimum eigenvalues for $N^{-1}A$ are $M(N^{-1}A)$ and $m(N^{-1}A)$ or M_1 and m_1 respectively, and for $I - N^{-1}A$ are $M(I - N^{-1}A)$ and $m(I - N^{-1}A)$ or M_2 and m_2 respectively, then

$$m_1 = 1 - M_2, \qquad m_2 = 1 - M_1. \qquad (5.2.1\text{-}4)$$

For $I - \omega N^{-1}A$, we have $M(I - \omega N^{-1}A)$ and $m(I - \omega N^{-1}A)$ or M_3, m_3 respectively, where it can be verified that

$$m_3 = - M_3 = \frac{m_2 - M_2}{2 - m_2 - M_2}. \qquad (5.2.1\text{-}5)$$

For the values of $\omega_1, \omega_2, \ldots, \omega_m$, Young[1954] has shown that the optimum choice of these parameters is given by

$$\omega_k = \frac{2}{(M_1 + m_1) - (M_1 - m_1)t_k}, \quad k = 1, 2, \ldots, m \qquad (5.2.1\text{-}6)$$

where

$$t_k = \cos\left[\frac{(2k - 1)\pi}{2m}\right], \quad k = 1, 2, \ldots, m \qquad (5.2.1\text{-}7)$$

are the zeroes of the kth Chebyshev polynomials.

For m parameters, the method is expected to converge faster than a single parameter case, but round-off errors may cause difficulties in the actual calculation. If the original iteration does not have this property (the SOR method has complex eigenvalues for $\omega > 1$), then no acceleration of the basic iteration may result.

Now, by applying the Richardson method of the form (5.2.1-2) to the AGE method in (5.2-9), we then have

$$\mathbf{u}^{(k+1)} = [I - 2r\omega_{k+1}(rI + G_2)^{-1}(rI + G_1)^{-1}A]\mathbf{u}^{(k)} + 2r\omega_{k+1}(rI + G_2)^{-1}(rI + G_1)^{-1}\mathbf{b} \qquad (5.2.1\text{-}8)$$

where ω_{k+1} is defined in (5.2-1-6). As before, we will consider a cycle of 2 or 3 parameters.

We shall remark here, although a cycle of 2 or 3 parameters has been chosen, however the order in which the ω_{k+1} are used within a cycle of 2 or 3 iterations is arbitrary. This ordering does not effect the theoretical rate of convergence, but it may be important because of the growth of round-off errors.

In the non-stationary case, we could simply do it by substituting $r = r_k$, giving the equation

$$\mathbf{u}^{(k+1)} = [I - 2r_k\omega_{k+1}(r_kI + G_2)^{-1}(r_kI + G_1)^{-1}A]\mathbf{u}^{(k)} + 2r_k\omega_{k+1}(r_kI + G_2)^{-1}(r_kI + G_1)^{-1}\mathbf{b}. \qquad (5.2.1\text{-}9)$$

However, we will only investigate the Richardson method when r is constant. i.e., equation (5.2.1-8). The algorithm for the method is similar to Algorithm 4.2.2-2 except the multiplication of $2r$, we change to $2r\omega_{k+1}$.

5.2.2 The Chebyshev semi-iterative method

In this section, we shall be concerned with the application of the 2nd degree Chebyshev semi-iterative procedure to speed up the AGE method. This procedure represents a slightly different process where

$$\mathbf{u}^{(1)} = \omega_0[(I - N^{-1}A)\mathbf{u}^{(0)} + N^{-1}\mathbf{b}], \qquad (5.2.2\text{-}1)$$

and $\mathbf{u}^{(k+1)} = \omega_k[(I - N^{-1}A)\mathbf{u}^{(k)} + N^{-1}\mathbf{b}] + (1 - \omega_k)\mathbf{u}^{(k-1)}$

$$k = 1, 2, \ldots \qquad (5.2.2\text{-}2)$$

where the sequence $\omega_1, \omega_2, \ldots$ is determined by,

$$\omega_0 = 1, \ \omega_1 = \left(1 - \frac{1}{2}\lambda_{max}^2 \right)^{-1}$$

$$\omega_{k+1} = \left(1 - \frac{1}{4}\lambda_{max}^2 \omega_k \right)^{-1}, \qquad k = 1, 2, \ldots \qquad (5.2.2\text{-}3)$$

with λ_{max} is the maximum eigenvalue of the matrix $(I - N^{-1}A)$.

It is obvious that ω_i monotonically tends to the limit ω_b, the SOR acceleration parameter. These parameters are also based on the use of Chebyshev polynomials, see Young [1971]. It should be noticed that as it is a second degree process, $\mathbf{u}^{(k+1)}$ is calculated not only from $\mathbf{u}^{(k)}$ but also from $\mathbf{u}^{(k-1)}$. A fuller account of the derivation of the method may be found in Varga [1962].

The spectral radius is given by,

$$(\omega - 1)^{k/2}\left\{ \frac{2}{1 + (\omega - 1)^k} \right\}, \qquad (5.2.2\text{-}4)$$

where

$$\omega = \frac{2}{1 + \sqrt{1 - \lambda_{max}^2}} \qquad (5.2.2\text{-}5)$$

Now, for the AGE method in (5.2-9), the Chebyshev semi-iterative procedure (5.2.2-1) - (5.2.2-2) takes the form,

$$\mathbf{u}^{(1)} = \mathcal{G}\mathbf{u}^{(0)} + \mathcal{C}, \qquad (5.2.2\text{-}6)$$

and $\mathbf{u}^{(k+1)} = \omega_k(\mathcal{G}\mathbf{u}^{(k)} + \mathcal{C}) + (1 - \omega_k)\mathbf{u}^{(k-1)}, \ k = 1, 2, \ldots \qquad (5.2.2\text{-}7)$

where

$$\mathcal{G} = I - 2r(rI + G_2)^{-1}(rI + G_1)^{-1}A, \quad \mathcal{C} = 2r(rI + G_2)^{-1}(rI + G_1)^{-1}\mathbf{b}.$$

Again, we will consider a cycle of 2 or 3 parameters. Thus, we have, for a cycle of 2 parameters,

$$\omega_1 = \left(1 - \frac{1}{2}\lambda_{max}^2 \right)^{-1} \text{ and } \omega_2 = \left(1 - \frac{1}{4}\lambda_{max}^2 \omega_1 \right)^{-1}, \qquad (5.2.2\text{-}8)$$

and for cycle of 3 parameters

$$\omega_1 = \left(1 - \frac{1}{2}\lambda_{max}^2 \right)^{-1}, \qquad \omega_2 = \left(1 - \frac{1}{4}\lambda_{max}^2 \omega_1 \right)^{-1} \text{ and}$$

$$\omega_3 = \left(1 - \frac{1}{4}\lambda_{max}^2 \omega_2 \right)^{-1}. \qquad (5.2.2\text{-}9)$$

In the non-stationary case, we could simply substitute $r = r_k$, in equations (5.2.2-6) and (5.2.2-7). However, we will only consider the solution to the given problem for a constant r, i.e., by using equations (5.2.2-6) and (5.2.2-7).

The Algorithm for this method consists of two parts. The solution for $\mathbf{u}^{(1)}$ in equation (5.2.2-6) is similar to the first iteration of Algorithm 4.2.2-2. The routine calculation is in equation (5.2.2-7) which is similar to the next iteration of Algorithm 4.2.2-2 except for the addition of the term $(1-\omega_k)\ \mathbf{u}^{(k-1)}$ and the multiplications of ω_k to each of the coefficient to the elements of the vector $\mathbf{u}^{(k)}$ and **b**.

5.2.3 The computational complexity

By comparison, the Richardson method and Chebyshev semi-iterative methods need only a little more computational work in relation to the CAGE-DG scheme.

In Richardson's method, the multiplication of ω_{k+1} are required in both the vector $\mathbf{u}^{(k)}$ and **b**. Thus, for large N, it is only $2N$ extra multiplications for every iteration.

The Chebyshev semi-iterative method takes slightly more work. By considering equation (5.2.2-7), we have the multiplication of ω_k for each element involved, and one subtraction plus one multiplication for the second component, i.e., $\mathbf{u}^{(k-1)}$. Thus, for large N, the extra work is only N addition and $2N$ multiplications for every iteration.

5.2.4 Experimental results

The problems given in Section 5.1.5 were investigated further by using the AGE method with the Richardson and Chebyshev semi-iterative methods. Our investigation is again concerned with the gains in the number of iterations, i.e. the rate of convergence.

Problem 1 - A Linear Problem

$$U'' - (1 + x^2)U = -1, \qquad -1 \le x \le 1,$$

$$U(-1) = 0, \; U(1) = 0. \qquad h = 2/(N+1).$$

The computed solution is obtained when $\| \mathbf{r} \| = \| \mathbf{b} - A\mathbf{u}^{(k+1)} \| < \varepsilon$.

The result is tabulated in *Table* 5.2.4-1.

	CAGE-DG Scheme		Richardson Method			Chebyshev Method		
N	Optimum r	Iter	r	ω	Iter	r	ω	Iter
10	0.560 - 0.580	12	0.570	1.0	12	0.570	1.0	12
20	0.303 - 0.305	20	0.304	1.0	20	0.304	1.0	20
40	0.160 - 0.161	39	0.161	1.0	39	0.161	1.0	39
80	0.080	70	0.080	1.0	70	0.080	1.0	70

Table 5.2.4-1: The number of iterations

The results show no gain in terms of the number of iterations. In *Table* 5.2.4-1, the optimal r for the Richardson and Chebyshev method are similar to the optimal r for the CAGE-DG scheme. By having $\omega = 1$, these methods are merely reduced to the CAGE-DG scheme which clearly give the least number of iterations at the given optimal r. This setback is due to similar arguments which have been described for the same problem in Section 5.1.5, i.e., Problem 1. Thus, we may conclude that, the application of single parameter is the best to solve this problem.

Problem 2 - A Mildly Nonlinear Problem.

$$-U'' + \frac{3}{2}U^2 = 0, \qquad 0 \le x \le 1,$$

$$U(0) = 4, \; U(1) = 1. \qquad h = 1/(N+1).$$

The exact solution to this problem is given by $U(x) = 4/(1+x)^2$.

The result are tabulated in *Table* 5.2.4-2.

	CAGE-DG Scheme		Richardson Method			Chebyshev Method		
N	Optimum r	Iter	r	ω	Iter	r	ω	Iter
10	0.572 - 0.579	17	0.575	1.0	17	0.575	1.0	17
20	0.317 - 0.323	35	0.320	1.0	35	0.320	1.0	35
40	0.164 - 0.179	72	0.170	1.0	72	0.170	1.0	72
80	0.089 - 0.093	143	0.090	1.0	143	0.090	1.0	143

Table 5.2.4-2: The number of iterations

This problem, yields a similar situation as the previous one. Since these methods for the linear problem do not give any improved results, we may expect that for the nonlinear problem, it would be even more difficult to obtain a better performance. The values of r and ω should be similar to those given in Problem 1. Thus, for these problems it appears that no advantage is obtained in going to the second order methods.

Problem 3 - A Linear Periodic Problem

$$-U'' + \rho U = (\rho+1)(\sin x + \cos x), \qquad 0 \le x \le 2\pi,$$

$$U(0) = U(2\pi),\ U'(0) = U'(2\pi), \qquad h = 2\pi/(N+1).$$

The exact solution is $U(x) = \sin x + \cos x$.

The matrix $(I - N^{-1}A)$ derived for the CAGE-DG scheme for this type of problem may be referred to in Evans [1990].

The result is tabulated in *Table* 5.2.4-3.

	CAGE-DG Scheme		Richardson Method			Chebyshev Method		
N	Optimum r	Iter	r	ω	Iter	r	ω	Iter
20	3.652 - 4.908	4	4.90	1.00	4	4.90	1.00	4
40	1.283 - 1.807	6	1.70	1.10	4	1.70	1.00	6
80	0.686 - 0.708	9	0.70	1.38	5	0.70	1.00	9
160	0.271 - 0.342	16	0.30	2.00	5	0.30	1.00	16
320	0.138 - 0.154	26	0.15	1.46	18	0.15	1.00	26

Table 5.2.3-3: The number of iterations when $\rho = 70$

The results obtained for the Richardson and Chebyshev methods in *Table* 5.2.3-3 are based on the heuristic search technique and since there appears to be no significant improvement, we discard this technique to obtain the Richardson and Chebyshev parameters. The results in *Tables* 5.1.5-3 and 5.1.5-4, also show that the best performance is obtained when using 2 parameters (PR, Wachspress). However, the Richardson and Chebyshev formulae have also been tested, but did not give any gain in terms of the number of iterations.

The value of ω for both methods is determined as follows. It is obvious that by having $\omega = 1$, both methods are identical to the CAGE-DG scheme for the periodic problem. By varying the value of ω, we can then determine the best ω and r. Although the results shown are only for $\rho = 70$, the gain in the number of iterations is also significant, especially for the Richardson method and larger ρ.

5.3 Summary

This chapter is mainly concerned with the application of multi-parameters to acceleration of the AGE methods. The chapter begins with a discussion of the existing AGE (ADI) parameters followed by other heuristic methods to determine these parameters. The application of multi-parameters is divided into two part. The first part is concerned with the use of the Jordan, PR, Wachspress and other existing parameters, and in the second part, the application of the second order methods, i.e., the Richardson method and the Chebyshev semi-iterative method. Since the size of the block submatrices is small (2x2), we are only interested in applying the 2 or 3 parameters to the AGE method.

The results obtained in Sections 5.1.5 and 5.2.4, show that the application of multi-parameters is only worth while when solving the problems governed by periodic boundary conditions which indicates that the size of the chosen problems is too small. We would expect to obtain better results for large size problems.

It has also been shown that although the eigenvalues of the iteration matrix are well separated, they are of high order in multiplicity. Thus, there is a possibility of rounding errors in the minimisation process which reduces the rate of convergence. The block submatrices in the AGE method are small (2x2) as compared to the ADI method which may have the bigger submatrices, i.e., (40x40). Consequently, this reduction in the size of the submatrices, could reduce the optimal set of multi-parameters to just a single parameter.

By comparison, the Richardson method give slightly better results over the application of Wachspress parameters. However, these parameters are not easily determined compared to the ease of use of the Wachspress parameter. Hence, we may conclude that the application of the Wachspress parameters, i.e., 2 parameter case is highly recommended for use with the AGE method for solving the linear two point boundary value problems governed by periodic boundary conditions.

PART III

THE NUMERICAL SOLUTION OF ELLIPTIC PARTIAL DIFFERENTIAL EQUATIONS

CHAPTER 6

THE OPTIMAL SINGLE AGE PARAMETER FOR ELLIPTIC PDE's

6.1 Determination of the optimal AGE acceleration parameter

The methods discussed in Part II, are mainly concerned with the solution for the two point boundary value problem, i.e, the one dimensional problem. Now, in Part III, we discuss the applicability and the viability of the methods for solving two and three dimensional problems. The scope will be limited to the investigation of the extension of the methods to solve the Helmholtz, Laplace and Poisson equations in two and three dimensions.

As shown in Part II, The PR scheme had a limitation in solving the two and three dimensional problem via the AGE method. Thus, our concern will be centred on utilising the Douglas-Rachford, Douglas and Guittet's schemes. First, we outline the model for two dimensional problem, i.e., the Helmholtz equation.

6.1.1 The two dimensional model problem

Consider the Helmholtz equation in two dimensions

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} - \rho U = f(x,y), \qquad 0 \le x,y \le 1, \qquad (6.1.1\text{-}1)$$

subject to $U = f(x,y)$ on the boundary of the unit square, $0 \le x, y \le 1$ with $\rho \ge 0$ and $f(x,y) \ge 0$. When $\rho = 0$ and $f(x,y) = 0$, the problem is reduced to the well known Laplace equation. If $\rho = 0$ and $f(x,y)$ is a constant but not zero, we then have a Poisson equation.

The square region is covered by a grid with sides parallel to coordinate axes and the grid spacing is h. For $h = 1/(N+1)$, the number of internal grid points or nodes is N^2. Figures 6.1.1-1 shows the nodes for $h = 1/6$, i.e, $N = 5$.

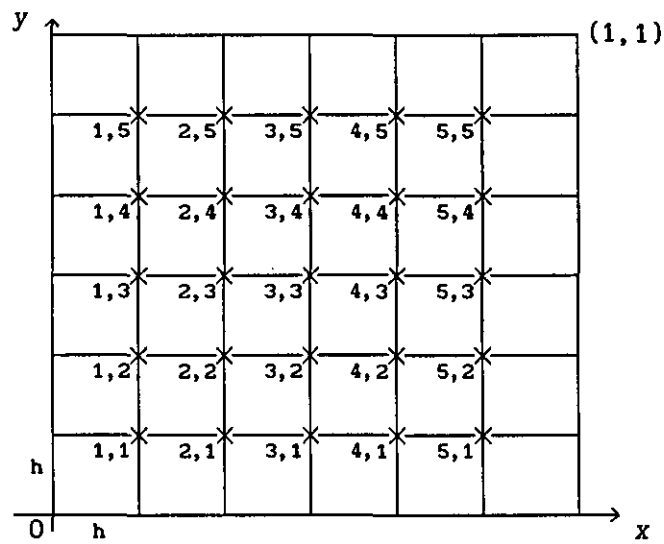


Figure 6.1.1-1: The number of nodes for $h = 1/6$.

The coordinates of a typical internal grid point are $x_i = ih$, $y_j = jh$, where i and j are integers, and the value of u at this grid point is denoted by $u_{i,j}$. For example, from *Figure* 6.1.1-1, the value of u at the point (1,1) is $u_{1,1}$.

By using the Taylor's theorem, we obtain

$$u_{i-1,j} = \left(u - h\frac{\partial U}{\partial x} + \frac{1}{2}h^2\frac{\partial^2 U}{\partial x^2} + \frac{1}{6}h^3\frac{\partial^3 U}{\partial x^3} + \frac{1}{24}h^4\frac{\partial^4 U}{\partial x^4} + \ldots \right)_{i,j} \qquad (6.1.1\text{-}2)$$

and

$$u_{i+1,j} = \left(u + h\frac{\partial U}{\partial x} + \frac{1}{2}h^2\frac{\partial^2 U}{\partial x^2} + \frac{1}{6}h^3\frac{\partial^3 U}{\partial x^3} + \frac{1}{24}h^4\frac{\partial^4 U}{\partial x^4} + \ldots \right)_{i,j} \qquad (6.1.1\text{-}3)$$

By adding (6.1.1-2) and (6.1.1-3), we have

$$u_{i-1,j} + u_{i+1,j} - 2u_{i,j} = \left(h^2\frac{\partial^2 U}{\partial x^2} + \frac{1}{12}h^4\frac{\partial^4 U}{\partial x^4} + \ldots \right)_{i,j} . \qquad (6.1.1\text{-}4)$$

Similarly

$$u_{i,j-1} + u_{i,j+1} - 2u_{i,j} = \left(h^2 \frac{\partial^2 U}{\partial y^2} + \frac{1}{12} h^4 \frac{\partial^4 U}{\partial y^4} + \ldots \right)_{i,j} . \qquad (6.1.1\text{-}5)$$

Now, by adding (6.1.1-4) and (6.1.1-5), we get

$$u_{i-1,j} + u_{i+1,j} - 4u_{i,j} + u_{i,j-1} + u_{i,j+1} =$$

$$\left[h^2 \left(\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} \right) + \frac{1}{12} h^4 \left(\frac{\partial^4 U}{\partial x^4} + \frac{\partial^4 U}{\partial y^4} \right) + \ldots \right]_{i,j} . \qquad (6.1.1\text{-}6)$$

which leads to the *five-point* finite difference replacement

$$u_{i-1,j} + u_{i+1,j} - 4u_{i,j} + u_{i,j-1} + u_{i,j+1} \approx 0 \qquad (6.1.1\text{-}7)$$

for Laplace's equation, with a local truncation error, E_{tr}, given by

$$E_{tr} = \frac{1}{12} h^4 \left(\frac{\partial^4 U}{\partial x^4} + \frac{\partial^4 U}{\partial y^4} \right)_{i,j} + \ldots \qquad (6.1.1\text{-}8)$$

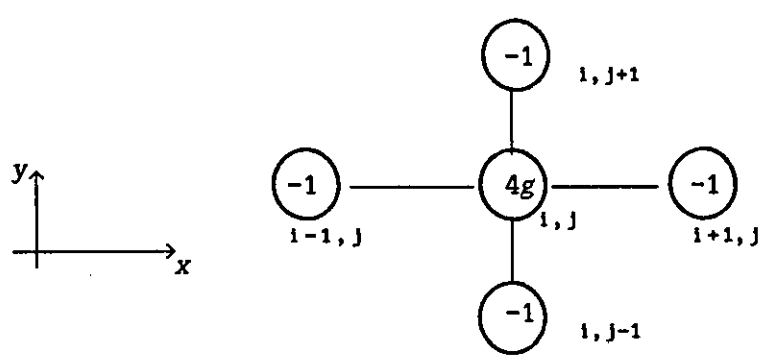
The principal part of the truncation error will only make sense if the derivatives of U are continuous up to order four in x and y.

Now, for the Helmholtz equation (6.1.1-1), we then have the five-point difference equation at the point $u_{i,j}$, $i, j = 1, 2, \ldots, N$,

$$- u_{i-1,j} - u_{i+1,j} + 4g u_{i,j} - u_{i,j-1} - u_{i,j+1} = - h^2 f(x_i, y_j) \qquad (6.1.1\text{-}9)$$

as an approximation to the solution of (6.1.1-1), where h is small.

This equation can be represented as a computational molecular form



(6.1.1-10)

with $i, j = 1, \ldots, N$, where N is the number of points and $g = 1 + \frac{1}{4}\rho h^2$.

Our task now is to solve the linear system $A\mathbf{u} = \mathbf{b}$, derived from equation (6.1.1-9) for the totality of equations, $i, j = 1, 2, \ldots, N$.

From this equation, we have

$$A = \begin{bmatrix} A_1 & -I & & 0 \\ -I & A_1 & -I & \\ & \ddots & \ddots & \ddots \\ & & -I & A_1 & -I \\ 0 & & & -I & A_1 \end{bmatrix}_{N^2 X N^2} \quad \text{with } A_1 = \begin{bmatrix} 4g & -1 & & 0 \\ -1 & 4g & -1 & \\ & \ddots & \ddots & \ddots \\ & & -1 & 4g & -1 \\ 0 & & & -1 & 4g \end{bmatrix}. \tag{6.1.1-11}$$

The vector $\mathbf{u}$ and $\mathbf{b}$ are given by

$$\mathbf{u} = [u_{1,1}, \ldots, u_{N,1}; u_{1,2}, \ldots, u_{N,2}; \ldots; u_{1,N}, \ldots, u_{N,N}]^T \tag{6.1.1-12}$$

and $$\mathbf{b} = [b_{1,1}, \ldots, b_{N,1}; b_{1,2}, \ldots, b_{N,2}; \ldots; b_{1,N}, \ldots, b_{N,N}]^T \tag{6.1.1-13}$$

where the elements of vector $\mathbf{b}$ are as follows:

Set $b_{i,j} = -h^2 f(x_i, y_j)$, $i, j = 1, \ldots, N$. Hence,

1. The elements next to the boundary, parallel to the x-axis are

$$b_{1,1} = b_{1,1} + u_{1,0}, \; b_{1,N} = b_{1,N} + u_{1,N+1}, \; i = 1, \ldots, N$$

and 2. The elements next to the boundary, parallel to the y-axis are

$$b_{1,j} = b_{1,j} + u_{0,j}, \; b_{N,j} = b_{N,j} + u_{N+1,j}, \; j = 1, \ldots, N.$$

We now consider the AGE method for solving $A\mathbf{u} = \mathbf{b}$, which is based on the splitting of the matrix A into

$$A = G_1 + G_2 + G_3 + G_4 \tag{6.1.1-14}$$

where G_1, G_2, G_3 and G_4 are symmetric and positive definite.

These matrices can be shown to be comprised of small (2x2) block systems or can be made so by a suitable permutation on their rows and corresponding column. As shown in the one dimensional case, this

procedure is convenient in the sense that the work required is much less than would be required to solve $A\mathbf{u} = \mathbf{b}$ directly.

For the model problem, let us consider when the case N is odd. Hence, for the two dimensional problem where the matrix A as in (6.1.1-11), we have

$$G_1 + G_2 = \begin{bmatrix} \bar{G} & & & 0 \\ & \bar{G} & & \\ & & \ddots & \\ 0 & & & \bar{G} \end{bmatrix}_{N^2XN^2}, \text{ with } \bar{G} = \begin{bmatrix} 2g & -1 & & 0 \\ -1 & 2g & -1 & \\ & \ddots & \ddots & \ddots \\ 0 & & -1 & 2g \end{bmatrix}$$

and we let $\bar{G} = \bar{G}_1 + \bar{G}_2$, then

$$\bar{G}_1 = \begin{bmatrix} g & -1 & & & \\ -1 & g & & & \\ & & \ddots & & \\ & & & g & -1 \\ & & & -1 & g \end{bmatrix} \text{ and } \bar{G}_2 = \begin{bmatrix} g & & & & \\ & g & -1 & & \\ & -1 & g & & \\ & & & \ddots & \\ & & & & g \end{bmatrix}.$$

Also, we have

$$G_3 + G_4 = \begin{bmatrix} \bar{G}' & -I & & 0 \\ -I & \bar{G}' & -I & \\ & \ddots & \ddots & \ddots \\ 0 & & -I & \bar{G}' \end{bmatrix}_{N^2XN^2}, \quad \bar{G}' = \begin{bmatrix} 2g & & & 0 \\ & 2g & & \\ & & \ddots & \\ 0 & & & 2g \end{bmatrix}.$$

(6.1.1-15)

Evidently by interchanging the order of direction we have $(G_1 + G_2)$ in the form of $(G_3 + G_4)$ and vice-versa.

6.1.2 The three dimensional model problem

To present the model for the three dimensional problem, let us consider the Helmholtz equation in three variables x, y and z

$$\frac{\partial U^2}{\partial x^2} + \frac{\partial U^2}{\partial y^2} + \frac{\partial U^2}{\partial z^2} - \sigma U = g(x,y,z), \qquad (x,y,z) \in \partial R, \tag{6.1.2-1}$$

bounded by

$$U(x,y,z) = f(x,y,z), \qquad (x,y,z) \in \partial R \tag{6.1.2-2}$$

where ∂R is the boundary of the unit cube $0 \le x,y,z \le 1$.

An equally spaced three-dimensional grid is placed over the cube in such a way that it leads to N^3 internal grid points where $h = 1/(N+1)$, with h is the grid spacing. The coordinates of a grid points are

$$x_i = ih, \; y_j = jh, \; z_k = kh \tag{6.1.2-3}$$

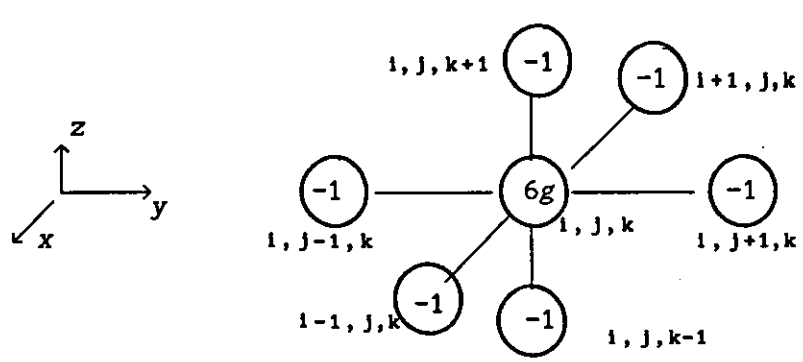
where i, j, k are integers, and the value of u satisfying a difference replacement of equation (6.1.2-1) at this grid point is given by $u_{i,j,k}$.

By following Section 6.1.1, the conventional *seven-point* finite difference replacement of equation (6.1.2-1), for small h, is given by

$$\begin{aligned} &- u_{i-1,j,k} - u_{i+1,j,k} + 6g u_{i,j,k} - u_{i,j-1,k} - u_{i,j+1,k} \\ &\qquad - u_{i,j,k-1} - u_{i,j,k+1} = - h^2 g(x_i, y_j, z_k) \end{aligned} \tag{6.1.2-4}$$

where i, j, $k = 1, \ldots, N$ and $g = 1 + \frac{1}{6}\sigma h^2$.

It is quite tedious to illustrate every internal grid point for a given N, but each grid point can be represented in molecular form as



(6.1.2-5)

The linear system $Au = b$ derived from equation (6.1.2-4) gives

$$A = \begin{bmatrix} A_2 & -J & & 0 \\ -J & A_2 & -J & \\ & \ddots & \ddots & \ddots \\ & & -J & A_2 & -J \\ 0 & & & -J & A_2 \end{bmatrix}_{N^3 X N^3}, \text{ with } J = \begin{bmatrix} I & & & 0 \\ & I & & \\ & & \ddots & \\ & & & I \\ 0 & & & & I \end{bmatrix}_{N^2 X N^2},$$

$$A_2 = \begin{bmatrix} A_1 & -I & & 0 \\ -I & A_1 & -I & \\ & \ddots & \ddots & \ddots \\ & & -I & A_1 & -I \\ 0 & & & -I & A_1 \end{bmatrix}_{N^2 X N^2}, \text{ and } A_1 = \begin{bmatrix} 6g & -1 & & 0 \\ -1 & 6g & -1 & \\ & \ddots & \ddots & \ddots \\ & & -1 & 6g & -1 \\ 0 & & & -1 & 6g \end{bmatrix},$$

(6.1.2-6)

the vector u =

$$[u_{1,1,1}, \ldots, u_{N,1,1}; u_{1,2,1}, \ldots, u_{N,2,1}; \ldots; u_{1,N,1}, \ldots, u_{N,N,1};$$
$$[u_{1,1,2}, \ldots, u_{N,1,2}; u_{1,2,2}, \ldots, u_{N,2,2}; \ldots; u_{1,N,2}, \ldots, u_{N,N,2};$$
$$\ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots$$
$$[u_{1,1,N}, \ldots, u_{N,1,N}; u_{1,2,N}, \ldots, u_{N,2,N}; \ldots; u_{1,N,N}, \ldots, u_{N,N,N}]^T$$

(6.1.2-7)

and the vector b =

$$[b_{1,1,1}, \ldots, b_{N,1,1}; b_{1,2,1}, \ldots, b_{N,2,1}; \ldots; b_{1,N,1}, \ldots, b_{N,N,1};$$
$$[b_{1,1,2}, \ldots, b_{N,1,2}; b_{1,2,2}, \ldots, b_{N,2,2}; \ldots; b_{1,N,2}, \ldots, b_{N,N,2};$$
$$\ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots$$
$$[b_{1,1,N}, \ldots, b_{N,1,N}; b_{1,2,N}, \ldots, b_{N,2,N}; \ldots; b_{1,N,N}, \ldots, b_{N,N,N}]^T$$

(6.1.2-8)

where the elements of vector b are as follows:

Set $b_{i,j,k} = -h^2 g(x_i, y_j, z_k)$, $i, j, k = 1, \ldots, N$. Hence,

1. The elements next to the boundary, parallel to the xy-plane are

$$b_{i,j,1} = b_{i,j,1} + u_{i,j,0},\quad b_{i,j,N} = b_{i,j,N} + u_{i,j,N+1},$$

$$i,\ j = 1,\ \ldots,\ N.$$

2. The elements next to the boundary, parallel to the xz-plane are

$$b_{i,1,k} = b_{i,1,k} + u_{i,0,k},\quad b_{i,N,k} = b_{i,N,k} + u_{i,N+1,k},$$

$$i,\ k = 1,\ \ldots,\ N,$$

and 3. The elements next to the boundary, parallel to the yz-plane are

$$b_{1,j,k} = b_{1,j,k} + u_{0,j,k},\quad b_{N,j,k} = b_{N,j,k} + u_{N+1,j,k},$$

$$j,\ k = 1,\ \ldots,\ N.$$

By applying the AGE method, we consider a splitting of A into six submatrices. i.e,

$$A = G_1 + G_2 + G_3 + G_4 + G_5 + G_6 \tag{6.1.2-9}$$

where G_i, $i = 1,\ \ldots,\ 6$ are symmetric and positive definite.

These matrices can also be shown to be comprised of small (2x2) block systems or can be made so by a suitable permutation on their rows and corresponding column. This procedure is convenient in the sense that work required is much less than would be required to solve $A\mathbf{u} = \mathbf{b}$ directly.

Let us consider when N is odd. Thus, for the model problem where the matrix A as in (6.1.2-6), we have

$$G_1 + G_2 = \begin{bmatrix} G_1' & & 0 \\ & G_1' & \\ & \ddots & \\ 0 & & G_1' \end{bmatrix}_{N^3 \times N^3},\quad \text{with } G_1' = \begin{bmatrix} G'' & & 0 \\ & G'' & \\ & \ddots & \\ 0 & & G'' \end{bmatrix}_{N^2 \times N^2},$$

where

$$G'' = \begin{bmatrix} 2g & -1 & & & 0 \\ -1 & 2g & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2g & -1 \\ 0 & & & -1 & 2g \end{bmatrix}.$$

If we let $G'' = G''_1 + G''_2$, then we have

$$G''_1 = \begin{bmatrix} g & -1 & & & \\ -1 & g & & & \\ & & \ddots & & \\ & & & g & -1 \\ & & & -1 & g \end{bmatrix}, \quad G''_2 = \begin{bmatrix} g & & & \\ & g & -1 & \\ & -1 & g & \\ & & & \ddots \end{bmatrix}$$

and also

$$G_3 + G_4 = \begin{bmatrix} G' & & & 0 \\ & G' & & \\ & & \ddots & \\ 0 & & & G' \end{bmatrix}_{N^3XN^3}, \text{ with } G' = \begin{bmatrix} G''_3 & -I & & 0 \\ -I & G''_3 & -I & \\ & \ddots & \ddots & \ddots \\ 0 & & -I & G''_3 \end{bmatrix}_{N^2XN^2},$$

$$G_5 + G_6 = \begin{bmatrix} \bar{G} & -J & & 0 \\ -J & \bar{G} & -J & \\ & \ddots & \ddots & \ddots \\ 0 & & -J & \bar{G} \end{bmatrix}_{N^3XN^3}, \text{ with } J = \begin{bmatrix} I & & & 0 \\ & I & & \\ & & \ddots & \\ 0 & & & I \end{bmatrix}_{N^2XN^2},$$

$$\text{and } \bar{G} = \begin{bmatrix} G''_3 & & & 0 \\ & G''_3 & & \\ & & \ddots & \\ 0 & & & G''_3 \end{bmatrix}_{N^2XN^2}, \text{ with } G''_3 = \begin{bmatrix} 2g & & & 0 \\ & 2g & & \\ & & \ddots & \\ 0 & & & 2g \end{bmatrix}.$$

(6.1.2-10)

It is obvious that by interchanging the order of direction, we have $(G_1 + G_2)$ in the form of $(G_3 + G_4)$ or $(G_5 + G_6)$ and vice-versa.

With the splitting of the matrix A in both model problems, we will investigate the viability of the Douglas-Rachford, Douglas and Guittet's schemes to solve the problems by using the AGE method.

6.1.3 The Douglas-Rachford (DR) and Douglas schemes

It has previously been shown that the DR and Douglas schemes for the AGE method to solve the two-point boundary-value problem, i.e., the problem in one dimension, is given by

$$(rI + G_1)\mathbf{u}^{(k+1/2)} = [(rI + G_1) - \omega A]\mathbf{u}^{(k)} + \omega\mathbf{b} \qquad (6.1.3\text{-}1)$$

$$(rI + G_2)\mathbf{u}^{(k+1)} = r\mathbf{u}^{(k+1/2)} + G_2\mathbf{u}^{(k)} \qquad (6.1.3\text{-}2)$$

where $A = G_1 + G_2$ and $r > 0$ is the iteration parameter. When $\omega = 1$, we have the DR scheme, and for $\omega = 2$, we have the Douglas scheme. The scheme can also be extended to solve the higher dimensional problems.

Consider the two dimensional problem where the splitting of A into the four submatrices is as in (6.1.1-14). The AGE method in the DR and Douglas (AGE-DG-2) schemes can be presented as

$$(rI + G_1)\mathbf{u}^{(k+1/4)} = [(rI + G_1) - \omega A]\mathbf{u}^{(k)} + \omega\mathbf{b} \qquad (6.1.3\text{-}3)$$

$$(rI + G_2)\mathbf{u}^{(k+1/2)} = r\mathbf{u}^{(k+1/4)} + G_2\mathbf{u}^{(k)} \qquad (6.1.3\text{-}4)$$

$$(rI + G_3)\mathbf{u}^{(k+3/4)} = r\mathbf{u}^{(k+1/2)} + G_3\mathbf{u}^{(k)} \qquad (6.1.3\text{-}5)$$

$$(rI + G_4)\mathbf{u}^{(k+1)} = r\mathbf{u}^{(k+3/4)} + G_4\mathbf{u}^{(k)} \qquad (6.1.3\text{-}6)$$

with $\omega = 1$, for the DR scheme and $\omega = 2$ for the Douglas scheme.

This scheme corresponds to sweeping through the mesh parallel to the coordinate x and y axes. At each stage of iteration, we solve the 2X2 block systems. It is obvious that the vector $\mathbf{u}^{(k+1)}$ is computed in four steps.

We now seek to analyse the convergence of the AGE-DG-2 scheme, i.e., equations (6.1.3-3) - (6.1.3-6).

It is obvious that the AGE-DG-2 scheme is consistent. We now show that this scheme is convergent. By eliminating all the intermediate vectors of $\mathbf{u}$, we then have the iteration matrix as

$$T_r = I - \omega r^3 \prod_{i=4}^{1} (rI + G_i)^{-1} A. \tag{6.1.3-7}$$

Let α, β, γ and δ are the respective eigenvalues of G_1, G_2, G_3 and G_4. Since G_i, $i = 1, \ldots, 4$ are symmetric and positive definite, hence all the eigenvalues are positive. By inspection, it can be shown that these eigenvalues are in the interval $[0,2]$.

We need to show that $\mathcal{S}(T_r) < 1$.

$$\begin{aligned}
\mathcal{S}(T_r) &= \|T_r\|_2 \\
&= \|I - \omega r^3 \prod_{i=4}^{1} (rI + G_i)^{-1} A\|_2 \\
&= \left|1 - \omega r^3 \left(\frac{1}{r+\alpha}\right)\left(\frac{1}{r+\beta}\right)\left(\frac{1}{r+\gamma}\right)\left(\frac{1}{r+\delta}\right)(\alpha+\beta+\gamma+\delta)\right| \\
&= \left|1 - \frac{2\omega r^3(\mu+\nu)}{(r+\mu)^2(r+\nu)^2}\right|, \text{ where } \mu = max\ (\alpha,\beta),\ \nu = max\ (\gamma,\delta), \\
&= \left|1 - \frac{4\omega r^3 \lambda}{(r+\lambda)^4}\right| < 1, \text{ where } \lambda = max\ (\mu,\nu), \text{ for } r > 0.
\end{aligned}$$

Thus, the AGE-DG-2 scheme is convergent.

In programming, we only need to consider the matrices G_1 and G_2 since after changing the direction, i.e., from row to column, equations (6.1.3-5) and (6.1.3-6) can be written as

$$(rI + G_1)\mathbf{u}_c^{(k+3/4)} = r\mathbf{u}_c^{(k+1/2)} + G_1\mathbf{u}_c^{(k)} \tag{6.1.3-8}$$

$$(rI + G_2)\mathbf{u}_c^{(k+1)} = r\mathbf{u}_c^{(k+3/4)} + G_2\mathbf{u}_c^{(k)}. \tag{6.1.3-9}$$

where $\mathbf{u}_c$ stands columnwise reordering. Hence, the new set of equations for the AGE-DG-2 scheme is given as

$$(rI + G_1)u^{(k+1/4)} = [(rI + G_1) - \omega A]u^{(k)} + \omega b \qquad (6.1.3\text{-}10)$$

$$(rI + G_2)u^{(k+1/2)} = ru^{(k+1/4)} + G_2u^{(k)} \qquad (6.1.3\text{-}11)$$

$$(rI + G_1)u_c^{(k+3/4)} = ru_c^{(k+1/2)} + G_1u_c^{(k)} \qquad (6.1.3\text{-}12)$$

$$(rI + G_2)u_c^{(k+1)} = ru_c^{(k+3/4)} + G_2u_c^{(k)}. \qquad (6.1.3\text{-}13)$$

The AGE-DG-2 scheme in equations (6.1.3-10) - (6.1.3-13) also converge and its convergence can be shown in similar way.

It is obvious that the matrices G_1 and G_2 consist of the 2X2 block submatrices $\hat{G}$, where

$$\hat{G} = \begin{bmatrix} g & -1 \\ -1 & g \end{bmatrix}, \text{ with } g = 1 + \tfrac{1}{4}\rho h^2 . \qquad (6.1.3\text{-}14)$$

Hence, for any $r > 0$, the matrices $(rI + G_1)$ and $(rI + G_2)$ are the form of $(rI + \hat{G})$, where

$$(rI + \hat{G}) = \begin{bmatrix} \alpha & -1 \\ -1 & \alpha \end{bmatrix}, \text{ with } \alpha = r + g \qquad (6.1.3\text{-}15)$$

and the inverse, $(rI + G_1)^{-1}$ and $(rI + G_2)^{-1}$ are in the form

$$(rI + \hat{G})^{-1} = d\begin{bmatrix} \alpha & 1 \\ 1 & \alpha \end{bmatrix}, \text{ with } d = \frac{1}{\alpha^2 - 1} . \qquad (6.1.3\text{-}16)$$

From equation (6.1.3-10), we set $C = (rI + G_1) - \omega A$.

Let us consider when $N = 5$. Thus, we have

$$C = \begin{bmatrix} C_1 & \omega I & & & 0 \\ \omega I & C_1 & \omega I & & \\ & \omega I & C_1 & \omega I & \\ & & \omega I & C_1 & \omega I \\ 0 & & & \omega I & C_1 \end{bmatrix}_{5^2X5^2} \quad \text{with } C_1 = \begin{bmatrix} t & s & & & 0 \\ s & t & \omega & & \\ & \omega & t & s & \\ & & s & t & \omega \\ 0 & & & \omega & t \end{bmatrix},$$

where $t = \alpha - 4g\omega$ and $s = \omega - 1$.

We now write the algorithm for general N for the AGE-DG-2 scheme using equations (6.1.3-10) - (6.1.3-13). In this algorithm, the number of points, N is considered odd.

Algorithm 6.1.3-1: The AGE-DG-2 scheme.

Set $u_{i,j}^{(k)} = 0$, $i, j = 0, \ldots, N+1$, $\alpha = r + g$, $d = 1/(\alpha^2 - 1)$,

$t = \alpha - 4g\omega$, $s = \omega - 1$, $\alpha_1 = \alpha d$.

Step 1: To compute $u^{(k+1/4)}$. Set $i, j = 1$.

while $j \le N$, compute

while $i \le N-2$, compute

$$r_1 = \omega u_{i,j-1}^{(k)} + \omega u_{i-1,j}^{(k)} + t u_{i,j}^{(k)} + s u_{i+1,j}^{(k)} + \omega u_{i,j+1}^{(k)} + \omega b_{i,j}$$

$$r_2 = \omega u_{i+1,j-1}^{(k)} + s u_{i,j}^{(k)} + t u_{i+1,j}^{(k)} + \omega u_{i+2,j}^{(k)} + \omega u_{i+1,j+1}^{(k)} + \omega b_{i+1,j}$$

$$u_{i,j}^{(k+1/4)} = \alpha_1 r_1 + r_2 d, \quad u_{i+1,j}^{(k+1/4)} = r_1 d + \alpha_1 r_2$$

$i = i + 2$

$$u_{N,j}^{(k+1/4)} = (\omega u_{N,j-1}^{(k)} + \omega u_{N-1,j}^{(k)} + t u_{N,j}^{(k)} + \omega u_{N,j+1}^{(k)} + \omega b_{N,j})/\alpha$$

$j = j + 1$.

Step 2: To compute $u^{(k+1/2)}$. Set $i = 2$, $j = 1$.

while $j \le N$, compute

$$u_{1,j}^{(k+1/2)} = (r u_{1,j}^{(k+1/4)} + g u_{1,j}^{(k)})/\alpha$$

while $i \le N-1$, compute

$$r_1 = r u_{i,j}^{(k+1/4)} + g u_{i,j}^{(k)} - u_{i+1,j}^{(k)}$$

$$r_2 = r u_{i+1,j}^{(k+1/4)} - u_{i,j}^{(k)} + g u_{i+1,j}^{(k)}$$

$$u_{i,j}^{(k+1/2)} = \alpha_1 r_1 + r_2 d, \quad u_{i+1,j}^{(k+1/2)} = r_1 d + \alpha_1 r_2$$

$i = i + 2$

$j = j + 1.$

Step 3: To compute $\mathbf{u}^{(k+3/4)}$. Set i, $j = 1$. (Change in direction).

while $i \le N$, compute

while $j \le N-2$, compute

$$r_1 = ru_{i,j}^{(k+1/2)} + gu_{i,j}^{(k)} - u_{i,j+1}^{(k)}$$

$$r_2 = ru_{i,j+1}^{(k+1/2)} - u_{i,j}^{(k)} + gu_{i,j+1}^{(k)}$$

$$u_{i,j}^{(k+3/4)} = \alpha_1 r_1 + r_2 d, \quad u_{i,j+1}^{(k+3/4)} = r_1 d + \alpha_1 r_2$$

$j = j + 2$

$$u_{i,N}^{(k+3/4)} = (ru_{i,N}^{(k+1/2)} + gu_{i,N}^{(k)})/\alpha$$

$i = i + 1.$

Step 4: To compute $\mathbf{u}^{(k+1)}$. Set $i = 1$, $j = 2$. (change in direction).

while $i \le N$, compute

$$u_{i,1}^{(k+1)} = (ru_{i,1}^{(k+3/4)} + gu_{i,1}^{(k)})/\alpha$$

while $j \le N-1$, compute

$$r_1 = ru_{i,j}^{(k+3/4)} + gu_{i,j}^{(k)} - u_{i,j+1}^{(k)}$$

$$r_2 = ru_{i,j+1}^{(k+3/4)} - u_{i,j}^{(k)} + gu_{i,j+1}^{(k)}$$

$$u_{i,j}^{(k+1)} = \alpha_1 r_1 + r_2 d, \quad u_{i,j+1}^{(k+1)} = r_1 d + \alpha_1 r_2$$

$j = j + 2$

$i = i + 1.$

Step 5: Repeat *Step* 1 to *Step* 4 until convergence is achieved.

We now investigate the DR and Douglas scheme for the 3 dimensional problem using the AGE method, the AGE-DG-3 scheme.

Let us recall the splitting of A in (6.1.2-7). Then, the AGE-DG-3 scheme can be presented as

$$(rI + G_1)u^{(k+1/6)} = [(rI + G_1) - \omega A]u^{(k)} + \omega b \qquad (6.1.3\text{-}17)$$

$$(rI + G_2)u^{(k+1/3)} = ru^{(k+1/6)} + G_2u^{(k)} \qquad (6.1.3\text{-}18)$$

$$(rI + G_3)u^{(k+1/2)} = ru^{(k+1/3)} + G_3u^{(k)} \qquad (6.1.3\text{-}19)$$

$$(rI + G_4)u^{(k+2/3)} = ru^{(k+1/2)} + G_4u^{(k)} \qquad (6.1.3\text{-}20)$$

$$(rI + G_5)u^{(k+5/6)} = ru^{(k+2/3)} + G_5u^{(k)} \qquad (6.1.3\text{-}21)$$

$$(rI + G_6)u^{(k+1)} = ru^{(k+5/6)} + G_6u^{(k)} \qquad (6.1.3\text{-}22)$$

with $\omega = 1$, for the DR scheme and $\omega = 2$ takes the Douglas scheme.

This scheme corresponds to sweeping through the mesh parallel to the three coordinate axes x, y and z. At each stage of the iteration, we solve the 2X2 block systems. It is obvious that the vector $u^{(k+1)}$ is computed in six steps.

We now seek to analyse the convergence of the AGE-DG-3 scheme, i.e., equations (6.1.3-14) - (6.1.3-19).

It is obvious that the AGE-DG-3 scheme is consistent. We now show that this scheme is convergent. By eliminating the intermediate vectors u, we then have the iteration matrix as

$$T_r = I - \omega r^5 \prod_{i=6}^{1} (rI + G_i)^{-1}A. \qquad (6.1.3\text{-}20)$$

Let α, β, γ, τ, σ and δ are the respective eigenvalues of G_1, G_2, G_3, G_4, G_5 and G_6. Since G_i, $i = 1, \ldots, 6$ are symmetric and positive definite, hence all the eigenvalues are positive. By inspection, it can be shown that these eigenvalues are in the interval [0,2].

We need to show that $\rho(T_r) < 1$.

$$\rho(T_r) = \|T_r\|_2$$

$$= \|I - \omega r^5 \prod_{i=6}^{1} (rI + G_i)^{-1}A\|_2$$

$$= \left|1 - \omega r^5 \left(\frac{1}{r+\alpha}\right)\left(\frac{1}{r+\beta}\right)\left(\frac{1}{r+\gamma}\right)\left(\frac{1}{r+\tau}\right)\left(\frac{1}{r+\sigma}\right)\left(\frac{1}{r+\delta}\right) \times (\alpha + \beta + \gamma + \tau + \sigma + \delta)\right|$$

$$= \left| 1 - \frac{2\omega r^5 (a + b + c)}{(r + a)^2 (r + b)^2 (r + c)^2} \right|,$$

where $a = max\ (\alpha, \beta)$, $b = max\ (\gamma, \tau)$ and $c = max\ (\sigma, \delta)$.

If $\lambda = max\ (a, b, c)$, then

$$\varphi(T_r) = \left| 1 - \frac{6\omega r^5 \lambda}{(r + \lambda)^6} \right| < 1, \text{ for } r > 0.$$

Thus, the AGE-DG-3 scheme is convergent.

Again, in programming, we only need to consider G_1 and G_2 matrices since after reordering the direction, the new set of equations becomes

$$(rI + G_1)\mathbf{u}^{(k+1/6)} = [(rI + G_1) - \omega A]\mathbf{u}^{(k)} + \omega \mathbf{b} \qquad (6.1.3\text{-}21)$$

$$(rI + G_2)\mathbf{u}^{(k+1/3)} = r\mathbf{u}^{(k+1/6)} + G_2\mathbf{u}^{(k)} \qquad (6.1.3\text{-}22)$$

$$(rI + G_1)\mathbf{u}_y^{(k+1/2)} = r\mathbf{u}_y^{(k+1/3)} + G_1\mathbf{u}_y^{(k)} \qquad (6.1.3\text{-}23)$$

$$(rI + G_2)\mathbf{u}_y^{(k+2/3)} = r\mathbf{u}_y^{(k+1/2)} + G_2\mathbf{u}_y^{(k)} \qquad (6.1.3\text{-}24)$$

$$(rI + G_1)\mathbf{u}_z^{(k+5/6)} = r\mathbf{u}_z^{(k+2/3)} + G_1\mathbf{u}_z^{(k)} \qquad (6.1.3\text{-}25)$$

$$(rI + G_2)\mathbf{u}_z^{(k+1)} = r\mathbf{u}_z^{(k+5/6)} + G_2\mathbf{u}_z^{(k)} \qquad (6.1.3\text{-}26)$$

where $\mathbf{u}_y$ stands for the y-direction and $\mathbf{u}_z$ stands for the z-direction. The scheme (6.1.3-21) - (6.1.3-26) also converges and its convergence can be shown in similar way.

It is obvious that the matrices G_1 and G_2 in the AGE-DG-3 scheme consist of the 2X2 block submatrices $\hat{G}$, where

$$\hat{G} = \begin{bmatrix} g & -1 \\ -1 & g \end{bmatrix}, \text{ with } g = 1 + \tfrac{1}{6}\sigma h^2 . \qquad (6.1.3\text{-}27)$$

Hence, for any $r > 0$, the matrices $(rI + G_1)$ and $(rI + G_2)$ are in the form of $(rI + \hat{G})$ in (6.1.3-15), whilst the inverse, $(rI + G_1)^{-1}$ and $(rI + G_2)^{-1}$ are in the form of $(rI + \hat{G})^{-1}$ in (6.1.3-16).

Let us consider the case when $N = 5$. From equation (6.1.3-21), we set $C = (rI + G_1) - \omega A$. Thus, we have

$$C = \begin{bmatrix} C_1 & \omega I & & & 0 \\ \omega I & C_1 & \omega I & & \\ & \omega I & C_1 & \omega I & \\ & & \omega I & C_1 & \omega I \\ 0 & & & \omega I & C_1 \end{bmatrix}_{5^3 X 5^3}, \text{ for}$$

$$C_1 = \begin{bmatrix} C_2 & \omega I & & & 0 \\ \omega I & C_2 & \omega I & & \\ & \omega I & C_2 & \omega I & \\ & & \omega I & C_2 & \omega I \\ 0 & & & \omega I & C_2 \end{bmatrix}_{5^2 X 5^2} \quad \text{with } C_2 = \begin{bmatrix} t & s & & 0 & \\ s & t & \omega & & \\ & \omega & t & s & \\ & & s & t & \omega \\ 0 & & & \omega & t \end{bmatrix},$$

where $t = \alpha - 6g\omega$ and $s = \omega - 1$.

We now write the algorithm for general N for the AGE-DG-3 scheme using equations (6.1.3-21) - (6.1.3-26). In this algorithm, the number of points, N is considered odd.

Algorithm 6.1.3-1: The AGE-DG-3 scheme.

Set $u^{(k)}_{i,j,k} = 0$, i, j, $k = 0, \ldots, N+1$, $\alpha = r + g$,

$d = 1/(\alpha^2 - 1)$, $t = \alpha - 6g\omega$, $s = \omega - 1$, $\alpha_1 = \alpha d$.

Step 1: To compute $\mathbf{u}^{(k+1/4)}$. Set i, j, $k = 1$.

while $k \le N$, compute

while $j \le N$, compute

while $i \le N-2$, compute

$$r_1 = \omega u^{(k)}_{i,j,k-1} + \omega u^{(k)}_{i,j-1,k} + \omega u^{(k)}_{i-1,j,k} + t u^{(k)}_{i,j,k} + s u^{(k)}_{i+1,j,k} + \omega u^{(k)}_{i,j+1,k} + \omega u^{(k)}_{i,j,k+1} + \omega b_{i,j,k}$$

$$r_2 = \omega u^{(k)}_{i+1,j,k-1} + \omega u^{(k)}_{i+1,j-1,k} + s u^{(k)}_{i,j,k} + t u^{(k)}_{i+1,j,k} + \omega u^{(k)}_{i+2,j,k} + \omega u^{(k)}_{i+1,j+1,k} + \omega u^{(k)}_{i+1,j,k+1} + \omega b_{i+1,j,k}$$

$$u_{i,j,k}^{(k+1/6)} = \alpha_1 r_1 + r_2 d, \quad u_{i+1,j,k}^{(k+1/6)} = r_1 d + \alpha_1 r_2$$

$i = i + 2$

$$u_{N,j,k}^{(k+1/6)} = (\omega u_{N,j,k-1}^{(k)} + \omega u_{N,j-1,k}^{(k)} + \omega u_{N-1,j,k}^{(k)} + t u_{N,j,k}^{(k)} + \omega u_{N,j+1,k}^{(k)} + \omega u_{N,j,k+1}^{(k)} + \omega b_{N,j,k})/\alpha$$

$j = j + 1$

$k = k + 1.$

Step 2: To compute $\mathbf{u}^{(k+1/3)}$. Set $i = 2$, j, $k = 1$.

while $k \le N$, compute

while $j \le N$, compute

$$u_{1,j,k}^{(k+1/3)} = (r u_{1,j,k}^{(k+1/6)} + g u_{1,j,k}^{(k)})/\alpha$$

while $i \le N-1$, compute

$$r_1 = r u_{i,j,k}^{(k+1/6)} + g u_{i,j,k}^{(k)} - u_{i+1,j,k}^{(k)}$$

$$r_2 = r u_{i+1,j,k}^{(k+1/6)} - u_{i,j,k}^{(k)} + g u_{i+1,j,k}^{(k)}$$

$$u_{i,j,k}^{(k+1/3)} = \alpha_1 r_1 + r_2 d, \quad u_{i+1,j,k}^{(k+1/3)} = r_1 d + \alpha_1 r_2$$

$i = i + 2$

$j = j + 1$

$k = k + 1.$

Step 3: To compute $\mathbf{u}^{(k+1/2)}$. Set i, j, $k = 1$. (Change in direction)

while $i \le N$, compute

while $k \le N$, compute

while $j \le N-1$, compute

$$r_1 = r u_{i,j,k}^{(k+1/3)} + g u_{i,j,k}^{(k)} - u_{i,j+1,k}^{(k)}$$

$$r_2 = r u_{i,j+1,k}^{(k+1/3)} - u_{i,j,k}^{(k)} + g u_{i,j+1,k}^{(k)}$$

$$u_{i,j,k}^{(k+1/2)} = \alpha_1 r_1 + r_2 d, \quad u_{i,j+1,k}^{(k+1/2)} = r_1 d + \alpha_1 r_2$$

$j = j + 2$

$$u_{i,N,k}^{(k+1/2)} = (ru_{i,N,k}^{(k+1/3)} + gu_{i,N,k}^{(k)})/\alpha$$

$k = k + 1$

$i = i + 1.$

Step 4: To compute $u^{(k+2/3)}$. Set $j = 2$, i, $k = 1$. (Change in direction)

while $i \leq N$, compute

while $k \leq N$, compute

$$u_{i,1,k}^{(k+2/3)} = (ru_{i,1,k}^{(k+1/2)} + gu_{i,1,k}^{(k)})/\alpha$$

while $j \leq N-1$, compute

$$r_1 = ru_{i,j,k}^{(k+1/2)} + gu_{i,j,k}^{(k)} - u_{i,j+1,k}^{(k)}$$

$$r_2 = ru_{i,j+1,k}^{(k+1/2)} - u_{i,j,k}^{(k)} + gu_{i,j+1,k}^{(k)}$$

$$u_{i,j,k}^{(k+2/3)} = \alpha_1 r_1 + r_2 d, \quad u_{i,j+1,k}^{(k+2/3)} = r_1 d + \alpha_1 r_2$$

$j = j + 2$

$k = k + 1$

$i = i + 1.$

Step 5: To compute $u^{(k+5/6)}$. Set i, j, $k = 1$. (Change in direction)

while $j \leq N$, compute

while $i \leq N$, compute

while $k \leq N-1$, compute

$$r_1 = ru_{i,j,k}^{(k+2/3)} + gu_{i,j,k}^{(k)} - u_{i,j,k+1}^{(k)}$$

$$r_2 = ru_{i,j,k+1}^{(k+2/3)} - u_{i,j,k}^{(k)} + gu_{i,j,k+1}^{(k)}$$

$$u_{i,j,k}^{(k+5/6)} = \alpha_1 r_1 + r_2 d, \quad u_{i,j,k+1}^{(k+5/6)} = r_1 d + \alpha_1 r_2$$

$k = k + 2$

$$u_{i,j,N}^{(k+5/6)} = (ru_{i,j,N}^{(k+2/3)} + gu_{i,j,N}^{(k)})/\alpha$$

$i = i + 1$

$j = j + 1.$

Step 6: To compute $\mathbf{u}^{(k+1)}$. Set i, $j = 1$, $k = 2$. (Change in direction)

while $j \le N$, compute

while $i \le N$, compute

$$u_{i,j,1}^{(k+1)} = (ru_{i,j,1}^{(k+5/6)} + gu_{i,j,1}^{(k)})/\alpha$$

while $k \le N-1$, compute

$$r_1 = ru_{i,j,k}^{(k+5/6)} + gu_{i,j,k}^{(k)} - u_{i,j,k+1}^{(k)}$$

$$r_2 = ru_{i,j,k+1}^{(k+5/6)} - u_{i,j,k}^{(k)} + gu_{i,j,k+1}^{(k)}$$

$$u_{i,j,k}^{(k+1)} = \alpha_1 r_1 + r_2 d, \quad u_{i,j,k+1}^{(k+1)} = r_1 d + \alpha_1 r_2$$

$k = k + 2$

$i = i + 1$

$j = j + 1$.

Step 7: Repeat *Step* 1 to *Step* 6 until convergence is achieved.

6.1.4 The Guittet formula

Now Guittet's formula has been shown to successfully solve the two-point boundary-value problem by using the AGE method. Guittet [1967], has shown that this scheme can be extended to solve the problem with the higher dimension.

For the two dimensional problem with the splitting of A as in (6.1.1-14), the AGE method in Guittet's formula, the AGE-GT-2 scheme can be presented as

$$(rI + G_1)\mathbf{u}^{(k+1/4)} = [\prod_{i=1}^{4} (rI + G_i) - \omega r^3 A]\mathbf{u}^{(k)} + \omega r^3 \mathbf{b} \qquad (6.1.4\text{-}1)$$

$$(rI + G_2)\mathbf{u}^{(k+1/2)} = \mathbf{u}^{(k+1/4)} \qquad (6.1.4\text{-}2)$$

$$(rI + G_3)\mathbf{u}^{(k+3/4)} = \mathbf{u}^{(k+1/2)} \qquad (6.1.4\text{-}3)$$

$$(rI + G_4)\mathbf{u}^{(k+1)} = \mathbf{u}^{(k+3/4)} \qquad (6.1.4\text{-}4)$$

where $r > 0$, r is the iteration parameter.

After eliminating the intermediate vectors **u**, we have the iteration matrix for the AGE-GT-2 scheme as,

$$T_r = I - \omega r^3 \prod_{i=4}^{1} (rI + G_i)^{-1} A \qquad (6.1.4\text{-}5)$$

which is similar to the iteration matrix of the AGE-DG-2 scheme. Thus, the AGE-GT-2 scheme converges and the proof of convergence is similar to the AGE-DG-2 scheme.

In programming, the computational work can be made more easier by interchanging the rows and columns in equations (6.1.4-3) and (6.1.4-4). However, it is better to consider the matrices G_3 and G_4 as part of equation (6.1.4-1) as this leads to a simpler computation for $\mathbf{u}^{(k+1/4)}$. Thus, the new set of equations become

$$(rI + G_1)\mathbf{u}^{(k+1/4)} = [\prod_{i=1}^{4} (rI + G_i) - \omega r^3 A]\mathbf{u}^{(k)} + \omega r^3 \mathbf{b} \qquad (6.1.4\text{-}6)$$

$$(rI + G_2)\mathbf{u}^{(k+1/2)} = \mathbf{u}^{(k+1/4)} \qquad (6.1.4\text{-}7)$$

$$(rI + G_1)\mathbf{u}_c^{(k+3/4)} = \mathbf{u}_c^{(k+1/2)} \qquad (6.1.4\text{-}8)$$

$$(rI + G_2)\mathbf{u}_c^{(k+1)} = \mathbf{u}_c^{(k+3/4)} \qquad (6.1.4\text{-}9)$$

where c denotes columnwise ordering.

The scheme also converges and its convergence can be shown in a similar way. The three dimensional problem with the splitting of A in (6.1.2-9), gives the AGE method in Guittet's formula, the AGE-GT-3 scheme as

$$(rI + G_1)\mathbf{u}^{(k+1/6)} = [\prod_{i=1}^{6} (rI + G_i) - \omega r^5 A]\mathbf{u}^{(k)} + \omega r^5 \mathbf{b} \qquad (6.1.4\text{-}10)$$

$$(rI + G_2)\mathbf{u}^{(k+1/3)} = \mathbf{u}^{(k+1/6)} \qquad (6.1.4\text{-}11)$$

$$(rI + G_3)\mathbf{u}^{(k+1/2)} = \mathbf{u}^{(k+1/3)} \qquad (6.1.4\text{-}12)$$

$$(rI + G_4)\mathbf{u}^{(k+2/3)} = \mathbf{u}^{(k+1/2)} \qquad (6.1.4\text{-}13)$$

$$(rI + G_5)\mathbf{u}^{(k+5/6)} = \mathbf{u}^{(k+2/3)} \qquad (6.1.4\text{-}14)$$

$$(rI + G_6)\mathbf{u}^{(k+1)} = \mathbf{u}^{(k+5/6)} \qquad (6.1.4\text{-}15)$$

where $r > 0$, r is the iteration parameter.

After eliminating the intermediate vectors **u**, we have the iteration matrix for the AGE-GT-3 scheme as,

$$T_r = I - \omega r^5 \prod_{i=6}^{1} (rI + G_i)^{-1} A \qquad (6.1.4\text{-}16)$$

which is similar to the iteration matrix of the AGE-DG-3 scheme.

Thus, the AGE-GT-3 scheme converges and the proof of convergence is similar to the AGE-DG-3 scheme.

As in the AGE-GT-2 scheme, the computation can be made simpler by considering the interchange between the x, y and z directions. However, the matrices G_3, G_4, G_5 and G_6 may be kept in equation (6.1.4-10) to ease the computational effort. Thus, we only need to consider the AGE-GT-3 scheme in the form

$$(rI + G_1)\mathbf{u}^{(k+1/6)} = [\prod_{i=1}^{6} (rI + G_i) - \omega r^5 A]\mathbf{u}^{(k)} + \omega r^5 \mathbf{b} \qquad (6.1.4\text{-}17)$$

$$(rI + G_2)\mathbf{u}^{(k+1/3)} = \mathbf{u}^{(k+1/6)} \qquad (6.1.4\text{-}18)$$

$$(rI + G_1)\mathbf{u}_y^{(k+1/2)} = \mathbf{u}_y^{(k+1/3)} \qquad (6.1.4\text{-}19)$$

$$(rI + G_2)\mathbf{u}_y^{(k+2/3)} = \mathbf{u}_y^{(k+1/2)} \qquad (6.1.4\text{-}20)$$

$$(rI + G_1)\mathbf{u}_z^{(k+5/6)} = \mathbf{u}_z^{(k+2/3)} \qquad (6.1.4\text{-}21)$$

$$(rI + G_2)\mathbf{u}_z^{(k+1)} = \mathbf{u}_z^{(k+5/6)} \qquad (6.1.4\text{-}22)$$

where y and z stands for the ordering in the y and z directions.

The scheme also converges and its convergence can be shown in a similar way.

The disadvantage with the AGE-GT-2 and AGE-GT-3 schemes is in the labourious calculation for obtaining the matrix

$$C_2 = \prod_{i=1}^{4} (rI + G_i) - \omega r^3 A \qquad (6.1.4\text{-}23)$$

for the AGE-GT-2 scheme, and

$$C_3 = \prod_{i=1}^{6} (rI + G_1) - \omega r^5 A \qquad (6.1.4\text{-}24)$$

for the AGE-GT-3 scheme. However, this is compensated by a simple calculation for the remaining equations.

We now illustrate the computational algorithm to determine the matrix C_2 in (6.1.4-23).

Let consider when $N = 7$. Thus,

$$A = \begin{bmatrix} A_1 & -I & & & & & \\ -I & A_1 & -I & & & 0 & \\ & -I & A_1 & -I & & & \\ & & -I & A_1 & -I & & \\ & & & -I & A_1 & -I & \\ & 0 & & & -I & A_1 & -I \\ & & & & & -I & A_1 \end{bmatrix}_{7^2 X 7^2},$$

where $A_1 = \begin{bmatrix} 4g & -1 & & & & & \\ -1 & 4g & -1 & & & 0 & \\ & -1 & 4g & -1 & & & \\ & & -1 & 4g & -1 & & \\ & & & -1 & 4g & -1 & \\ & 0 & & & -1 & 4g & -1 \\ & & & & & -1 & 4g \end{bmatrix}_{7 X 7}$.

The other matrices are given as follows:

$$(rI + G_1) = \begin{bmatrix} G_1' & & & & & & \\ & G_1' & & & & 0 & \\ & & G_1' & & & & \\ & & & G_1' & & & \\ & & & & G_1' & & \\ & 0 & & & & G_1' & \\ & & & & & & G_1' \end{bmatrix}_{7^2 X 7^2},$$

$$(rI + G_2) = \begin{bmatrix} G_2' & & & & & & \\ & G_2' & & & & 0 & \\ & & G_2' & & & & \\ & & & G_2' & & & \\ & & & & G_2' & & \\ & 0 & & & & G_2' & \\ & & & & & & G_2' \end{bmatrix}_{7^2 X 7^2},$$

where

$$G_1' = \left[\begin{array}{cc|cc|cc|c} \alpha & -1 & & & & & \\ -1 & \alpha & & & & & \\ \hline & & \alpha & -1 & & & \\ & & -1 & \alpha & & & \\ \hline & & & & \alpha & -1 & \\ & & & & -1 & \alpha & \\ \hline & & & & & & \alpha \end{array}\right], \quad G_2' = \left[\begin{array}{c|cc|cc|cc} \alpha & & & & & & \\ \hline & \alpha & -1 & & & & \\ & -1 & \alpha & & & & \\ \hline & & & \alpha & -1 & & \\ & & & -1 & \alpha & & \\ \hline & & & & & \alpha & -1 \\ & & & & & -1 & \alpha \end{array}\right].$$

and where $\alpha = r + g$.

Having chosen G_3 and G_4 matrices, we can then determine $(rI + G_3)$ and $(rI + G_4)$. Let us consider the selection of the matrices G_3 and G_4 that gives the matrices $(rI + G_3)$ and $(rI + G_4)$ as

$$(rI + G_3) = \begin{bmatrix} G_3' & -I & & & & & \\ -I & G_3' & & & & 0 & \\ & & G_3' & -I & & & \\ & & -I & G_3' & & & \\ & & & & G_3' & -I & \\ & 0 & & & -I & G_3' & \\ & & & & & & G_3' \end{bmatrix}_{7^2 X 7^2},$$

$$(rI + G_4) = \begin{bmatrix} G_3' & & & & & & \\ & G_3' & -I & & & 0 & \\ & -I & G_3' & & & & \\ & & & G_3' & -I & & \\ & & & -I & G_3' & & \\ & 0 & & & & G_3' & -I \\ & & & & & -I & G_3' \end{bmatrix}_{7^2 X 7^2},$$

where

$$G_3' = \begin{bmatrix} \alpha & & & & & & \\ & \alpha & & & & 0 & \\ & & \alpha & & & & \\ & & & \alpha & & & \\ & & & & \alpha & & \\ & 0 & & & & \alpha & \\ & & & & & & \alpha \end{bmatrix}, \text{ for } \alpha = r + g.$$

Now,

$$C_2 = \prod_{i=1}^{4} (rI + G_i) - \omega r^3 A$$

$$= \begin{bmatrix} Q & P & R & & & & \\ P & Q & P & & & 0 & \\ & P & Q & P & R & & \\ & R & P & Q & P & & \\ & & & P & Q & P & R \\ & 0 & & R & P & Q & P \\ & & & & & P & Q \end{bmatrix}_{7^2 X 7^2},$$

where

$$P = \omega r^3 - G_1' G_2' G_3'$$

$$= \begin{bmatrix} x & w & -\alpha & & & & \\ w & x & w & & & 0 & \\ & w & x & w & -\alpha & & \\ & -\alpha & w & x & w & & \\ & & & w & x & w & -\alpha \\ & 0 & & -\alpha & w & x & w \\ & & & & & w & x \end{bmatrix},$$

$$Q = G_1' G_2' G_3'^2 - \omega r^3 A$$

$$= \begin{bmatrix} y & x & w & & & & \\ x & y & x & & & 0 & \\ & x & y & x & w & & \\ & w & x & y & x & & \\ & & & x & y & x & w \\ & 0 & & w & x & y & x \\ & & & & & x & y \end{bmatrix},$$

and $R = G_1' G_2' =$

$$\begin{bmatrix} w & -\alpha & 1 & & & & \\ -\alpha & w & -\alpha & & & 0 & \\ & -\alpha & w & -\alpha & 1 & & \\ & 1 & -\alpha & w & -\alpha & & \\ & & & -\alpha & w & -\alpha & 1 \\ & 0 & & 1 & -\alpha & w & -\alpha \\ & & & & & -\alpha & w \end{bmatrix},$$

with $w = \alpha^2$, $x = \omega r^3 - \alpha^3$ and $y = \alpha^4 - 4g\omega r^3$.

We now write the algorithm for the AGE-GT-2 scheme in detail. In this algorithm, the number of points, N is assumed odd.

Algorithm 6.1.4-1: The AGE-GT-2 scheme.

Set $u_{i,j}^{(k)} = 0$, $i, j = 0, \ldots, N+1$, $\alpha = r + g$, $d = 1/(\alpha^2 - 1)$,
$w = \alpha^2$, $x = \omega r^3 - \alpha^3$, $y = \alpha^4 - 4g\omega r^3$ and $\alpha_1 = \alpha d$.

Step 1: To compute $\mathbf{u}^{(k+1/4)}$. Set $i, j = 1$.

while $j \le N-2$, compute

while $i \le N-2$, compute

$$\begin{aligned} r_1 = {} & wu_{i-1,j-1}^{(k)} + xu_{i,j-1}^{(k)} + wu_{i+1,j-1}^{(k)} - \alpha u_{i+2,j-1}^{(k)} \\ & + xu_{i-1,j}^{(k)} + yu_{i,j}^{(k)} + xu_{i+1,j}^{(k)} + wu_{i+2,j}^{(k)} + wu_{i-1,j+1}^{(k)} \\ & + xu_{i,j+1}^{(k)} + wu_{i+1,j+1}^{(k)} - \alpha u_{i+2,j+1}^{(k)} - \alpha u_{i-1,j+2}^{(k)} \\ & + wu_{i,j+2}^{(k)} - \alpha u_{i+1,j+2}^{(k)} + u_{i+2,j+2}^{(k)} + \omega r^3 b_{i,j} \end{aligned}$$

$$\begin{aligned} r_2 = {} & -\alpha u_{i-1,j-1}^{(k)} + wu_{i,j-1}^{(k)} + xu_{i+1,j-1}^{(k)} + wu_{i+2,j-1}^{(k)} \\ & + wu_{i-1,j}^{(k)} + xu_{i,j}^{(k)} + yu_{i+1,j}^{(k)} + xu_{i+2,j}^{(k)} - \alpha u_{i-1,j+1}^{(k)} \\ & + wu_{i,j+1}^{(k)} + xu_{i+1,j+1}^{(k)} + wu_{i+2,j+1}^{(k)} + u_{i-1,j+2}^{(k)} \\ & - \alpha u_{i,j+2}^{(k)} + wu_{i+1,j+2}^{(k)} - \alpha u_{i+2,j+2}^{(k)} + \omega r^3 b_{i+1,j} \end{aligned}$$

$$u_{i,j}^{(k+1/4)} = \alpha_1 r_1 + r_2 d, \quad u_{i+1,j}^{(k+1/4)} = r_1 d + \alpha_1 r_2$$

$i = i + 2$

$$u_{N,j}^{(k+1/4)} = (wu_{N-1,j-1}^{(k)} + xu_{N,j-1}^{(k)} + xu_{N-1,j}^{(k)} + yu_{N,j}^{(k)}$$
$$+ wu_{N-1,j+1}^{(k)} + xu_{N,j+1}^{(k)} - \alpha u_{N-1,j+2}^{(k)}$$
$$+ wu_{N,j+2}^{(k)} + \omega r^3 b_{N,j})/\alpha$$

$i = 1.$

while $i \le N-2$, compute

$$r_3 = -\alpha u_{i-1,j-1}^{(k)} + wu_{i,j-1}^{(k)} - \alpha u_{i+1,j-1}^{(k)} + u_{i+2,j-1}^{(k)}$$
$$+ wu_{i-1,j}^{(k)} + xu_{i,j}^{(k)} + wu_{i+1,j}^{(k)} - \alpha u_{i+2,j}^{(k)} + xu_{i-1,j+1}^{(k)}$$
$$+ yu_{i,j+1}^{(k)} + xu_{i+1,j+1}^{(k)} + wu_{i+2,j+1}^{(k)} + wu_{i-1,j+2}^{(k)}$$
$$+ xu_{i,j+2}^{(k)} + wu_{i+1,j+2}^{(k)} - \alpha u_{i+2,j+2}^{(k)} + \omega r^3 b_{i,j+1}$$

$$r_4 = u_{i-1,j-1}^{(k)} - \alpha u_{i,j-1}^{(k)} + wu_{i+1,j-1}^{(k)} - \alpha u_{i+2,j-1}^{(k)}$$
$$- \alpha u_{i-1,j}^{(k)} + wu_{i,j}^{(k)} + xu_{i+1,j}^{(k)} + wu_{i+2,j}^{(k)} + wu_{i-1,j+1}^{(k)}$$
$$+ xu_{i,j+1}^{(k)} + yu_{i+1,j+1}^{(k)} + xu_{i+2,j+1}^{(k)} - \alpha u_{i-1,j+2}^{(k)}$$
$$+ wu_{i,j+2}^{(k)} + xu_{i+1,j+2}^{(k)} + wu_{i+2,j+2}^{(k)} + \omega r^3 b_{i+1,j+1}$$

$$u_{i,j}^{(k+1/4)} = \alpha_1 r_3 + r_4 d, \quad u_{i+1,j}^{(k+1/4)} = r_3 d + \alpha_4 r_2$$

$i = i + 2$

$$u_{N,j}^{(k+1/4)} = (-\alpha u_{N-1,j-1}^{(k)} + wu_{N,j-1}^{(k)} + wu_{N-1,j}^{(k)} + xu_{N,j}^{(k)}$$
$$+ xu_{N-1,j+1}^{(k)} + yu_{N,j+1}^{(k)} + wu_{N-1,j+2}^{(k)}$$
$$+ xu_{N,j+2}^{(k)} + \omega r^3 b_{i,j+1})/\alpha$$

$j = j + 2$

$i = 1$

while $i \le N-2$, compute

$$r_5 = wu_{i-1,N-1}^{(k)} + xu_{i,N-1}^{(k)} + wu_{i+1,N-1}^{(k)} - \alpha u_{i+2,N-1}^{(k)} + xu_{i-1,N}^{(k)}$$
$$+ yu_{i,N}^{(k)} + xu_{i+1,N}^{(k)} + wu_{i+2,N}^{(k)} + \omega r^3 b_{i,N}$$

$$r_6 = -\alpha u^{(k)}_{i-1,N-1} + w u^{(k)}_{i,N-1} + x u^{(k)}_{i+1,N-1} + w u^{(k)}_{i+2,N-1} + w u^{(k)}_{i-1,N}$$
$$+ x u^{(k)}_{i,N} + y u^{(k)}_{i+1,N} + x u^{(k)}_{i+2,N} + \omega r^3 b_{i+1,N}$$

$$u^{(k+1/4)}_{i,N} = \alpha_1 r_5 + r_6 d, \quad u^{(k+1/4)}_{i+1,N} = r_5 d + \alpha_1 r_6$$

$i = i + 2$

$$u^{(k+1/4)}_{N,N} = (w u^{(k)}_{N-1,N-1} + x u^{(k)}_{N,N-1} + x u^{(k)}_{N-1,N} + y u^{(k)}_{N,N} + \omega r^3 b_{N,N})/\alpha$$

Step 2: To compute $u^{(k+1/2)}$. Set $i = 2$, $j = 1$.

while $j \le N$, compute

$$u^{(k+1/2)}_{1,j} = u^{(k+1/4)}_{1,j}/\alpha$$

while $i \le N-1$, compute

$$u^{(k+1/2)}_{i,j} = \alpha_1 u^{(k+1/4)}_{i,j} + d u^{(k+1/4)}_{i+1,j},$$

$$u^{(k+1/2)}_{i+1,j} = d u^{(k+1/4)}_{i,j} + \alpha_1 u^{(k+1/4)}_{i+1,j}$$

$i = i + 2$

$j = j + 1$

Step 3: To compute $u^{(k+3/4)}$. Set i, $j = 1$. (Change in direction).

while $i \le N$, compute

while $j \le N-2$, compute

$$u^{(k+3/4)}_{i,j} = \alpha_1 u^{(k+1/2)}_{i,j} + d u^{(k+1/2)}_{i,j+1}$$

$$u^{(k+3/4)}_{i,j+1} = d u^{(k+1/2)}_{i,j} + \alpha_1 u^{(k+1/2)}_{i,j+1}$$

$j = j + 2$

$$u^{(k+3/4)}_{i,N} = u^{(k+1/2)}_{i,N}/\alpha$$

$i = i + 1$

Step 4: To compute $u^{(k+1)}$. Set $i = 1$, $j = 2$. (change in direction).

while $i \le N$, compute

$$u^{(k+1)}_{i,1} = u^{(k+3/4)}_{i,1}/\alpha$$

while $j \le N-1$, compute

$$u_{i,j}^{(k+1)} = \alpha_1 u_{i,j}^{(k+3/4)} + du_{i,j+1}^{(k+3/4)}$$

$$u_{i,j+1}^{(k+1)} = du_{i,j}^{(k+3/4)} + \alpha_1 u_{i,j+1}^{(k+3/4)}$$

$j = j + 2$

$i = i + 1$

Step 5: Repeat *Step* 1 to *Step* 4 until convergence is achieved.

Based on the Algorithm 6.1.4-1, more computational work will be required to derive the matrix C_3 for the AGE-GT-3 scheme. Hence, it is just sufficient to show Algorithm 6.1.4-1. Moreover, Mitchell [1969] has stated that, the best splitting for general ω is in the form of the AGE-DG-2 and the AGE-DG-3 schemes. The details of the amount of work needed will be shown later. Concerning ω, Guittet [1967] has also considered the choice of the parameter ω and showed that the best value of ω is 2, for the Douglas scheme which is also applicable to the AGE-DG-2 and the AGE-DG-3 schemes.

6.1.5 The Successive Overrelaxation (SOR) method

For the purpose of comparison, we write the algorithm for the SOR method to solve the two and three dimensional model problems discussed in the previous sections.

A) For the two dimensional problem, equation (6.1.1-1).

Algorithm 6.1.5-1: The SOR method for equation (6.1.1-1).

Set $u_{i,j}^{(k)} = 0$, i, $j = 0, \ldots, N+1$.

Step 1: To compute $\mathbf{u}^{(k+1)}$.

for $i = 1$ to N, compute

for $j = 1$ to N, compute

$$u_{i,j}^{(k+1)} = \omega\,[(b_{i,j} + u_{i-1,j}^{(k+1)} + u_{i,j-1}^{(k+1)} + u_{i+1,j}^{(k)} + u_{i,j+1}^{(k)})/4g] + (1 - \omega)u_{i,j}^{(k)}.$$

Step 2: Repeat *Step* 1 until convergence is achieved.

B) For the three dimensional problem, equation (6.1.2-1).

Algorithm 6.1.5-2: The SOR method for equation (6.1.2-1).

Set $u_{i,j,k}^{(k)} = 0,\ i,\ j,\ k = 0, \ldots, N+1.$

Step 1: To compute $u^{(k+1)}$.

for $i = 1$ to N, compute

for $j = 1$ to N, compute

for $k = 1$ to N, compute

$$u_{i,j,k}^{(k+1)} = \omega\,[(b_{i,j,k} + u_{i-1,j,k}^{(k+1)} + u_{i,j-1,k}^{(k+1)} + u_{i,j,k-1}^{(k+1)} + u_{i+1,j,k}^{(k)} + u_{i,j+1,k}^{(k)} + u_{i,j,k+1}^{(k)})/6g] + (1 - \omega)u_{i,j,k}^{(k)}.$$

Step 2: Repeat *Step* 1 until convergence is achieved.

6.1.6 The computational complexity

In this section, we will be concerned with the estimation of the amount of arithmetic calculation needed in the algorithms derived in Sections 6.1.3 to 6.1.5.

In solving equation (6.1.1-1), i.e., the two dimensional problem, the amount of work per iteration is determined from the AGE-DG-2, AGE-GT-2 schemes and the SOR method, whilst the solution for the three dimensional problem, i.e., equation (6.1.2-1), the calculation is based on the AGE-DG-3 scheme and the SOR method.

Table 6.1.6-1 and 6.1.6-2 show the amount of operations for every iteration in solving equations (6.1.1-1) and (6.1.2-1) respectively.

Method	Multiplication	Addition	Overall
AGE-DG-2	$17(N-1)^2$	$15(N-1)^2$	$32(N-1)^2$
AGE-GT-2	$13(N-1)^2$	$20(N-1)^2$	$33(N-1)^2$
SOR	$3N^2$	$6N^2$	$9N^2$

Table 6.1.6-1: The amount of operations per iteration for solving equation (6.1.1-1)

Method	Multiplication	Addition	Overall
AGE-DG-3	$25(N-1)^3$	$23(N-1)^3$	$48(N-1)^3$
SOR	$3N^3$	$8N^3$	$11N^3$

Table 6.1.6-2: The amount of operations per iteration for solving equation (6.1.2-1)

Tables 6.1.6-1 and 6.1.6-2 show that the SOR method is far better than the AGE-DG and AGE-GT schemes in terms of the amount of computational work. However, with the limited application of the SOR method, i.e, the method requires with a single parameter, then the AGE-DG and AGE-GT schemes need further consideration. Since the computational work of the AGE-GT scheme is quite substantial, then we may consider further that the best method is the AGE-DG.

In the next section, the numerical results are presented to show the number of iteration needed for each method.

6.1.7 Experimental results

Numerical results presented here are for the AGE-DG-2, AGE-DG-3 and AGE-GT-2 schemes and also for the SOR method for solving both the two

and three dimensional problems. Five problems are considered with the first three concerned with the solution of the two dimensional problem.

Problems 4 and 5 are devoted to the solutions for problem in three dimensions. Problems 1 and 4 are the Helmholtz equation in two and three dimensions respectively. The results are presented for various values of ρ and σ. Problems 2 and 5 are the Laplace equation in two and three dimensions, whilst Problem 3 is a Poisson equation.

Problem 1 - The Helmholtz equation in two dimensions.

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} - \rho U = 6 - \rho(2x^2 + y^2), \qquad 0 \le x, y \le 1,$$

subject to the boundary conditions

$$U(x,0) = 2x^2, \quad U(x,1) = 2x^2 + 1, \qquad 0 \le x \le 1,$$

$$U(0,y) = y^2, \quad U(1,y) = 2 + y^2, \qquad 0 \le y \le 1.$$

The exact solution is given by $U(x,y) = 2x^2 + y^2$.

The results are tabulated as follows:

ρ = 0	SOR		AGE-DG-2 (ω = 2)	
N^2	ω	iter	r	iter
81	1.54	25	0.95-1.04	25
361	1.74	49	0.56	49
1521	1.86	92	0.33	108
6241	1.93	178	0.19	230

Table 6.1.7-1: Problem 1 with ρ = 0

ρ = 5	SOR		AGE-DG-2 (ω = 2)	
N^2	ω	iter	r	iter
81	1.49-1.50	24	1.04-1.11	22
361	1.71-1.72	46	0.59-0.61	44
1521	1.84	89	0.35	95
6241	1.92	171	0.20	205

Table 6.1.7-2: Problem 1 with ρ = 5

ρ = 20	SOR		AGE-DG-2 (ω = 2)	
N^2	ω	iter	r	iter
81	1.40-1.45	22	1.19-1.37	18
361	1.63	42	0.71-0.72	34
1521	1.79	83	0.41	73
6241	1.89	164	0.24	158

Table 6.1.7-3: Problem 1 with ρ = 20

ρ = 40	SOR		AGE-DG-2 (ω = 2)	
N^2	ω	iter	r	iter
81	1.31-1.34	20	1.38-1.63	15
361	1.55-1.56	40	0.79-0.84	28
1521	1.74-1.74	80	0.47	58
6241	1.85	155	0.28	127

Table 6.1.7-4: Problem 1 with ρ = 40

ρ = 70	SOR		AGE-DG-2 (ω = 2)	
N^2	ω	iter	r	iter
81	1.23-1.26	18	1.75-1.83	12
361	1.50	34	0.91-0.99	23
1521	1.70	66	0.54	46
6241	1.83	131	0.33	101

Table 6.1.7-5: Problem 1 with ρ = 70

ρ = 100	SOR		AGE-DG-2 (ω = 2)	
N^2	ω	iter	r	iter
81	1.21-1.23	16	1.89-2.20	11
361	1.46	30	1.01-1.12	20
1521	1.67	58	0.60-0.61	40
6241	1.81	114	0.35-0.36	87

Table 6.1.7-6: Problem 1 with ρ = 100

ρ = 200	SOR		AGE-DG-2 (ω = 2)	
N^2	ω	iter	r	iter
81	1.12-1.20	13	2.57-3.09	9
361	1.36-1.38	23	1.29-1.46	15
1521	1.59-1.60	44	0.74-0.77	29
6241	1.76	83	0.44	61

Table 6.1.7-7: Problem 1 with ρ = 200

Problem 2 - The Laplace equation in two dimensions

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = 0, \qquad 0 \le x, y \le 1,$$

subject to the boundary conditions

$$U(x,0) = U(x,1) = \sin \pi x, \qquad 0 \le x \le 1,$$

$$U(0,y) = U(1,y) = 0, \qquad 0 \le y \le 1.$$

The exact solution is given by

$$U(x,y) = \operatorname{sech} \frac{\pi}{2} \cosh \pi((y - \tfrac{1}{2}) \sin \pi x.$$

The results are tabulated in *Table* 6.1.7-8.

	SOR		AGE-DG-2 (ω = 2)	
N^2	ω	iter	r	iter
81	1.53-1.55	23	0.96-1.00	22
361	1.74	43	0.57-0.58	47
1521	1.86	85	0.34	102
6241	1.93	167	0.19	209

Table 6.1.7-8: Problem 2

Problem 3 - The Poisson equation.

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = -2, \qquad 0 \le x, y \le 1,$$

subject to the boundary conditions

$$U(x,0) = U(x,1) = x(1 - x), \qquad 0 \le x \le 1,$$

$$U(0,y) = 0, \; U(1,y) = \sinh \pi \sin \pi y, \qquad 0 \le y \le 1.$$

The exact solution is given by

$$U(x,y) = \sinh \pi x \sin \pi y + x(1 - x).$$

The results are tabulated in *Table* 6.1.7-9.

	SOR		AGE-DG-2 (ω = 2)	
N^2	ω	iter	r	iter
81	1.54	26	0.97-1.08	27
361	1.74	52	0.58-0.59	56
1521	1.86	94	0.34	116
6241	1.93	191	0.18	228

Table 6.1.7-9: Problem 3

Problem 4 - The three dimensional Helmholtz problem

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} + \frac{\partial^2 U}{\partial z^2} - \sigma U = (3-\sigma) \cosh x \cosh y \cosh z, \quad 0 \le x,y,z \le 1,$$

subject to the boundary conditions

$$U(x,y,0) = \cosh x \cosh y, \qquad 0 \le x,y \le 1,$$

$$U(x,y,1) = \cosh 1 \cosh x \cosh y, \qquad 0 \le x,y \le 1,$$

$$U(x,0,z) = \cosh x \cosh z, \qquad 0 \le x,z \le 1,$$

$$U(x,1,z) = \cosh 1 \cosh x \cosh z, \qquad 0 \le x,z \le 1,$$

$$U(0,y,z) = \cosh y \cosh z, \qquad 0 \le y,z \le 1,$$

$$U(1,y,z) = \cosh 1 \cosh y \cosh z, \qquad 0 \le y,z \le 1.$$

The exact solution is given by $U(x,y,z) = \cosh x \cosh y \cosh z$.

The results are tabulated as follows.

σ = 0	SOR		AGE-DG-3 (ω = 2)	
N^3	ω	iter	r	iter
729	1.51	30	1.53-1.53	26
1331	1.57-1.59	37	1.32-1.37	33
2197	1.62	43	1.21	39
3375	1.65	49	1.11	46
4913	1.68-1.70	56	1.03-1.04	54

Table 6.1.7-10: Problem 4 with σ = 0

σ = 5	SOR		AGE-DG-3 (ω = 2)	
N^3	ω	iter	r	iter
729	1.41-1.51	30	1.60-1.62	24
1331	1.53-1.56	36	1.40-1.43	30
2197	1.58-1.60	42	1.26-1.28	36
3375	1.62-1.63	48	1.15-1.19	43
4913	1.66	54	1.08	49

Table 6.1.7-11: Problem 4 with σ = 5

σ = 25	SOR		AGE-DG-3 (ω = 2)	
N^3	ω	iter	r	iter
729	1.40-1.41	24	1.88-1.92	19
1331	1.47-1.48	29	1.65	23
2197	1.52-1.54	34	1.46-1.49	28
3375	1.57	38	1.33-1.37	33
4913	1.61	43	1.23-1.26	38

Table 6.1.7-12: Problem 4 with σ = 25

σ = 50	SOR		AGE-DG-3 (ω = 2)	
N^3	ω	iter	r	iter
729	1.36-1.38	22	2.04-2.12	17
1331	1.44	26	1.75-1.87	21
2197	1.49-1.50	30	1.57-1.66	25
3375	1.54-1.55	34	1.43-1.50	29
4913	1.58-1.59	39	1.33-1.37	33

Table 6.1.7-13: Problem 4 with σ = 50

σ = 100	SOR		AGE-DG-3 (ω = 2)	
N^3	ω	iter	r	iter
729	1.30-1.31	17	2.61-2.80	13
1331	1.37	20	2.28-2.31	15
2197	1.40-1.43	24	1.97-2.09	18
3375	1.46-1.47	27	1.76-1.90	21
4913	1.50	30	1.66-1.67	23

Table 6.1.7-14: Problem 4 with σ = 100

σ = 200	SOR		AGE-DG-3 (ω = 2)	
N^3	ω	iter	r	iter
729	1.23	13	3.29-3.92	11
1331	1.27-1.31	16	2.88-3.07	12
2197	1.33-1.34	18	2.46-2.72	14
3375	1.36-1.39	21	2.18-2.44	16
4913	1.41-1.42	23	2.06-2.07	17

Table 6.1.7-15: Problem 4 with σ = 200

Problem 5 - The Laplace equation in three dimensions

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} + \frac{\partial^2 U}{\partial z^2} = 0, \qquad 0 \le x,y,z \le 1,$$

subject to the boundary conditions

$$U(x,y,0) = U(x,y,1) = 0, \qquad 0 \le x,y \le 1,$$

$$U(0,y,z) = U(1,y,z) = 0, \qquad 0 \le y,z \le 1,$$

$$U(x,0,z) = 0,\ U(x,1,z) = \sin \pi x \sin \pi z, \qquad 0 \le x,z \le 1.$$

The exact solution is given by

$$U(x,y,z) = \operatorname{sech} \frac{\pi}{\sqrt{2}} \sin \pi x \cosh [\sqrt{2}\pi(y - \tfrac{1}{2}) \sin \pi z.$$

The results are tabulated in *Table* 6.1.7-16.

	SOR		AGE-DG-3 (ω = 2)	
N^3	ω	iter	r	iter
729	1.50 - 1.53	27	1.42	21
2197	1.62	36	1.08 - 1.10	31
4913	1.69	46	0.88 - 0.89	40
9261	1.74	56	0.74 - 0.75	49

Table 6.1.7-16: Problem 5

The results presented in this section are purely experimental, as no theory exists for the optimal single parameter for solving the two and three dimensional problems. The theoretical background given for the prescribed method is only concerned with the convergence of the method. However, as shown for the one dimensional problems, the AGE

method again appears to be competitive to the SOR method in the number of iterations when solving two and three dimensional problems governed by Dirichlet boundary conditions especially when the matrix A is strongly diagonally dominant.

Evidently, the bounds of the eigenvalues for the problems given in this section can be determined from the (2x2) block submatrices, i.e., $a = g - 1$ and $b = g + 1$, where $g = 1 + \frac{1}{4}\rho h^2$ for the two dimensional problems and $g = 1 + \frac{1}{6}\sigma h^2$ for the three dimensional problems, where ρ and σ are constants.

For the one dimensional problems, with $g = 1 + \frac{1}{2}\rho h^2$, where ρ is a constant, the relation $r = \sqrt{ab}$ is satisfied for large ρ as the AGE-PR(1) scheme is theoretically derived from the ADI-PR method. It is then, shown that the AGE-DG scheme gives a similar convergence as the AGE-PR(1) scheme. Thus, in order to determine the theoretical optimal r for the two and three dimensional problems we notice that $a \longrightarrow 0$, hence as a guide to the experimental value of r, we should look closely to the pattern of r in the one dimensional problem when $\rho = 0$.

For a symmetric matrix, when $\rho = 0$, the value of $r \in [0,1]$, i.e., say, $r_1 \approx 0.5$. The results for the two and three dimensional problems clearly shows that when $\sigma = 0$, we obtain, $r_2 \approx 1$, and $r_3 \approx 1.5$ respectively. This assumption is based on $N = 9$ as a starting number of points. The values of r_2 and r_3 may be interpreted in many ways. The simplest is by considering the splitting of the matrices and the repetition of the same matrices used for solving the problems. Thus, this gives the value $r_2 = 2r_1$ and $r_3 = 3r_1$. The value of r for the case of larger numbers of points can then be considered to lie in the interval $[0, r_2]$ or $[0, r_3]$ as shown in *Tables* 6.1.7-1 to 6.17-16.

For a larger σ, i.e., $\sigma > 25$ and a smaller number of points, the results show that $r > b$. But, for a larger number of points, the values of r again appears to fall in the interval $[0, r_2]$ or $[0, r_3]$.

Although the the AGE-DG-2 and AGE-DG-3 schemes needs more computational work, we see from the simplicity of the schemes and that its parameters are easily determined, then these schemes may well be considered competitive to solve the two and three dimensional problems.

6.2 The solution with different boundary conditions

We have shown in Section 3.2 that the AGE method has been successfully applied to solve the two-point boundary-value problem with different boundary conditions. Now, with the AGE-DG-2 scheme discussed in the previous section, we will investigate the solution for the problem in two dimensions governed by periodic and Neumann boundary conditions. In this section, we will consider a general problem, i.e., the Helmholtz equation in two dimensions.

6.2.1 Periodic boundary conditions

Consider the Helmholtz equation in two dimensions

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} - \rho U = f(x,y), \qquad 0 \le x,y \le 1, \tag{6.2.1-1}$$

in a square region $0 \le x, y \le 1$, governed by the periodic boundary conditions

$$U(x,0) = U(x,1),\ U_y(x,0) = U_y(x,1), \qquad 0 \le x \le 1, \tag{6.2.1-2}$$

$$U(0,y) = U(1,y),\ U_x(0,y) = U_x(1,y) \qquad 0 \le y \le 1. \tag{6.2.1-3}$$

The periodic boundary conditions (6.2.1-2) and (6.2.1-3) are treated in a similar fashion as in Section 3.2.1.

Now, applying the centred finite difference approximations, for a small mesh size h, yields the conventional five-point formula

$$- u_{i,j-1} - u_{i-1,j} + 4gu_{i,j} - u_{i+1,j} - u_{i,j+1} = h^2 f(x_i, y_j)$$

$$i,\ j,\ = 1,\ \ldots,\ N \qquad (6.2.1\text{-}4)$$

where $g = 1 + \frac{1}{4}\rho h^2$ and N is the number of grid points or nodes in the region. Let us consider $h = 1/N$ and if we take $N = 6$, then the grid points can be represented as in *Figure* 6.2.1-1.

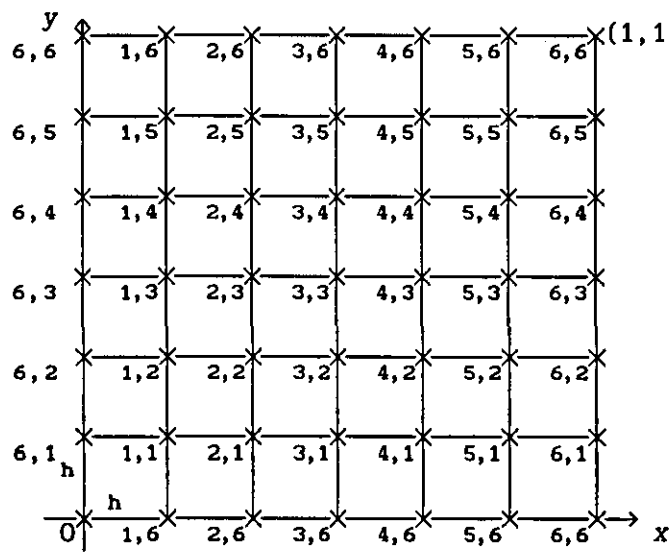


Figure 6.2.1-1: The number of nodes for $h = 1/6$.

In general, for an even N, equation (6.2.1-4) can be written in a matrix form $A\mathbf{u} = \mathbf{b}$, giving

$$A = \begin{bmatrix} A_1 & -I & & & -I \\ -I & A_1 & -I & 0 & \\ & \ddots & \ddots & \ddots & \\ & 0 & -I & A_1 & -I \\ -I & & & -I & A_1 \end{bmatrix}_{N^2 X N^2} \quad \text{with } A_1 = \begin{bmatrix} 4g & -1 & & & -1 \\ -1 & 4g & -1 & 0 & \\ & \ddots & \ddots & \ddots & \\ & 0 & -1 & 4g & -1 \\ -1 & & & -1 & 4g \end{bmatrix}.$$

(6.2.1-5)

The vector $\mathbf{u}$ and $\mathbf{b}$ are given by

$$\mathbf{u} = [u_{1,1},\ \ldots,\ u_{N,1};\ u_{1,2},\ \ldots,\ u_{N,2};\ \ldots;\ u_{1,N},\ \ldots,\ u_{N,N}]^T$$

(6.2.1-6)

and $\mathbf{b} = [b_{1,1}, \ldots, b_{N,1}; b_{1,2}, \ldots, b_{N,2}; \ldots; b_{1,N}, \ldots, b_{N,N}]^T$

(6.2.1-7)

where the elements of vector $\mathbf{b}$ are

$$b_{i,j} = h^2 f(x_i, y_j), \quad i, j = 1, \ldots, N.$$

Now, consider the splitting of the matrix A into

$$A = G_1 + G_2 + G_3 + G_4 \tag{6.2.1-8}$$

where G_1, G_2, G_3 and G_4 are symmetric and positive definite submatrices.

Let us consider the matrices $(G_1 + G_2)$ and $(G_3 + G_4)$ as follows:

$$G_1 + G_2 = \begin{bmatrix} A_2 & & & 0 \\ & A_2 & & \\ & & \ddots & \\ 0 & & & A_2 \end{bmatrix}_{N^2 X N^2} \quad \text{where } A_2 = \begin{bmatrix} 2g & -1 & & -1 \\ -1 & 2g & -1 & 0 \\ & \ddots & \ddots & \ddots \\ 0 & & -1 & 2g \end{bmatrix},$$

and

$$G_3 + G_4 = \begin{bmatrix} A_3 & -I & & -I \\ -I & A_3 & -I & 0 \\ & \ddots & \ddots & \ddots \\ 0 & & -I & A_3 \end{bmatrix}_{N^2 X N^2} \quad \text{where } A_3 = \begin{bmatrix} 2g & & & 0 \\ & 2g & & \\ & & \ddots & \\ 0 & & & 2g \end{bmatrix}.$$

Let us consider further the matrices G_1 and G_2 for the case when N is even, i.e.,

$$G_1 = \begin{bmatrix} G_1' & & & 0 \\ & G_1' & & \\ & & \ddots & \\ 0 & & & G_1' \end{bmatrix}_{N^2 X N^2}, \quad G_2 = \begin{bmatrix} G_2' & & & 0 \\ & G_2' & & \\ & & \ddots & \\ 0 & & & G_2' \end{bmatrix}_{N^2 X N^2}$$

where

$$G_1' = \begin{bmatrix} g & & & & -1 \\ & g & -1 & & \\ & -1 & g & & \\ & & & g & -1 \\ & & & -1 & g \\ -1 & & & & g \end{bmatrix}, \quad G_2' = \begin{bmatrix} g & -1 & & & \\ -1 & g & & & \\ & & g & -1 & \\ & & -1 & g & \\ & & & & g \quad -1 \\ & & & & -1 \quad g \end{bmatrix}.$$

(6.2.1-9)

It can be shown that, after reordering the direction, i.e., from row to column, the matrices $(G_1 + G_2)$ becomes $(G_3 + G_4)$ and vice-versa.

Let $\alpha = r + g$, $r > 0$, where r is the iteration parameter. Thus, we have the matrices $\overline{G}_1 = (rI + G_1)$ and $\overline{G}_2 = (rI + G_2)$ which are given as

$$\overline{G}_1 = \begin{bmatrix} G_1'' & & & \\ & G_1'' & & 0 \\ & & \ddots & \\ 0 & & & G_1'' \end{bmatrix}_{N^2XN^2}, \quad \overline{G}_2 = \begin{bmatrix} G_2'' & & & \\ & G_2'' & & 0 \\ & & \ddots & \\ 0 & & & G_2'' \end{bmatrix}_{N^2XN^2}$$

where

$$G_1'' = \begin{bmatrix} \alpha & & & & -1 \\ & \alpha & -1 & & \\ & -1 & \alpha & & \\ & & & \alpha & -1 \\ & & & -1 & \alpha \\ -1 & & & & \alpha \end{bmatrix}, \quad G_2'' = \begin{bmatrix} \alpha & -1 & & & \\ -1 & \alpha & & & \\ & & \alpha & -1 & \\ & & -1 & \alpha & \\ & & & & \alpha \quad -1 \\ & & & & -1 \quad \alpha \end{bmatrix}$$

and invertible.

We now seek to analyse the convergence of the AGE-DG-2 scheme for the Poisson equation subject to periodic boundary conditions. It has been shown in Section 3.2 (for a problem subject to periodic boundary condition) that by a suitable matrix permutation P,

$$G_2'' = PG_1''P^T.$$

Thus, the eigenvalues of G_1'' and G_2'' are similar. Hence, the AGE-DG-2 scheme for solving the periodic problem is convergent.

It is obvious that, if one choose G_2' in the form of G_1' and vice-versa, the scheme is also convergent. However, by having G_1' and G_2' as given in (6.2.1-9), the algorithm is found to be much shorter.

By using the AGE-DG-2 scheme in Section 6.1.3, the algorithm can be presented as follows. First, to obtain the matrix

$$C = [(rI + G_1) - \omega A].$$

Suppose that $N = 6$. Then, the matrix C is given as

$$C = \begin{bmatrix} C_1 & \omega I & & & & \omega I \\ \omega I & C_1 & \omega I & & 0 & \\ & \omega I & C_1 & \omega I & & \\ & & \omega I & C_1 & \omega I & \\ & 0 & & \omega I & C_1 & \omega I \\ \omega I & & & & \omega I & C_1 \end{bmatrix}_{6^2 X 6^2},$$

with

$$C_1 = \begin{bmatrix} t & \omega & & & & s \\ \omega & t & s & & 0 & \\ & s & t & \omega & & \\ & & \omega & t & s & \\ & 0 & & s & t & \omega \\ s & & & & \omega & t \end{bmatrix},$$

where $t = \alpha - 4\omega$ and $s = \omega - 1$.

We now write the algorithm for a general N (even) in detail.

Algorithm 6.2.1-1: The AGE-DG-2 Scheme for equation (6.2.1-1).

Set $u_{i,j}^{(k)} = 0$, $i, j = 0, \ldots, N+1$, $\alpha = r + g$, $d = 1/(\alpha^2 - 1)$,

$$t = \alpha - 4g\omega, \quad s = \omega - 1, \quad \alpha_1 = \alpha d.$$

Step 1: To compute $\mathbf{u}^{(k+1/4)}$.

A. The elements $u_{1,1}^{(k+1/4)}$ to $u_{N,1}^{(k+1/4)}$. Set $i = 2$.

$$r_1 = tu_{1,1}^{(k)} + \omega u_{2,1}^{(k)} + su_{N,1}^{(k)} + \omega u_{1,2}^{(k)} + \omega u_{1,N}^{(k)} + \omega b_{1,1}$$

$$r_2 = su_{1,1}^{(k)} + \omega u_{N-1,1}^{(k)} + tu_{N,1}^{(k)} + \omega u_{N,2}^{(k)} + \omega u_{N,N}^{(k)} + \omega b_{N,1}$$

$$u_{1,1}^{(k+1/4)} = \alpha_1 r_1 + r_2 d, \quad u_{N,1}^{(k+1/4)} = r_1 d + \alpha_1 r_2$$

while $i \le N-2$, compute

$$r_1 = \omega u_{i-1,1}^{(k)} tu_{i,1}^{(k)} + su_{i+1,1}^{(k)} + \omega u_{i,2}^{(k)} + \omega u_{i,N}^{(k)} + \omega b_{i,1}$$

$$r_2 = su_{i,1}^{(k)} + tu_{i+1,1}^{(k)} + \omega u_{i+2,1}^{(k)} + \omega u_{i+1,2}^{(k)} + \omega u_{i+1,N}^{(k)} + \omega b_{i+1,1}$$

$$u_{i,1}^{(k+1/4)} = \alpha_1 r_1 + r_2 d, \quad u_{i+1,1}^{(k+1/4)} = r_1 d + \alpha_1 r_2$$

$i = i + 2$.

B. The Elements $u_{1,2}^{(k+1/4)}$ to $u_{N-1,N}^{(k+1/4)}$. Set $i, j = 2$.

while $j \le N-1$, compute

$$r_1 = \omega u_{1,j-1}^{(k)} + tu_{1,j}^{(k)} + \omega u_{2,j}^{(k)} + su_{N,j}^{(k)} + \omega u_{1,j+1}^{(k)} + \omega b_{1,j}$$

$$r_2 = \omega u_{N,j-1}^{(k)} + su_{1,j}^{(k)} + \omega u_{N-1,j}^{(k)} + tu_{N,j}^{(k)} + \omega u_{N,j+1}^{(k)} + \omega b_{N,j}$$

$$u_{1,j}^{(k+1/4)} = \alpha_1 r_1 + r_2 d, \quad u_{N,j}^{(k+1/4)} = r_1 d + \alpha_1 r_2$$

while $i \le N-2$, compute

$$r_1 = \omega u_{1,j-1}^{(k)} + \omega u_{i-1,j}^{(k)} + tu_{i,j}^{(k)} + su_{i+1,j}^{(k)} + \omega u_{i,j+1}^{(k)} + \omega b_{i,j}$$

$$r_2 = \omega u_{i+1,j-1}^{(k)} + su_{i,j}^{(k)} + tu_{i+1,j}^{(k)} + \omega u_{i+2,j}^{(k)} + \omega u_{i+1,j+1}^{(k)} + \omega b_{i+1,j}$$

$$u_{i,j}^{(k+1/4)} = \alpha_1 r_1 + r_2 d, \quad u_{i+1,j}^{(k+1/4)} = r_1 d + \alpha_1 r_2$$

$i = i + 2$

$j = j + 1.$

C. The elements $u_{1,N}^{(k+1/4)}$ to $u_{N,N}^{(k+1/4)}$. Set $i = 2$.

$$r_1 = \omega u_{1,1}^{(k)} + \omega u_{1,N-1}^{(k)} + t u_{1,N}^{(k)} + \omega u_{2,N}^{(k)} + s u_{N,N}^{(k)} + \omega b_{1,N}$$

$$r_2 = \omega u_{N,1}^{(k)} + \omega u_{N,N-1}^{(k)} + s u_{1,N}^{(k)} + \omega u_{N-1,N}^{(k)} + t u_{N,N}^{(k)} + \omega b_{N,N}$$

$$u_{1,N}^{(k+1/4)} = \alpha_1 r_1 + r_2 d, \quad u_{N,N}^{(k+1/4)} = r_1 d + \alpha_1 r_2$$

while $i \le N-2$, compute

$$r_1 = \omega u_{1,1}^{(k)} + \omega u_{1,N-1}^{(k)} + \omega u_{i-1,N}^{(k)} + t u_{i,N}^{(k)} + s u_{i+1,N}^{(k)} + \omega b_{i,N}$$

$$r_2 = \omega u_{i+1,1}^{(k)} + \omega u_{i+1,N-1}^{(k)} + s u_{i,N}^{(k)} + t u_{i+1,N}^{(k)} + \omega u_{i+2,N}^{(k)} + \omega b_{i+1,N}$$

$$u_{i,N}^{(k+1/4)} = \alpha_1 r_1 + r_2 d, \quad u_{i+1,N}^{(k+1/4)} = r_1 d + \alpha_1 r_2$$

$i = i + 2.$

Step 2: To compute $\mathbf{u}^{(k+1/2)}$. $i = 1$, $j = 1$.

while $j \le N$, compute

while $i \le N-1$, compute

$$r_1 = r u_{i,j}^{(k+1/4)} + g u_{i,j}^{(k)} - u_{i+1,j}^{(k)}$$

$$r_2 = r u_{i+1,j}^{(k+1/4)} - u_{i,j}^{(k)} + g u_{i+1,j}^{(k)}$$

$$u_{i,j}^{(k+1/2)} = \alpha_1 r_1 + r_2 d, \quad u_{i+1,j}^{(k+1/2)} = r_1 d + \alpha_1 r_2$$

$i = i + 2$

$j = j + 1.$

Step 3: To compute $\mathbf{u}^{(k+3/4)}$. $i = 1$, $j = 2$. (Change in direction).

while $i \le N$, compute

$$r_1 = r u_{1,1}^{(k+1/2)} + g u_{1,1}^{(k)} - u_{1,N}^{(k)}$$

$$r_2 = ru_{i,N}^{(k+1/2)} - u_{i,1}^{(k)} + gu_{i,N}^{(k)}$$

$$u_{i,1}^{(k+3/4)} = \alpha_1 r_1 + r_2 d, \quad u_{i,N}^{(k+3/4)} = r_1 d + \alpha_1 r_2$$

while $j \le N-2$, compute

$$r_1 = ru_{i,j}^{(k+1/2)} + gu_{i,j}^{(k)} - u_{i,j+1}^{(k)}$$

$$r_2 = ru_{i,j+1}^{(k+1/2)} - u_{i,j}^{(k)} + gu_{i,j+1}^{(k)}$$

$$u_{i,j}^{(k+3/4)} = \alpha_1 r_1 + r_2 d, \quad u_{i,j+1}^{(k+3/4)} = r_1 d + \alpha_1 r_2$$

$j = j + 2$

$i = i + 1$.

Step 4: To compute $\mathbf{u}^{(k+1)}$. $i = 1$, $j = 1$. (change in direction).

while $i \le N$, compute

while $j \le N-1$, compute

$$r_1 = ru_{i,j}^{(k+3/4)} + gu_{i,j}^{(k)} - u_{i,j+1}^{(k)}$$

$$r_2 = ru_{i,j+1}^{(k+3/4)} - u_{i,j}^{(k)} + gu_{i,j+1}^{(k)}$$

$$u_{i,j}^{(k+1)} = \alpha_1 r_1 + r_2 d, \quad u_{i,j+1}^{(k+1)} = r_1 d + \alpha_1 r_2$$

$j = j + 2$

$i = i + 1$.

Step 5: Repeat *Step* 1 to *Step* 4 until convergence is achieved.

6.2.2 Neumann boundary conditions

We now consider the Helmholtz equation in two dimensions

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} - \rho U = f(x,y), \qquad 0 \le x,y \le 1, \tag{6.2.2-1}$$

subject to the Neumann boundary conditions

$$U_y(x,0) = g_0(x), \; U_y(x,1) = g_1(x), \tag{6.2.2-2}$$

$$U_x(0,y) = h_0(y), \; U_x(1,y) = h_1(y) \tag{6.2.2-3}$$

on the boundary of a square region $0 \le x, y \le 1$.

Let h be the grid spacing and if $h = 1/N$, then for $N = 4$, the grid points can illustrated as in *Figure* 6.2.2-1.

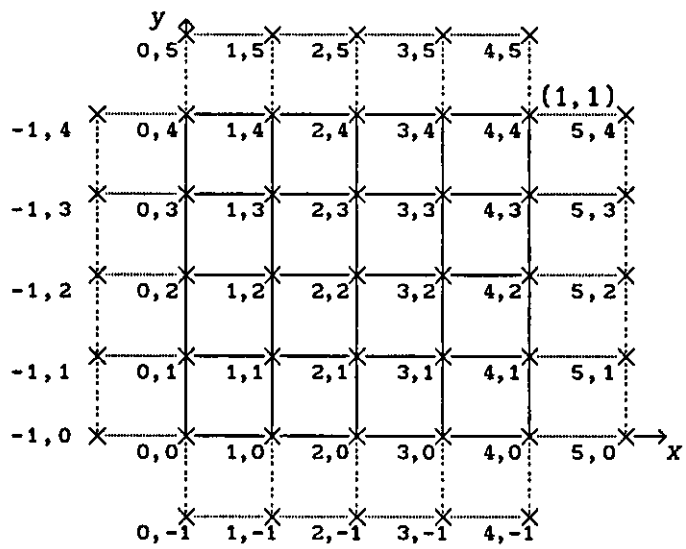


Figure 6.2.2-1: The number of nodes for $h = 1/4$.

Unlike the Dirichlet problem, we are now dealing with fictitious points, i.e., these are the points which lie just outside the region. These false boundaries are illustrated in *Figure* 6.2.2-1 as the points joining to the region by the dotted line. These points can be eliminated by using the centred approximation, for small h,

$$\left[\frac{\partial U}{\partial x}\right]_{i,j} = \frac{u_{i+1,j} - u_{i-1,j}}{2h} \quad \text{and} \quad \left[\frac{\partial U}{\partial y}\right]_{i,j} = \frac{u_{i,j+1} - u_{i,j-1}}{2h} \tag{6.2.2-5}$$

which gives

$$u_{i-1,j} = u_{i+1,j} - 2h\left[\frac{\partial U}{\partial x}\right]_{i,j} \quad \text{and} \quad u_{i,j-1} = u_{i,j+1} - 2h\left[\frac{\partial U}{\partial y}\right]_{i,j} \tag{6.2.2-6}$$

For example, $u_{-1,0} = u_{1,0} - 2h\left[\frac{\partial U}{\partial x}\right]_{0,0}$, etc.

Obviously, we have to solve for $(N+1)^2$ points including the points at the boundaries. By using the centred finite difference approximations, for small h, we have a conventional five-point formula

$$- u_{i,j-1} - u_{i-1,j} + 4gu_{i,j} - u_{i+1,j} - u_{i,j+1} = h^2 f(x_i, y_j)$$

$$i,\ j,\ = 1,\ \ldots,\ N \qquad (6.2.2\text{-}7)$$

where $g = 1 + \frac{1}{4}\rho h^2$, which gives the linear system $A\mathbf{u} = \mathbf{b}$. For our model problem, the linear system derived from equation (6.2.2-7) gives

$$A = \begin{bmatrix} A_1 & -2I & & & \\ -I & A_1 & -I & 0 & \\ & \ddots & \ddots & \ddots & \\ & 0 & -I & A_1 & -I \\ & & & -2I & A_1 \end{bmatrix}_{N^2 X N^2} \quad \text{with } A_1 = \begin{bmatrix} 4g & -2 & & & \\ -1 & 4g & -1 & 0 & \\ & \ddots & \ddots & \ddots & \\ & 0 & -1 & 4g & -1 \\ & & & -2 & 4g \end{bmatrix}.$$

(6.2.2-8)

The vector $\mathbf{u}$ and $\mathbf{b}$ are given by

$$\mathbf{u} = [u_{0,0},\ \ldots,\ u_{N,0};\ u_{0,1},\ \ldots,\ u_{N,1};\ \ldots;\ u_{0,N},\ \ldots,\ u_{N,N}]^T$$

(6.2.2-9)

$$\mathbf{b} = [b_{0,0},\ \ldots,\ b_{N,0};\ b_{0,1},\ \ldots,\ b_{N,1};\ \ldots;\ b_{0,N},\ \ldots,\ b_{N,N}]^T$$

(6.2.2-10)

where the elements of vector $\mathbf{b}$ are

$$b_{i,0} = h^2 f(x_i, y_0) - 2h\left[\frac{\partial U}{\partial y}\right]_{i,0},\quad b_{i,N} = h^2 f(x_i, y_N) + 2h\left[\frac{\partial U}{\partial y}\right]_{i,N},$$

$$i = 1,\ \ldots,\ N.$$

$$b_{0,j} = h^2 f(x_0, y_j) - 2h\left[\frac{\partial U}{\partial x}\right]_{0,j},\quad b_{N,j} = h^2 f(x_N, y_j) + 2h\left[\frac{\partial U}{\partial x}\right]_{N,j},$$

$$j = 1,\ \ldots,\ N,$$

$$b_{0,0} = h^2 f(x_0, y_0) - 2h\left\{\left[\frac{\partial U}{\partial x}\right]_{0,0} + \left[\frac{\partial U}{\partial y}\right]_{0,0}\right\}$$

$$b_{N,0} = h^2 f(x_N, y_0) + 2h\left\{\left[\frac{\partial U}{\partial x}\right]_{N,0} - \left[\frac{\partial U}{\partial y}\right]_{N,0}\right\}$$

$$b_{0,N} = h^2 f(x_0, y_N) + 2h\left\{\left[\frac{\partial U}{\partial y}\right]_{0,N} - \left[\frac{\partial U}{\partial x}\right]_{0,N}\right\}$$

and $$b_{N,N} = h^2 f(x_N, y_N) + 2h\left\{\left[\frac{\partial U}{\partial x}\right]_{N,N} + \left[\frac{\partial U}{\partial y}\right]_{N,N}\right\}.$$

Evidently, for $\rho = 0$, the matrix A is singular since one of the eigenvalues is zero. Thus, we will only consider $\rho > 0$.

Now, consider the splitting of the matrix A into

$$A = G_1 + G_2 + G_3 + G_4 \tag{6.2.2-11}$$

where the matrices G_1, G_2, G_3 and G_4 have non-negative eigenvalues.

We now seek to analyse the convergence of the AGE-DG-2 scheme for the Helmholtz equation in two dimensions subject to Neumann boundary conditions.

Let us consider the matrices $(G_1 + G_2)$ and $(G_3 + G_4)$ as

$$G_1 + G_2 = \begin{bmatrix} G_1' & & & \\ & G_1' & & 0 \\ & & \ddots & \\ 0 & & & G_1' \end{bmatrix}_{N^2 X N^2} \quad \text{where } G_1' = \begin{bmatrix} 2g & -2 & & \\ -1 & 2g & -1 & 0 \\ & \ddots & \ddots & \ddots \\ 0 & & -1 & 2g & -1 \\ & & & -2 & 2g \end{bmatrix}, \tag{6.2.2-12}$$

$$G_3 + G_4 = \begin{bmatrix} G_3' & -2I & & \\ -I & G_3' & -I & 0 \\ & \ddots & \ddots & \ddots \\ 0 & & -I & G_3' & -I \\ & & & -2I & G_3' \end{bmatrix}_{N^2 X N^2} \quad \text{where } G_3' = \begin{bmatrix} 2g & & & \\ & 2g & & 0 \\ & & \ddots & \\ 0 & & & 2g \end{bmatrix}. \tag{6.2.2-13}$$

Now, let us consider further for the case when N is odd, the matrices

$$G_1 = \begin{bmatrix} G_1'' & & & \\ & G_1'' & & 0 \\ & & \ddots & \\ 0 & & & G_1'' \end{bmatrix}_{N^2 X N^2} \quad \text{where } G_1'' = \begin{bmatrix} \tilde{G} & & & \\ & \hat{G} & & \\ & & \ddots & \\ & & & \hat{G} \\ & & & & \overline{G} \end{bmatrix}, \tag{6.2.2-14}$$

$$G_2 = \begin{bmatrix} G_2'' & & & \\ & G_2'' & 0 & \\ & & \ddots & \\ & 0 & & G_2'' \end{bmatrix}_{N^2 X N^2} \quad \text{where } G_2'' = \begin{bmatrix} 0 & & \\ & \hat{G} & \\ & & \ddots \\ & & & \hat{G} \\ & & & 0 \end{bmatrix}$$

(6.2.2-15)

and where $\hat{G} = \begin{bmatrix} g & -1 \\ -1 & g \end{bmatrix}$, $\tilde{G} = \begin{bmatrix} 2g & -2 \\ -1 & g \end{bmatrix}$ and $\bar{G} = \begin{bmatrix} g & -1 \\ -2 & 2g \end{bmatrix}$.

The matrix $(G_3 + G_4)$ can have the form of the matrix $(G_1 + G_2)$ by interchanging from row to column, and vice-versa.

From (6.2.2-14), the eigenvalues of the matrix G_1 are given as

$$g - 1,\ g + 1,\ \frac{3}{2}g - \frac{1}{2}\sqrt{g^2 + 8},\ \text{and}\ \frac{3}{2}g + \frac{1}{2}\sqrt{g^2 + 8}$$

and from (6.2.2-15), we have the eigenvalues of the matrix G_2 as

$$0,\ g - 1,\ \text{and}\ g + 1.$$

It is clear that the matrix G_2 is singular. However, for any iteration parameter, $r > 0$, the matrix $(rI + G_2)$ is non-singular, thus its inverse, $(rI + G_2)^{-1}$ exists. The inverse of the matrix $(rI + G_1)$, i.e., $(rI + G_1)^{-1}$ also exists. As been shown in Section 3.2.2 for the one dimensional problem, the proof for the convergence is similar. Thus, for any $\rho > 0$, the AGE-DG-2 scheme for solving the Neumann problem is convergent.

By using the AGE-DG-2 scheme, we now present the algorithm for the model problem governed by the Neumann boundary conditions.

Let us consider the case when $N = 5$. Then, the solution vector u are $u_{0,0}, \ldots, u_{5,0}; \ldots; u_{0,5}, \ldots, u_{5,5}$.

First, to obtain the matrix $C = [(rI + G_1) - \omega A]$.

For $N = 5$ the matrix C is given as

$$C = \begin{bmatrix} C_1 & s_2 I & & & & & \\ \omega I & C_1 & \omega I & & 0 & & \\ & \omega I & C_1 & \omega I & & & \\ & & \omega I & C_1 & \omega I & & \\ & & & \omega I & C_1 & \omega I & \\ & 0 & & & s_2 I & C_1 & \end{bmatrix}_{6^2 X 6^2},$$

$$\text{with } C_1 = \begin{bmatrix} t_1 & s_1 & & & & & \\ s & t & \omega & & 0 & & \\ & \omega & t & s & & & \\ & & s & t & \omega & & \\ & & & \omega & t & s & \\ & 0 & & & s_1 & t_1 & \end{bmatrix},$$

where $t_1 = \alpha_1 - 4g\omega$, $t = \alpha - 4g\omega$, $s = \omega - 1$, $s_1 = 2s$, and $s_2 = 2\omega$, with $\alpha = r + g$, and $\alpha_1 = \alpha + 2g$. The matrices G_1 and G_2 are as in (6.2.2-14) and (6.2.2-15) respectively. For any $r > 0$,

$$(rI + \hat{G})^{-1} = d\begin{bmatrix} \alpha & 1 \\ 1 & \alpha \end{bmatrix}, \quad (rI + \tilde{G})^{-1} = d_1\begin{bmatrix} \alpha & 2 \\ 1 & \alpha_1 \end{bmatrix}$$

and $(rI + \bar{G})^{-1} = d_1\begin{bmatrix} \alpha_1 & 1 \\ 2 & \alpha \end{bmatrix}$, where $d = 1/(\alpha^2 - 1)$, $d_1 = 1/(\alpha\alpha_1 - 2)$.

We now write the algorithm for a general N (odd) in detail.

Algorithm 6.2.2-1: The AGE-DG-2 scheme for equation 6.2.2-1.

Set $u_{i,j}^{(k)} = 0$, $i, j = 0, \ldots, N$, $g = 1 + \frac{1}{4}\rho h^2$, $\alpha = r + g$, $\alpha_1 = r + 2g$, $d = 1/(\alpha^2 - 1)$, $d_1 = 1/(\alpha\alpha_1 - 2)$, $t = \alpha - 4g\omega$, $t_1 = \alpha_1 - 4g\omega$, $s_2 = 2\omega$, $s = \omega - 1$ and $s_1 = 2s$.

Step 1: To compute $\mathbf{u}^{(k+1/4)}$.

A. The elements $u_{0,0}^{(k+1/4)}$ to $u_{N,0}^{(k+1/4)}$. Set $i = 2$.

$$r_1 = t_1 u_{0,0}^{(k)} + s_1 u_{1,0}^{(k)} + s_2 u_{0,1}^{(k)} + \omega b_{0,0}$$

$$r_2 = s u_{0,0}^{(k)} + t u_{1,0}^{(k)} + \omega u_{2,0}^{(k)} + s_2 u_{1,1}^{(k)} + \omega b_{1,0}$$

$$u_{0,0}^{(k+1/4)} = (\alpha r_1 + 2r_2)d_1, \quad u_{1,0}^{(k+1/4)} = (r_1 + \alpha_1 r_2)d_1$$

while $i \le N-3$, compute

$$r_1 = \omega u_{i-1,0}^{(k)} + t u_{i,0}^{(k)} + s u_{i+1,0}^{(k)} + s_2 u_{i,1}^{(k)} + \omega b_{i,0}$$

$$r_2 = s u_{i,0}^{(k)} + t u_{i+1,0}^{(k)} + \omega u_{i+2,0}^{(k)} + s_2 u_{i+1,1}^{(k)} + \omega b_{i+1,0}$$

$$u_{i,0}^{(k+1/4)} = (\alpha r_1 + r_2)d, \quad u_{i+1,0}^{(k+1/4)} = (r_1 + \alpha r_2)d$$

$i = i + 2$.

$$r_1 = \omega u_{N-2,0}^{(k)} + t u_{N-1,0}^{(k)} + s u_{N,0}^{(k)} + s_2 u_{N-1,1}^{(k)} + \omega b_{N-1,0}$$

$$r_2 = s_1 u_{N-1,0}^{(k)} + t_1 u_{N,0}^{(k)} + s_2 u_{N,1}^{(k)} + \omega b_{N,0}$$

$$u_{N-1,0}^{(k+1/4)} = (\alpha_1 r_1 + r_2)d_1, \quad u_{N,0}^{(k+1/4)} = (2r_1 + \alpha r_2)d_1.$$

B. The elements $u_{0,1}^{(k+1/4)}$ to $u_{N,N-1}^{(k+1/4)}$. Set $i = 2$, $j = 1$.

while $j \le N-1$, compute

$$r_1 = \omega u_{0,j-1}^{(k)} + t_1 u_{0,j}^{(k)} + s_1 u_{1,j}^{(k)} + \omega u_{0,j+1}^{(k)} + \omega b_{0,j}$$

$$r_2 = \omega u_{1,j-1}^{(k)} + s u_{0,j}^{(k)} + t u_{1,j}^{(k)} + \omega u_{2,j}^{(k)} + \omega u_{1,j+1}^{(k)} + \omega b_{1,j}$$

$$u_{0,j}^{(k+1/4)} = (\alpha r_1 + 2r_2)d_1, \quad u_{1,j}^{(k+1/4)} = (r_1 + \alpha_1 r_2)d_1$$

while $i \le N-3$, compute

$$r_1 = \omega u_{i,j-1}^{(k)} + \omega u_{i-1,j}^{(k)} + t u_{i,j}^{(k)} + s u_{i+1,j}^{(k)} + \omega u_{i,j+1}^{(k)} + \omega b_{i,j}$$

$$r_2 = \omega u_{i+1,j-1}^{(k)} + s u_{i,j}^{(k)} + t u_{i+1,j}^{(k)} + \omega u_{i+2,j}^{(k)} + \omega u_{i+1,j+1}^{(k)} + \omega b_{i+1,j}$$

$$u_{i,j}^{(k+1/4)} = (\alpha r_1 + r_2)d, \quad u_{i+1,j}^{(k+1/4)} = (r_1 + \alpha r_2)d$$

$i = i + 2$

$$r_1 = \omega u^{(k)}_{N-1,j-1} + \omega u^{(k)}_{N-2,j} + t u^{(k)}_{N-1,j} + s u^{(k)}_{N,j} + \omega u^{(k)}_{N-1,j+1} + \omega b_{N-1,j}$$

$$r_2 = \omega u^{(k)}_{N,j-1} + s_1 u^{(k)}_{N-1,j} + t_1 u^{(k)}_{N,j} + \omega u^{(k)}_{N,j+1} + \omega b_{N,j}$$

$$u^{(k+1/4)}_{N-1,j} = (\alpha_1 r_1 + r_2)d_1, \quad u^{(k+1/4)}_{N,j} = (2r_1 + \alpha r_2)d_1$$

$j = j + 1$.

C. The elements $u^{(k+1/4)}_{0,N}$ to $u^{(k+1/4)}_{N,N}$. Set $i = 2$.

$$r_1 = s_2 u^{(k)}_{0,N-1} + t_1 u^{(k)}_{0,N} + s_1 u^{(k)}_{1,N} + \omega b_{0,N}$$

$$r_2 = s_2 u^{(k)}_{1,N-1} + s u^{(k)}_{0,N} + t u^{(k)}_{1,N} + \omega u^{(k)}_{2,N} + \omega b_{1,N}$$

$$u^{(k+1/4)}_{0,N} = (\alpha r_1 + 2r_2)d_1, \quad u^{(k+1/4)}_{1,N} = (r_1 + \alpha_1 r_2)d_1$$

while $i \le N-3$, compute

$$r_1 = s_2 u^{(k)}_{i,N-1} + \omega u^{(k)}_{i-1,N} + t u^{(k)}_{i,N} + s u^{(k)}_{i+1,N} + \omega b_{i,N}$$

$$r_2 = s_2 u^{(k)}_{i+1,N-1} + s u^{(k)}_{i,N} + t u^{(k)}_{i+1,N} + \omega u^{(k)}_{i+2,N} + \omega b_{i+1,N}$$

$$u^{(k+1/4)}_{i,N} = (\alpha r_1 + r_2)d, \quad u^{(k+1/4)}_{i+1,N} = (r_1 + \alpha r_2)d$$

$i = i + 2$

$$r_1 = s_2 u^{(k)}_{N-1,N-1} + \omega u^{(k)}_{N-2,N} + t u^{(k)}_{N-1,N} + s u^{(k)}_{N,N} + \omega b_{N-1,N}$$

$$r_2 = s_2 u^{(k)}_{N,N-1} + s_1 u^{(k)}_{N-1,N} + t_1 u^{(k)}_{N,N} + \omega b_{N,N}$$

$$u^{(k+1/4)}_{N-1,N} = (\alpha_1 r_1 + r_2)d_1, \quad u^{(k+1/4)}_{N,N} = (2r_1 + \alpha r_2)d_1.$$

Step 2: To compute $u^{(k+1/2)}$. $i = 1$, $j = 0$.

while $j \le N$, compute

$$u^{(k+1/2)}_{0,j} = u^{(k+1/4)}_{0,j}$$

while $i \le N-2$, compute

$$r_1 = r u^{(k+1/4)}_{i,j} + g u^{(k)}_{i,j} - u^{(k)}_{i+1,j}$$

$$r_2 = ru_{i+1,j}^{(k+1/4)} - u_{i,j}^{(k)} + gu_{i+1,j}^{(k)}$$

$$u_{i,j}^{(k+1/2)} = (\alpha r_1 + r_2)d, \quad u_{i+1,j}^{(k+1/2)} = (r_1 + \alpha r_2)d$$

$i = i + 2$

$$u_{N,j}^{(k+1/2)} = u_{N,j}^{(k+1/4)}$$

$j = j + 1$.

Step 3: To compute $\mathbf{u}^{(k+3/4)}$. $i = 0$, $j = 2$. (Change in direction).

while $i \leq N$, compute

$$r_1 = ru_{i,0}^{(k+1/2)} + 2gu_{i,0}^{(k)} - 2u_{i,1}^{(k)}$$

$$r_2 = ru_{i,1}^{(k+1/2)} - u_{i,0}^{(k)} + gu_{i,1}^{(k)}$$

$$u_{i,0}^{(k+3/4)} = (\alpha r_1 + 2r_2)d_1, \quad u_{i,1}^{(k+3/4)} = (r_1 + \alpha_1 r_2)d_1$$

while $j \leq N-3$, compute

$$r_1 = ru_{i,j}^{(k+1/2)} + gu_{i,j}^{(k)} - u_{i,j+1}^{(k)}$$

$$r_2 = ru_{i,j+1}^{(k+1/2)} - u_{i,j}^{(k)} + gu_{i,j+1}^{(k)}$$

$$u_{i,j}^{(k+3/4)} = (\alpha r_1 + r_2)d, \quad u_{i,j+1}^{(k+3/4)} = (r_1 + \alpha r_2)d$$

$j = j + 2$

$$r_1 = ru_{i,N-1}^{(k+1/2)} + gu_{i,N-1}^{(k)} - u_{i,N}^{(k)}$$

$$r_2 = ru_{i,N}^{(k+1/2)} - 2u_{i,N-1}^{(k)} + 2gu_{i,N}^{(k)}$$

$$u_{i,N-1}^{(k+3/4)} = (\alpha_1 r_1 + r_2)d_1, \quad u_{i,N}^{(k+3/4)} = (2r_1 + \alpha r_2)d_1$$

$i = i + 1$.

Step 4: To compute $\mathbf{u}^{(k+1)}$. $i = 0$, $j = 1$. (change in direction).

while $i \leq N$, compute

$$u_{i,0}^{(k+1)} = u_{i,0}^{(k+3/4)}$$

while $j \leq N-2$, compute

$$r_1 = ru_{i,j}^{(k+3/4)} + gu_{i,j}^{(k)} - u_{i,j+1}^{(k)}$$

$$r_2 = ru_{i,j+1}^{(k+3/4)} - u_{i,j}^{(k)} + gu_{i,j+1}^{(k)}$$

$$u_{i,j}^{(k+1)} = (\alpha r_1 + r_2)d, \quad u_{i,j+1}^{(k+1)} = (r_1 + \alpha r_2)d$$

$$j = j + 2$$

$$u_{i,N}^{(k+1)} = u_{i,N}^{(k+3/4)}$$

$$i = i + 1.$$

Step 5: Repeat *Step* 1 to *Step* 4 until convergence is achieved.

6.2.3 Experimental results

Numerical results presented in this section are for the AGE-DG-2 scheme and SOR method for solving the two dimensional problems subject to the periodic and Neumann boundary conditions discussed in Section 6.2.1 and 6.2.2. One problem is considered for each boundary condition.

Problem 1 - Periodic boundary conditions.

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} - \rho U = -(8\pi^2 + \rho)\sin 2\pi x \sin 2\pi y, \qquad 0 \le x, y \le 1,$$

subject to the boundary condition

$$U(x,0) = U(x,1),\ U_y(x,0) = U_y(x,1), \qquad 0 \le x \le 1$$

$$U(0,y) = U(1,y),\ U_x(0,y) = U_x(1,y), \qquad 0 \le y \le 1.$$

By applying the centred finite difference approximation, for small h, yields

$$-u_{i,j-1} - u_{i-1,j} + 4gu_{i,j} - u_{i+1,j} - u_{i,j+1} =$$

$$(8\pi^2 + \rho)h^2 \sin 2\pi x_i \sin 2\pi y_j,\ i,\ j,\ = 1,\ \ldots,\ N,$$

with $g = 1 + \frac{1}{4}\rho h^2$. The exact solution is $U(x,y) = \sin 2\pi x \sin 2\pi y$.

In the linear system $A\mathbf{u} = \mathbf{b}$, we have the matrix A as in (6.2.1-5), the vector $\mathbf{u}$ as in (6.2.1-6) and the vector $\mathbf{b}$ as (6.2.2-7) with its elements

$$b_{i,j} = (8\pi + \rho)h \sin 2\pi x_i \sin 2\pi y_j,\ i,\ j = 1,\ \ldots,\ N.$$

The results for various ρ are tabulated in *Tables* 6.2.3-1 - 6.2.3-7.

ρ = 0	SOR		AGE-DG-2 (ω = 2)	
N^2	ω	iter	r	iter
100	1.27-1.32	30	1.75-2.07	12
400	1.54-1.58	57	1.06-1.11	24
1600	1.75	108	0.58	48
6400	1.85-1.86	202	0.21	38

Table 6.2.3-1: Problem 1, ρ = 0

ρ = 1	SOR		AGE-DG-2 (ω = 2)	
N^2	ω	iter	r	iter
100	1.31-1.32	29	1.75-2.09	12
400	1.54-1.55	56	1.06-1.12	24
1600	1.74-1.75	107	0.58-0.59	48
6400	1.85-1.86	199	0.21	37

Table 6.2.3-2: Problem 1, ρ = 1

ρ = 5	SOR		AGE-DG-2 (ω = 2)	
N^2	ω	iter	r	iter
100	1.25-1.26	32	1.74-2.16	12
400	1.51-1.53	54	1.11-1.12	23
1600	1.73-1.74	102	0.59-0.60	47
6400	1.85-1.86	190	0.18	41

Table 6.2.3-3: Problem 1, ρ = 5

ρ=10	SOR		AGE-DG-2 (ω = 2)	
N^2	ω	iter	r	iter
100	1.22-1.27	32	1.99-2.11	11
400	1.47-1.53	52	1.09-1.17	23
1600	1.71-1.73	97	0.59-0.62	46
6400	1.85	180	0.18	40

Table 6.2.3-4: Problem 1, ρ = 10

ρ=20	SOR		AGE-DG-2 (ω = 2)	
N^2	ω	iter	r	iter
100	1.27-1.42	26	1.95-2.28	11
400	1.48-1.49	47	1.12-1.23	22
1600	1.70-1.72	88	0.62-0.63	43
6400	1.84-1.85	164	0.22	32

Table 6.2.3-5: Problem 1, ρ = 20

ρ=50	SOR		AGE-DG-2 (ω = 2)	
N^2	ω	iter	r	iter
100	1.28-1.31	19	2.22-2.56	10
400	1.47-1.50	37	1.25-1.33	19
1600	1.69-1.73	70	0.68-0.69	37
6400	1.84-1.85	130	0.24	32

Table 6.2.3-6: Problem 1, ρ = 50

ρ=100	SOR		AGE-DG-2 (ω = 2)	
N^2	ω	iter	r	iter
100	1.26-1.29	15	2.60-3.00	9
400	1.42-1.58	29	1.44-1.49	16
1600	1.68-1.72	53	0.76-0.77	31
6400	1.84-1.85	98	0.21	30

Table 6.2.3-7: Problem 1, ρ = 100

ρ=200	SOR		AGE-DG-2 (ω = 2)	
N^2	ω	iter	r	iter
100	1.10-1.29	12	3.19-3.79	8
400	1.47-1.49	20	1.72-1.77	13
1600	1.66-1.72	37	0.88-0.92	25
6400	1.82-1.85	68	0.22	25

Table 6.2.3-8: Problem 1, ρ = 200

The results presented for Problem 1 with various ρ in this section are based on the numerical experiments. These results show a good improvement in terms of the number of iterations for the AGE-DG-2 scheme over the SOR method which give clear evidence of the superiority of the AGE method. These gains are significant irrespective of the value of ρ. The factors which help to speed up the convergence, especially for the larger number of points are the commutativity properties between G_1 and G_2, and when the matrix A becomes strongly diagonally dominant.

Our concern here is the determination of the optimal single parameter r. To date, no such theory exists on how this parameter might be determined. Again, we will use some similarities from the solution of the one dimensional problem governed by periodic boundary conditions. Let us consider the starting value of N as 10. This will give the grid mesh size $h = 1/10$.

It has been shown that for the one dimensional problem, the experimental $r \in (1,b)$, where $b = 2 + \frac{1}{2}\rho h^2$, is the largest eigenvalue. The value of r, however, is greater than b when the matrix A is strongly diagonally dominant. Since we are now solving a similar but larger problem, then for some ρ, we may expect that the value of r to fall in the interval $(1,b)$, where $b = 2 + \frac{1}{4}\rho h^2$ and for larger ρ, $r > b$.

The results for Problem 1, agree with these assumptions and for $\rho \leq 10$, we may consider the value of $r \in (1,b)$. For $\rho > 0$, we consider $r > b$ as a guide to the experimental value of r.

With these gains, i.e., the number of iterations, the simplicity of the method and that the value of r can easily be determined, indicates that the AGE method to solve the Helmholtz equation governed by periodic boundary conditions might be well be recommended in preference to SOR method.

Problem 2 - Neumann boundary conditions.

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} - \rho U = 6 - \rho(2x^2 + y^2), \quad \rho > 0, \qquad 0 \le x, y \le 1$$

subject to the boundary conditions

$$U_y(x,0) = 0, \; U_y(x,1) = 2, \qquad 0 \le x \le 1,$$

$$U_x(0,y) = 0, \; U_x(1,y) = 4, \qquad 0 \le y \le 1.$$

By applying central finite difference approximations, for small h, yields

$$- u_{i,j-1} - u_{i-1,j} + 4g u_{i,j} - u_{i+1,j} - u_{i,j+1} = h^2[\rho(2x_i^2 + y_j^2) - 6 \,]$$

$$i, \; j, \; = 1, \; \ldots, \; N,$$

where $g = 1 + \frac{1}{4}\rho h^2$, with $h = 1/N$.

The exact solution is $U(x,y) = 2x^2 + y^2$.

In the linear system $A\mathbf{u} = \mathbf{b}$, we have the matrix A as in (6.2.2-8), the vector $\mathbf{u}$ as in (6.2.2-9) and the vector $\mathbf{b}$ as (6.2.2-10) with its elements

$$b_{i,N} = h^2[\rho(2x_i^2 + y_N^2) - 6 \,] + 4h, \; i = 0, \; \ldots, \; N,$$

$$b_{N,j} = h^2[\rho(2x_N^2 + y_j^2) - 6 \,] + 8h, \; j = 0, \; \ldots, \; N$$

and $b_{i,j} = h^2[\rho(2x_i^2 + y_j^2) - 6 \,], \; i, \; j = 0 \; \ldots, \; N-1.$

The results for various ρ are tabulated in *Tables* 6.2.3-9 - 6.2.3-16.

$\rho = 1$	SOR		AGE-DG-2 ($\omega = 2$)	
$(N+1)^2$	ω	iter	r	iter
100	1.86	77	0.09	54
400	1.93	167	0.01	49
1600	1.97	340	0.005	99
6400	1.984	567	0.002	186

Table 6.2.3-9: Problem 2, $\rho = 1$

ρ = 5	SOR		AGE-DG-2 (ω = 2)	
$(N+1)^2$	ω	iter	r	iter
100	1.72-1.73	41	0.35	32
400	1.85	82	0.07-0.08	48
1600	1.93	171	0.02	66
6400	1.96	349	0.01	132

Table 6.2.3-10: Problem 2, ρ = 5

ρ=10	SOR		AGE-DG-2 (ω = 2)	
$(N+1)^2$	ω	iter	r	iter
100	1.62	32	0.50-0.54	25
400	1.80	65	0.14-0.17	36
1600	1.90	130	0.04	61
6400	1.95	235	0.02	119

Table 6.2.3-11: Problem 2, ρ = 10

ρ=20	SOR		AGE-DG-2 (ω = 2)	
$(N+1)^2$	ω	iter	r	iter
100	1.50-1.51	26	0.72-0.80	19
400	1.74	52	0.26-0.27	29
1600	1.87	101	0.07	43
6400	1.93	204	0.04	87

Table 6.2.3-12: Problem 2, ρ = 20

ρ=50	SOR		AGE-DG-2 (ω = 2)	
$(N+1)^2$	ω	iter	r	iter
100	1.34	20	1.20-1.29	13
400	1.60	40	0.47-0.51	22
1600	1.78	81	0.14-0.18	32
6400	1.89	161	0.07	58

Table 6.2.3-13: Problem 2, ρ = 50

ρ=100	SOR		AGE-DG-2 (ω = 2)	
$(N+1)^2$	ω	iter	r	iter
100	1.26	16	1.80-1.86	10
400	1.52	31	0.71-0.79	17
1600	1.72	60	0.25	25
6400	1.85	116	0.11-0.12	46

Table 6.2.3-14: Problem 2, ρ = 100

ρ=200	SOR		AGE-DG-2 (ω = 2)	
$(N+1)^2$	ω	iter	r	iter
100	1.18-1.20	12	2.31-3.49	9
400	1.39-1.41	23	1.03-1.21	13
1600	1.64	42	0.42-0.43	20
6400	1.79	80	0.16-0.17	33

Table 6.2.3-15: Problem 2, ρ = 200

ρ=400	SOR		AGE-DG-2 (ω = 2)	
$(N+1)^2$	ω	iter	r	iter
100	1.09-1.13	9	3.60-5.80	8
400	1.27-1.29	16	1.55-1.76	10
1600	1.52-1.53	30	0.65-0.72	16
6400	1.72	56	0.21-0.27	24

Table 6.2.3-16: Problem 2, ρ = 400

The results repeat the impressive performance by the AGE method over the SOR method as shown in the one dimensional problems governed by Neumann boundary conditions. In addition, the gains in terms of the number of iterations are significant irrespective of the value of ρ.

Our concern is again to find a suitable method to determine the optimal single parameter r, theoretically, that would become a guide to determine the value of r, experimentally. It has been shown that for the one dimensional problems, $r \in [a,\sqrt{ab}]$ for smaller ρ, and $r \in [\sqrt{ab},b]$ for larger value of ρ. However, this assumption is not valid for the two dimensional problem as the relation $r = \sqrt{ab}$ no longer exists.

Instead, we will consider the relation $r = (a+b)/2$. Since we are solving a similar but higher dimensional problems, we may expect that the value of r would either be in $[a,(a+b)/2]$ for smaller values of ρ, or in $[(a+b)/2,b]$, for larger ρ. The results in *Table* 6.2.3-17 confirm this argument.

N = 9	AGE-DG-2	The eigenvalues		
ρ	r	a	$(a+b)/2$	b
1	0.09	0.003	1.504	3.005
5	0.35	0.015	1.521	3.026
10	0.50-0.54	0.031	1.541	3.052
20	0.72-0.80	0.062	1.582	3.103
50	1.20-1.29	0.154	1.707	3.259
100	1.80-1.86	0.309	1.915	3.521
200	2.31-3.49	0.617	2.336	4.055
400	3.60-5.80	1.235	3.194	5.154

Table 6.2.3-17: Problem 2 - the range for r

The results outline that for $\rho \leq 100$, the value of r is within the interval $[a,(a+b)/2]$. For larger value of ρ, r is in $[(a+b)/2,b]$. By having this starting value, the value of r for larger numbers of points, can easily be determined, i.e., this value is in a smaller range than the previous interval.

It should be noticed that, these results do not include the case when $\rho = 0$ as the matrix A will yield a zero eigenvalue. Consequently, the matrix A is singular and the computed solution will not converge to the exact solution but to the corresponding eigenvector of the matrix A. The results obtained for other values of ρ are only from one problem but it is sufficient. A given problem differs only on the right hand side vector b, hence, it will only affect the number of iterations.

With this findings, the AGE method can be considered to be better than the SOR method, and be recommended to solve the Helmholtz equation in two dimensions.

6.3 The solution with alternative computational forms

It has been shown in Section 4.2 that the solution of the two-point boundary-value problem with alternative forms give an improvement on the CPU time. In this section, we will investigate further the forms given in Section 4.2 to solve the two and three dimensional problems.

With the limitation of the method, our discussion in Section 6.1 and 6.2 is mainly concerned with the AGE-DG and AGE-GT schemes. As shown in Section 6.1, between these two schemes the AGE-DG performed better in terms of computational work. Thus, we would expect more laborious work if the AGE-GT scheme is rearranged into other forms. Thus, we will confine our investigation only to the AGE-DG scheme.

6.3.1 The computational form of the AGE-DG method

Let us recall the respective Algorithms 6.1.3-1 and 6.1.3-2 for the AGE-DG-2 and AGE-DG-3 schemes. In these algorithms, we have used the intermediate variables r_1 and r_2 prior to computing the vector **u** and its intermediate value at each step. Our aim now is to eliminate these variables so that each equation will be in a complete computational molecule form. Having eliminated r_1 and r_2 at each step, the algorithm for the AGE-DG-2 scheme in this form, i.e., the COMP-AGE-DG-2 scheme can be written as follows.

Algorithm 6.3.1-1: The COMP-AGE-DG-2 scheme.

Set $u_{i,j}^{(k)} = 0$, $i, j = 0, \ldots, N+1$, $\alpha = r + g$, $d = 1/(\alpha^2 - 1)$, $s = \omega - 1$,

$t = \alpha - 4g\omega$, $D = d\omega$, $A = D\alpha$, $B = d(\alpha t + s)$, $C = d(\alpha s + t)$,

$Q = dr$, $P = Q\alpha$, $R = d(\alpha g - 1)$, $T = d(g - \alpha)$.

Step 1: To compute $u^{(k+1/4)}$. $i = 1$, $j = 1$.

while $j \le N$, compute

while $i \le N-2$, compute

$$u_{i,j}^{(k+1/4)} = + Au_{i-1,j}^{(k)} + Au_{i,j-1}^{(k)} + Au_{i,j+1}^{(k)} + Bu_{i,j}^{(k)} + Cu_{i+1,j}^{(k)} + Du_{i+2,j}^{(k)} + Du_{i+1,j-1}^{(k)} + Du_{i+1,j+1}^{(k)} + Ab_{i,j} + Db_{i+1,j}$$

$$u_{i+1,j}^{(k+1/4)} = Du_{i-1,j}^{(k)} + Du_{i,j-1}^{(k)} + Du_{i,j+1}^{(k)} + Cu_{i,j}^{(k)} + Bu_{i+1,j}^{(k)} + Au_{i+2,j}^{(k)} + Au_{i+1,j-1}^{(k)} + Au_{i+1,j+1}^{(k)} + Db_{i,j} + Ab_{i+1,j}$$

$i = i + 2$

$$u_{N,j}^{(k+1/4)} = (\omega u_{N,j-1}^{(k)} + \omega u_{N-1,j}^{(k)} + tu_{N,j}^{(k)} + \omega u_{N,j+1}^{(k)} + \omega b_{N,j})/\alpha$$

$j = j + 1$.

Step 2: To compute $u^{(k+1/2)}$. $i = 2$, $j = 1$.

while $j \le N$, compute

$$u_{1,j}^{(k+1/2)} = (ru_{1,j}^{(k+1/4)} + gu_{1,j}^{(k)})/\alpha$$

while $i \le N-1$, compute

$$u_{i,j}^{(k+1/2)} = Pu_{i,j}^{(k+1/4)} + Qu_{i+1,j}^{(k+1/4)} + Ru_{i,j}^{(k)} + Tu_{i+1,j}^{(k)}$$

$$u_{i+1,j}^{(k+1/2)} = Qu_{i,j}^{(k+1/4)} + Pu_{i+1,j}^{(k+1/4)} + Tu_{i,j}^{(k)} + Ru_{i+1,j}^{(k)}$$

$i = i + 2$

$j = j + 1$.

Step 3: To compute $u^{(k+3/4)}$. $i = 1$, $j = 1$. (Change in direction).

while $i \le N$, compute

while $j \le N-2$, compute

$$u_{i,j}^{(k+3/4)} = Pu_{i,j}^{(k+1/2)} + Qu_{i,j+1}^{(k+1/2)} + Ru_{i,j}^{(k)} + Tu_{i,j+1}^{(k)}$$

$$u_{i,j+1}^{(k+3/4)} = Qu_{i,j}^{(k+1/2)} + Pu_{i,j+1}^{(k+1/2)} + Tu_{i,j}^{(k)} + Ru_{i,j+1}^{(k)}$$

$j = j + 2$

$$u_{i,N}^{(k+3/4)} = (ru_{i,N}^{(k+1/2)} + gu_{i,N}^{(k)})/\alpha$$

$i = i + 1.$

Step 4: To compute $\mathbf{u}^{(k+1)}$. $i = 1$, $j = 2$. (change in direction).

while $i \leq N$, compute

$$u_{i,1}^{(k+1)} = (ru_{i,1}^{(k+3/4)} + gu_{i,1}^{(k)})/\alpha$$

while $j \leq N-1$, compute

$$u_{i,j}^{(k+1)} = Pu_{i,j}^{(k+3/4)} + Qu_{i,j+1}^{(k+3/4)} + Ru_{i,j}^{(k)} + Tu_{i,j+1}^{(k)}$$

$$u_{i,j+1}^{(k+1)} = Qu_{i,j}^{(k+3/4)} + Pu_{i,j+1}^{(k+3/4)} + Tu_{i,j}^{(k)} + Su_{i,j+1}^{(k)}$$

$j = j + 2$

$i = i + 1.$

Step 5: Repeat *Step* 1 to *Step* 4 until convergence is achieved.

The computational molecule for each intermediate vector u derived from Algorithm 6.3.1-1, can be given as follows:

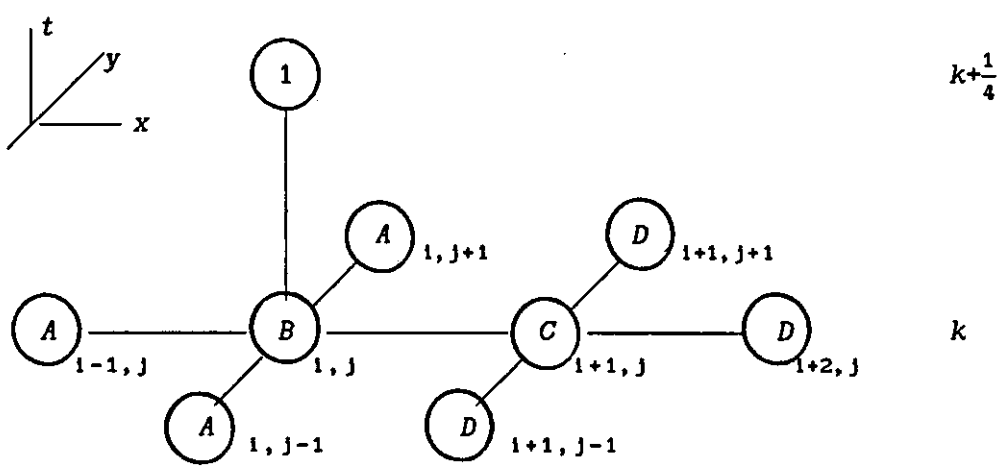


Figure 6.3.1-1: The computational molecule for $u_{i,j}^{(k+1/4)}$, for large N, Algorithm 6.3.1-1.

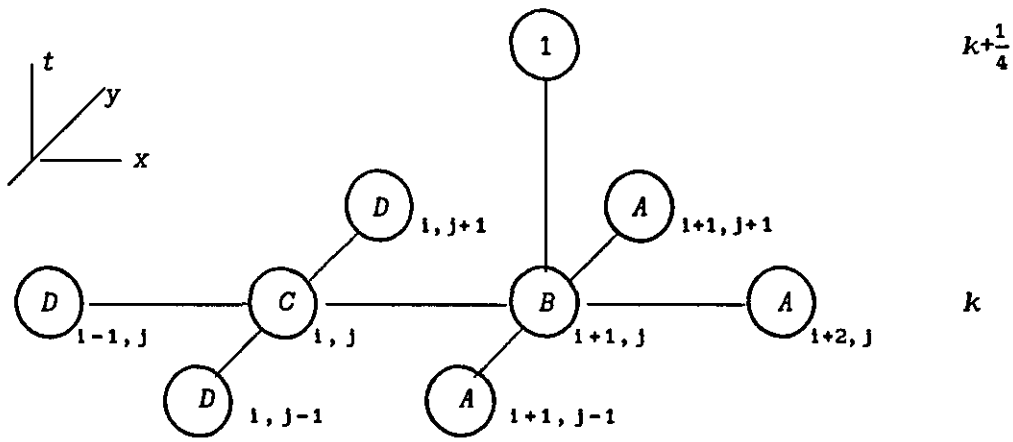


Figure 6.3.1-2: The computational molecule for $u_{i+1,j}^{(k+1/4)}$, for large N, Algorithm 6.3.1-1.

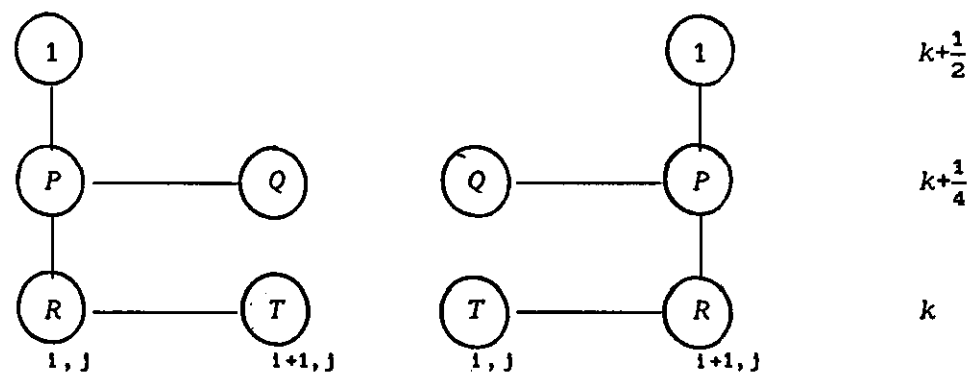


Figure 6.3.1-3: The computational molecule for $u_{i,j}^{(k+1/2)}$ and $u_{i,j+1}^{(k+1/2)}$, for large N, Algorithm 6.3.1-1.

The computation for the vectors $\mathbf{u}^{(k+3/4)}$ and $\mathbf{u}^{(k+1)}$ has a similar form as $\mathbf{u}^{(k+1/2)}$. Thus, is it sufficient to present only the molecules for this vector.

The algorithm for the AGE-DG-3 scheme in the computational form, i.e., the COMP-AGE-DG-3 scheme, can be presented as follows:

Algorithm 6.3.1-2: The COMP-AGE-DG-3 scheme.

Set $u_{i,j,k}^{(k)} = 0$, i, j, $k = 0, \ldots, N+1$, $\alpha = r + g$, $d = 1/(\alpha^2 - 1)$,

$s = \omega - 1$, $t = \alpha - 4g\omega$, $D = d\omega$, $A = D\alpha$, $B = d(\alpha t + s)$,

$C = d(\alpha s + t)$, $Q = dr$, $P = Q\alpha$, $R = d(\alpha g - 1)$, $T = d(g - \alpha)$.

Step 1: To compute $\mathbf{u}^{(k+1/6)}$. Set i, j, $k = 1$.

while $k \le N$, compute

while $j \le N$, compute

while $i \le N-2$, compute

$$u_{i,j,k}^{(k+1/6)} = Au_{i-1,j,k}^{(k)} + Au_{i,j,k-1}^{(k)} + Au_{i,j,k+1}^{(k)} + Bu_{i,j,k}^{(k)} + Au_{i,j-1,k}^{(k)} + Au_{i,j+1,k}^{(k)} + Du_{i+1,j,k-1}^{(k)} + Du_{i+1,j,k+1}^{(k)} + Cu_{i+1,j,k}^{(k)} + Du_{i+1,j-1,k}^{(k)} + Du_{i+1,j+1,k}^{(k)} + Du_{i+2,j,k}^{(k)} + Ab_{i,j,k} + Db_{i+1,j,k}$$

$$u_{i+1,j,k}^{(k+1/6)} = Du_{i-1,j,k}^{(k)} + Du_{i,j,k-1}^{(k)} + Du_{i,j,k+1}^{(k)} + Cu_{i,j,k}^{(k)} + Du_{i,j-1,k}^{(k)} + Du_{i,j+1,k}^{(k)} + Au_{i+1,j,k-1}^{(k)} + Au_{i+1,j,k+1}^{(k)} + Bu_{i+1,j,k}^{(k)} + Au_{i+1,j-1,k}^{(k)} + Au_{i+1,j+1,k}^{(k)} + Au_{i+2,j,k}^{(k)} + Db_{i,j,k} + Ab_{i+1,j,k}$$

$i = i + 2$

$$u_{N,j,k}^{(k+1/6)} = (\omega u_{N,j,k-1}^{(k)} + \omega u_{N,j-1,k}^{(k)} + \omega u_{N-1,j,k}^{(k)} + tu_{N,j,k}^{(k)} + \omega u_{N,j+1,k}^{(k)} + \omega u_{N,j,k+1}^{(k)} + \omega b_{N,j,k})/\alpha$$

$j = j + 1$

$k = k + 1$.

Step 2: To compute $\mathbf{u}^{(k+1/3)}$. Set $i = 2$, j, $k = 1$.

while $k \le N$, compute

while $j \le N$, compute

$$u_{1,j,k}^{(k+1/3)} = (ru_{1,j,k}^{(k+1/6)} + gu_{1,j,k}^{(k)})/\alpha$$

while $i \le N-1$, compute

$$u_{i,j,k}^{(k+1/3)} = Pu_{i,j,k}^{(k+1/6)} + Qu_{i+1,j,k}^{(k+1/6)} + Ru_{i,j,k}^{(k)} + Tu_{i+1,j,k}^{(k)}$$

$$u_{i+1,j,k}^{(k+1/3)} = Qu_{i,j,k}^{(k+1/6)} + Pu_{i+1,j,k}^{(k+1/6)} + Tu_{i,j,k}^{(k)} + Ru_{i+1,j,k}^{(k)}$$

$i = i + 2$

$j = j + 1$

$k = k + 1.$

Step 3: To compute $\mathbf{u}^{(k+1/2)}$. Set i, j, $k = 1$. (Change in direction)

while $i \leq N$, compute

while $k \leq N$, compute

while $j \leq N-1$, compute

$$u_{i,j,k}^{(k+1/2)} = Pu_{i,j,k}^{(k+1/3)} + Qu_{i,j+1,k}^{(k+1/3)} + Ru_{i,j,k}^{(k)} + Tu_{i,j+1,k}^{(k)}$$

$$u_{i,j+1,k}^{(k+1/2)} = Qu_{i,j,k}^{(k+1/3)} + Pu_{i,j+1,k}^{(k+1/3)} + Tu_{i,j,k}^{(k)} + Ru_{i,j+1,k}^{(k)}$$

$j = j + 2$

$$u_{i,N,k}^{(k+1/2)} = (ru_{i,N,k}^{(k+1/3)} + gu_{i,N,k}^{(k)})/\alpha$$

$k = k + 1$

$i = i + 1$

Step 4: To compute $\mathbf{u}^{(k+2/3)}$. Set i, $k = 1$, $j = 2$.

(Change in direction)

while $i \leq N$, compute

while $k \leq N$, compute

$$u_{i,1,k}^{(k+2/3)} = (ru_{i,1,k}^{(k+1/2)} + gu_{i,1,k}^{(k)})/\alpha$$

while $j \leq N-1$, compute

$$u_{i,j,k}^{(k+2/3)} = Pu_{i,j,k}^{(k+1/2)} + Qu_{i,j+1,k}^{(k+1/2)} + Ru_{i,j,k}^{(k)} + Tu_{i,j+1,k}^{(k)}$$

$$u_{i,j+1,k}^{(k+2/3)} = Qu_{i,j,k}^{(k+1/2)} + Pu_{i,j+1,k}^{(k+1/2)} + Tu_{i,j,k}^{(k)} + Ru_{i,j+1,k}^{(k)}$$

$j = j + 2$

$k = k + 1$

$i = i + 1.$

Step 5: To compute $\mathbf{u}^{(k+5/6)}$. Set i, j, $k = 1$. (Change in direction)

while $j \le N$, compute

while $i \le N$, compute

while $k \le N-1$, compute

$$u_{i,j,k}^{(k+5/6)} = Pu_{i,j,k}^{(k+2/3)} + Qu_{i,j,k+1}^{(k+2/3)} + Ru_{i,j,k}^{(k)} + Tu_{i,j,k+1}^{(k)}$$

$$u_{i,j,k+1}^{(k+5/6)} = Qu_{i,j,k}^{(k+2/3)} + Pu_{i,j,k+1}^{(k+2/3)} + Tu_{i,j,k}^{(k)} + Ru_{i,j,k+1}^{(k)}$$

$k = k + 2$

$$u_{i,j,N}^{(k+5/6)} = (ru_{i,j,N}^{(k+2/3)} + gu_{i,j,N}^{(k)})/\alpha$$

$i = i + 1$

$j = j + 1.$

Step 6: To compute $\mathbf{u}^{(k+1)}$. Set i, $j = 1$, $k = 2$. (Change in direction)

while $j \le N$, compute

while $i \le N$, compute

$$u_{i,j,1}^{(k+1)} = (ru_{i,j,1}^{(k+5/6)} + gu_{i,j,1}^{(k)})/\alpha$$

while $k \le N-1$, compute

$$u_{i,j,k}^{(k+1)} = Pu_{i,j,k}^{(k+5/6)} + Qu_{i,j,k+1}^{(k+5/6)} + Ru_{i,j,k}^{(k)} + Tu_{i,j,k+1}^{(k)}$$

$$u_{i,j,k+1}^{(k+1)} = Qu_{i,j,k}^{(k+5/6)} + Pu_{i,j,k+1}^{(k+5/6)} + Tu_{i,j,k}^{(k)} + Ru_{i,j,k+1}^{(k)}$$

$k = k + 2$

$i = i + 1$

$j = j + 1.$

Step 7: Repeat *Step* 1 to *Step* 6 until convergence is achieved.

The computational molecule for each intermediate vector u derived from Algorithm 6.3.1-2, can be given as follows:

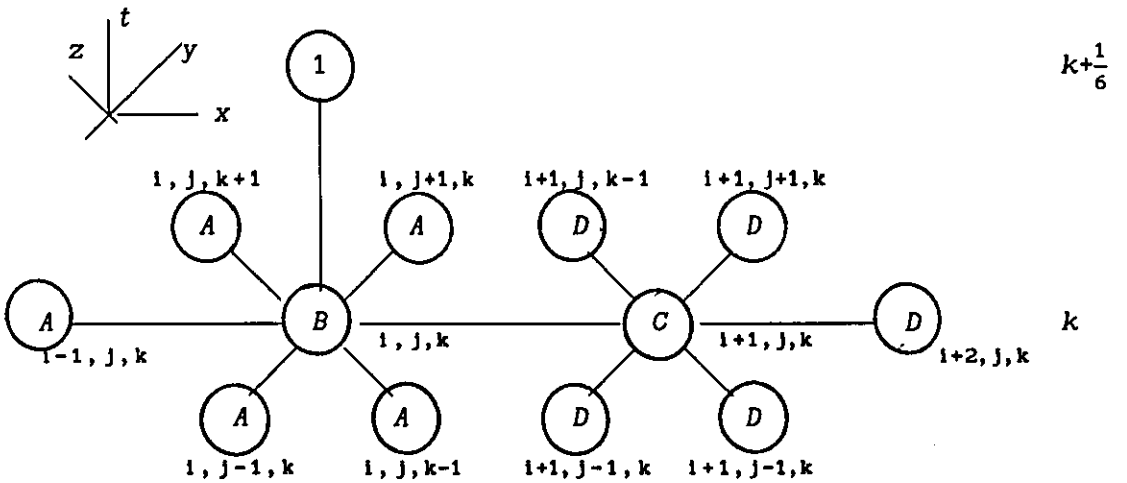


Figure 6.3.1-4: The computational molecule for $u_{i,j,k}^{(k+1/6)}$, for large N, Algorithm 6.3.1-2.

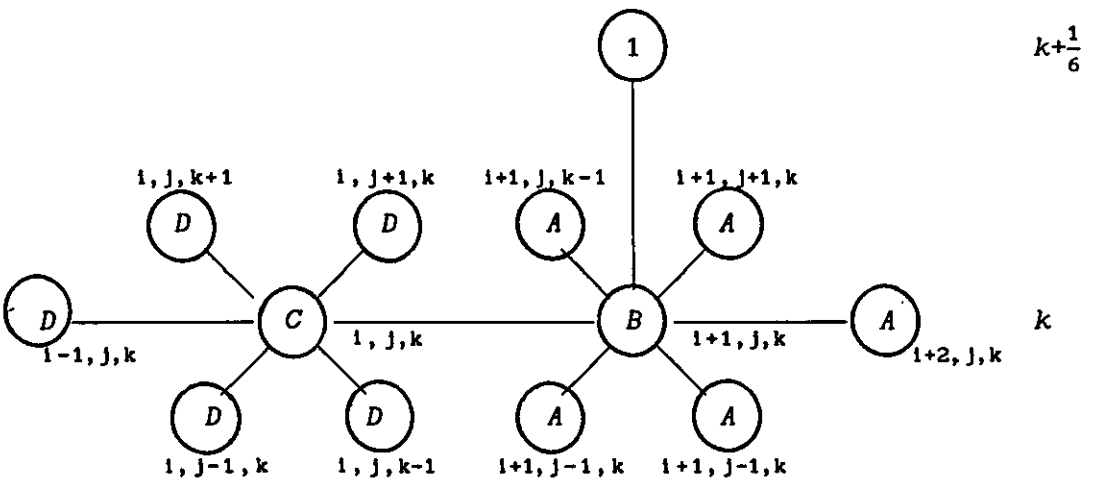


Figure 6.3.1-5: The computational molecule for $u_{i+1,j,k}^{(k+1/6)}$, for large N, Algorithm 6.3.1-2.

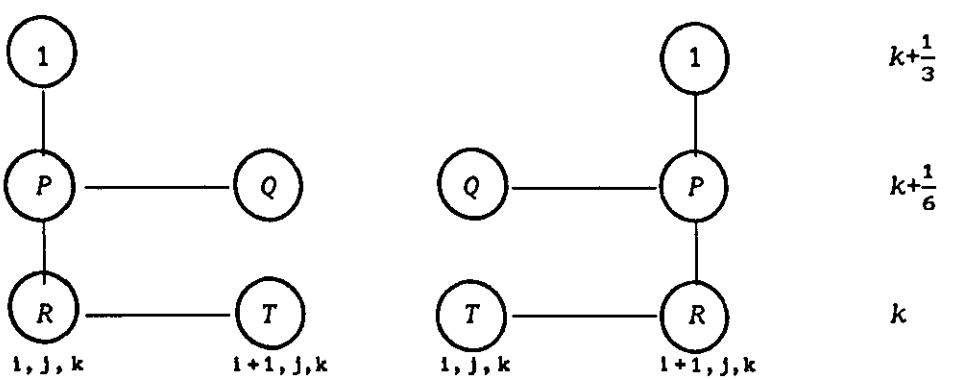


Figure 6.3.1-6: The computational molecule for $u_{i,j,k}^{(k+1/3)}$ and $u_{i+1,j,k}^{(k+1/3)}$, for large N, Algorithm 6.3.1-2.

From the Algorithm 6.3.1-2, we find that the vectors $u^{(k+1/2)}$, $u^{(k+2/3)}$, $u^{(k+5/6)}$ and $u^{(k+1)}$ have a similar computational form as $u^{(k+1/3)}$. Thus, it is sufficient to present the molecules from this vector.

Before we summarize our findings, i.e., the number of operations per iteration for this scheme, let us consider the other form of presentation of the AGE-DG scheme, called the coupled AGE (CAGE) method.

6.3.2 The CAGE form of the AGE-DG method

We now consider the solution of vector **u** in a coupled form, where the four equations and six equations in the respective AGE-DG-2 and AGE-DG-3 schemes are made up into two and three equations. In other words, the first equation is combined with the second equation, the third with the fourth and the fifth with the sixth (in the AGE-DG-3 scheme).

Let us recall the AGE-DG-2 scheme, i.e., equations (6.1.3-10) - (6.1.3-13) which can be written explicitly as

$$u^{(k+1/4)} = [I - \omega(rI + G_1)^{-1}A]u^{(k)} + \omega(rI + G_1)^{-1}b \qquad (6.3.2\text{-}1)$$

$$u^{(k+1/2)} = (rI + G_2)^{-1}[ru^{(k+1/4)} + G_2u^{(k)}] \qquad (6.3.2\text{-}2)$$

$$u_c^{(k+3/4)} = (rI + G_1)^{-1}[ru_c^{(k+1/2)} + G_1u_c^{(k)}] \qquad (6.3.2\text{-}3)$$

$$u_c^{(k+1)} = (rI + G_2)^{-1}[ru_c^{(k+3/4)} + G_2u_c^{(k)}]. \qquad (6.3.2\text{-}4)$$

In the coupled form, equations (6.3.2-1) - (6.3.2-4) can be explicitly combined to give a set of new equations, i.e., the CAGE-DG-2 scheme,

$$u^{(k+1/2)} = [I - \omega r(rI + G_2)^{-1}(rI + G_1)^{-1}A]u^{(k)} + \omega r(rI + G_2)^{-1}(rI + G_1)^{-1}b \qquad (6.3.2\text{-}5)$$

$$u_c^{(k+1)} = r^2(rI + G_2)^{-1}(rI + G_1)^{-1}w_c^* + u_c^{(k)} \qquad (6.3.2\text{-}6)$$

where $w_c^* = u_c^{(k+1/2)} - u_c^{(k)}$.

Let us consider A and $(G_1 + G_2)$ as given in (6.1.1-11) and (6.1.1-15) respectively. Obviously, for $\omega = 2$, equation (6.3.2-5) is similar to the CAGE in one dimensional problem, i.e., equation (4.2.2-3), except that the matrices A, G_1 and G_2 are derived from a two dimensional problem. Thus, we may expect that the matrix

$$C_r = I - \omega r(rI + G_2)^{-1}(rI + G_1)^{-1}A \qquad (6.3.2\text{-}7)$$

at least has the form of the matrix in (4.2.2-6), and

$$D_r = \omega r(rI + G_2)^{-1}(rI + G_1)^{-1} \text{ and } E_r = r^2(rI + G_2)^{-1}(rI + G_1)^{-1} \qquad (6.3.2\text{-}8)$$

have the form of (4.2.2-7). We now write the matrices C_r, D_r and E_r for the case when $N = 7$.

From (6.1.1-15), for general N (odd) and for any $r > 0$, we have

$$(rI + G_1)^{-1} = \begin{bmatrix} \tilde{G}_1^{-1} & & & 0 \\ & \tilde{G}_1^{-1} & & \\ & & \ddots & \\ & & & \tilde{G}_1^{-1} \\ 0 & & & \tilde{G}_1^{-1} \end{bmatrix}_{N^2 X N^2}, \quad \tilde{G}_1^{-1} = \left[\begin{array}{cc|cc|c} \alpha d & d & & & \\ d & \alpha d & & & \\ \hline & & \ddots & & \\ \hline & & & \alpha d & d \\ & & & d & \alpha d \\ \hline & & & & 1/\alpha \end{array}\right] \qquad (6.3.2\text{-}9)$$

$$(rI + G_2)^{-1} = \begin{bmatrix} \tilde{G}_2^{-1} & & & 0 \\ & \tilde{G}_2^{-1} & & \\ & & \ddots & \\ & & & \tilde{G}_2^{-1} \\ 0 & & & \tilde{G}_2^{-1} \end{bmatrix}_{N^2 X N^2}, \quad \tilde{G}_2^{-1} = \left[\begin{array}{c|cc|cc} 1/\alpha & & & & \\ \hline & \alpha d & d & & \\ & d & \alpha d & & \\ \hline & & & \ddots & \\ \hline & & & & \alpha d \quad d \\ & & & & d \quad \alpha d \end{array}\right] \qquad (6.3.2\text{-}10)$$

where $\alpha = r + g$ and $d = 1/(\alpha^2 - 1)$.

Now, we multiply the matrices $(rI + G_2)^{-1}$ and $(rI + G_1)^{-1}$ gives

$$(rI + G_2)^{-1}(rI + G_1)^{-1} = \begin{bmatrix} \tilde{G}_2^{-1} & & 0 \\ & \tilde{G}_2^{-1} & \\ & & \ddots \\ 0 & & \tilde{G}_2^{-1} \end{bmatrix} \begin{bmatrix} \tilde{G}_1^{-1} & & 0 \\ & \tilde{G}_1^{-1} & \\ & & \ddots \\ 0 & & \tilde{G}_1^{-1} \end{bmatrix}_{N^2XN^2}$$

$$= \begin{bmatrix} \tilde{G}_2^{-1}\tilde{G}_1^{-1} & & & 0 \\ & \tilde{G}_2^{-1}\tilde{G}_1^{-1} & & \\ & & \ddots & \\ & & & \tilde{G}_2^{-1}\tilde{G}_1^{-1} \\ 0 & & & \end{bmatrix}_{N^2XN^2} \qquad (6.3.2\text{-}11)$$

where, for the case $N = 7$,

$$\tilde{G}_2^{-1}\tilde{G}_1^{-1} = \begin{bmatrix} d & S_4 & & & & \\ S_2 & S_3 & S_2 & S_1 & & \\ S_1 & S_2 & S_3 & S_2 & & \\ & & S_2 & S_3 & S_2 & S_1 \\ & & S_1 & S_2 & S_3 & S_2 \\ & & & S_2 & S_3 & S_4 \\ & & & S_1 & S_2 & d \end{bmatrix} \qquad (6.3.2\text{-}12)$$

where $S_1 = d^2$, $S_2 = S_1\alpha$, $S_3 = S_2\alpha$ and $S_4 = d/\alpha$. Clearly,

$$D_r = \omega r\, \tilde{G}_2^{-1}\tilde{G}_1^{-1} \text{ and } E_r = r^2\, \tilde{G}_2^{-1}\tilde{G}_1^{-1}. \qquad (6.3.2\text{-}13)$$

Now, for the matrix C_r with the matrix A as in (6.1.1-11), we have

$$C_r = I - \omega r(rI + G_2)^{-1}(rI + G_1)^{-1}A$$

$$= \begin{bmatrix} A_r & -D_r & & & & & \\ -D_r & A_r & -D_r & & & 0 & \\ & -D_r & A_r & -D_r & & & \\ & & -D_r & A_r & -D_r & & \\ & & & -D_r & A_r & -D_r & \\ & & & & -D_r & A_r & -D_r \\ & 0 & & & & -D_r & A_r \end{bmatrix}_{7^2X7^2} \qquad (6.3.2\text{-}14)$$

where

$$A_r = \begin{bmatrix} A & B & C & & & & \\ Q & R & S & T & V & 0 & \\ T & S & R & Q & P & & \\ & P & Q & R & S & T & V \\ & V & T & S & R & Q & P \\ & 0 & & P & Q & W & X \\ & & & V & T & Y & Z \end{bmatrix} \qquad (6.3.2\text{-}15)$$

with $d_1 = d\omega r$, $C = d_1/\alpha$, $V = d_1 d$, $A = 1 + C(1 - 4g\alpha)$, $B = C(1 - 4g)$,

$P = V\alpha$, $Q = P(\alpha - 4g)$, $R = 1 + 2P(1 - 2g\alpha)$, $S = V(1 + \alpha^2 - 4g\alpha)$,

$T = V(\alpha - 4g)$, $W = 1 + C + P(1 - 4g\alpha)$, $X = C(d\alpha^3 - 4g)$,

$Y = d_2[1 + d(1 - 4g\alpha)]$, $Z = 1 + P - 4d_1 g$.

The computation that involves the vector **b** can be performed outside the iteration loop and can be assigned to a single array. We now write the algorithm for the CAGE-DG-2 scheme for general N (odd).

Algorithm 6.3.2-1: The CAGE-DG-2 scheme.

Set $u_{i,j}^{(k)} = 0$, $i, j = 0, \ldots, N+1$, $\alpha = r + g$, $d = 1/(\alpha^2 - 1)$,

and other coefficients as given in (6.3.2-11) - (6.3.2-14).

Step 1: To compute the vector **v**. Set $i = 2$, $j = 1$.

while $j \le N$, compute

$v_{1,j} = \omega r(db_{1,j} + S_4 b_{2,j})$

while $i \le N-3$, compute

$v_{i,j} = \omega r(S_2 b_{i-1,j} + S_3 b_{i,j} + S_2 b_{i+1,j} + S_1 b_{i+2,j})$

$v_{i+1,j} = \omega r(S_1 b_{i-1,j} + S_2 b_{i,j} + S_3 b_{i+1,j} + S_2 b_{i+2,j})$

$i = i + 2$

$v_{N-1,j} = \omega r(S_2 b_{N-1,j} + S_3 b_{N,j} + S_4 b_{N+1,j})$

$v_{N,j} = \omega r(S_1 b_{N-1,j} + S_2 b_{N,j} + db_{N,j})$

$j = j + 1$.

Step 2: To compute $u^{(k+1/2)}$. Set $i = 2$, $j = 1$.

while $j \le N$, compute

$$u_{1,j}^{(k+1/2)} = -du_{1,j-1}^{(k)} - S_4 u_{2,j-1}^{(k)} + Au_{1,j}^{(k)} + Bu_{2,j}^{(k)} + Cu_{3,j}^{(k)} - du_{1,j+1}^{(k)} - S_4 u_{2,j+1}^{(k)}$$

while $i \le N-3$, compute

$$u_{i,j}^{(k+1/2)} = -S_2 u_{i-1,j-1}^{(k)} - S_3 u_{i,j-1}^{(k)} - S_2 u_{i+1,j-1}^{(k)} - S_1 u_{i+2,j-1}^{(k)} + Pu_{i-2,j}^{(k)} + Qu_{i-1,j}^{(k)} + Ru_{i,j}^{(k)} + Su_{i+1,j}^{(k)} + Tu_{i+2,j}^{(k)} + Vu_{i+3,j}^{(k)} - S_2 u_{i-1,j+1}^{(k)} - S_3 u_{i,j+1}^{(k)} - S_2 u_{i+1,j+1}^{(k)} - S_1 u_{i+2,j+1}^{(k)} + v_{i,j}$$

$$u_{i+1,j}^{(k+1/2)} = -S_1 u_{i-1,j-1}^{(k)} - S_2 u_{i,j-1}^{(k)} - S_3 u_{i+1,j-1}^{(k)} - S_2 u_{i+2,j-1}^{(k)} + Vu_{i-2,j}^{(k)} + Tu_{i-1,j}^{(k)} + Su_{i,j}^{(k)} + Ru_{i+1,j}^{(k)} + Qu_{i+2,j}^{(k)} + Pu_{i+3,j}^{(k)} - S_1 u_{i-1,j+1}^{(k)} - S_2 u_{i,j+1}^{(k)} - S_3 u_{i+1,j+1}^{(k)} - S_2 u_{i+2,j+1}^{(k)} + v_{i+1,j}$$

$i = i + 2$

$$u_{N-1,j}^{(k+1/2)} = -S_2 u_{N-1,j-1}^{(k)} - S_3 u_{N,j-1}^{(k)} - S_4 u_{N+1,j-1}^{(k)} + Pu_{N-2,j}^{(k)} + Qu_{N-1,j}^{(k)} + Wu_{N,j}^{(k)} + Xu_{N+1,j}^{(k)} - S_2 u_{N-1,j+1}^{(k)} - S_3 u_{N,j+1}^{(k)} - S_4 u_{N+1,j+1}^{(k)} + v_{N-1,j}$$

$$u_{N,j}^{(k+1/2)} = -S_1 u_{N-1,j-1}^{(k)} - S_2 u_{N,j-1}^{(k)} - du_{N+1,j-1}^{(k)} + Vu_{N-2,j}^{(k)} + Tu_{N-1,j}^{(k)} + Yu_{N,j}^{(k)} + Zu_{N+1,j}^{(k)} - S_1 u_{N-1,j+1}^{(k)} - S_2 u_{N,j+1}^{(k)} - du_{N+1,j+1}^{(k)} + v_{N,j}$$

$j = j + 1$.

Step 3: To compute the vector w^*.

for $j = 1$ to N, compute

for $i = 1$ to N, compute

$$w^{*}_{i,j} = u^{(k+1/2)}_{i,j} - u^{(k)}_{i,j}.$$

Step 4: To compute $\mathbf{u}^{(k+1)}$. Set $i = 1$, $j = 2$. (change in direction).

while $i \le N$, compute

$$u^{(k+1)}_{1,1} = r^2(dw^{*}_{1,1} + S_4 w^{*}_{1,2}) + u^{(k)}_{1,1}$$

while $j \le N-3$, compute

$$u^{(k+1)}_{i,j} = r^2(S_2 w^{*}_{i,j-1} + S_3 w^{*}_{i,j} + S_2 w^{*}_{i,j+1} + S_1 w^{*}_{i,j+2}) + u^{(k)}_{i,j}$$

$$u^{(k+1)}_{i,j+1} = r^2(S_1 w^{*}_{i,j-1} + S_2 w^{*}_{i,j} + S_3 w^{*}_{i,j+1} + S_2 w^{*}_{i,j+2}) + u^{(k)}_{i,j+1}$$

$j = j + 2$

$$u^{(k+1)}_{i,N-1} = r^2(S_2 w^{*}_{i,N-1} + S_3 w^{*}_{i,N} + S_4 w^{*}_{i,N+1}) + u^{(k)}_{i,N-1}$$

$$u^{(k+1)}_{i,N} = r^2(S_1 w^{*}_{i,N-1} + S_2 w^{*}_{i,N} + dw^{*}_{i,N}) + u^{(k)}_{i,N}$$

$i = i + 1$.

Step 5: Repeat *Step* 2 to *Step* 4 until convergence is achieved.

From Algorithm 6.3.2-1, it can be deduced that for large N, the computational molecules for the CAGE-DG-2 scheme as follows.

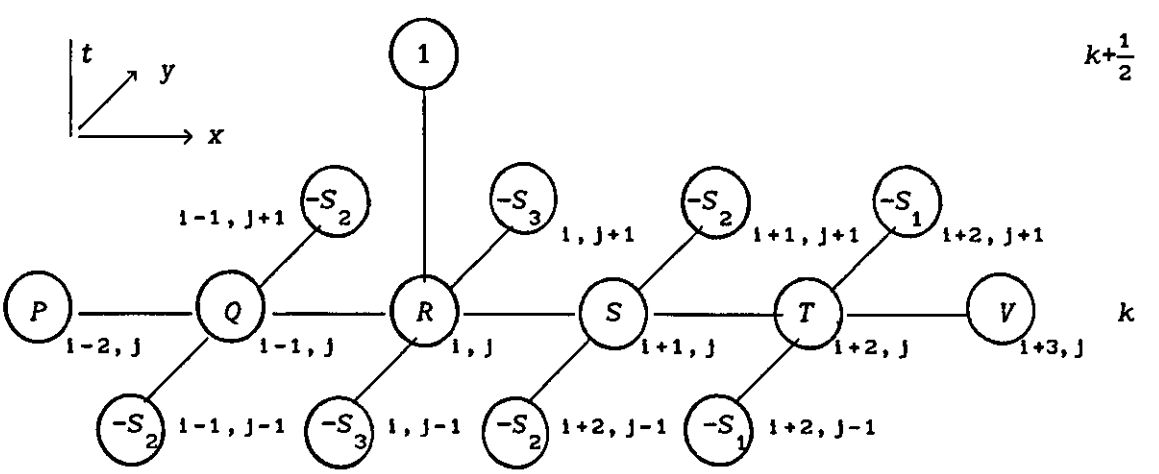


Figure 6.3.2-1: Computational molecules for computing $u^{(k+1/2)}_{i,j}$, for large N, Algorithm 6.3.2-1, the CAGE-DG-2 scheme.

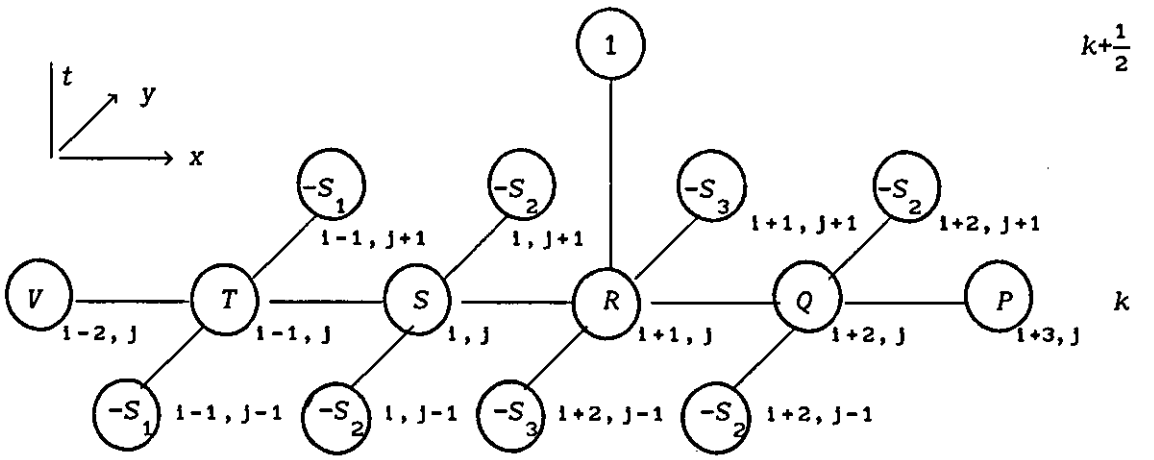


Figure 6.3.2-2: Computational molecules for computing $u_{i+1,j}^{(k+1/2)}$, for large N, Algorithm 6.3.2-1, the CAGE-DG-2 scheme.

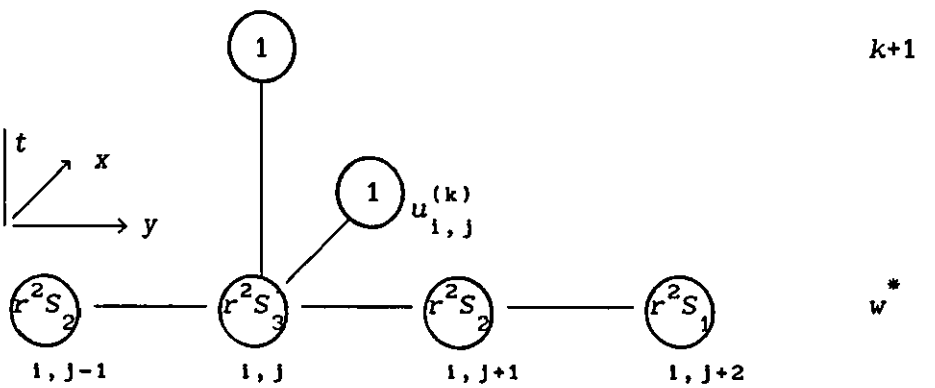


Figure 6.3.2-3: Computational molecules for computing $u_{i,j}^{(k+1)}$, for large N, Algorithm 6.3.2-1, the CAGE-DG-2 scheme.

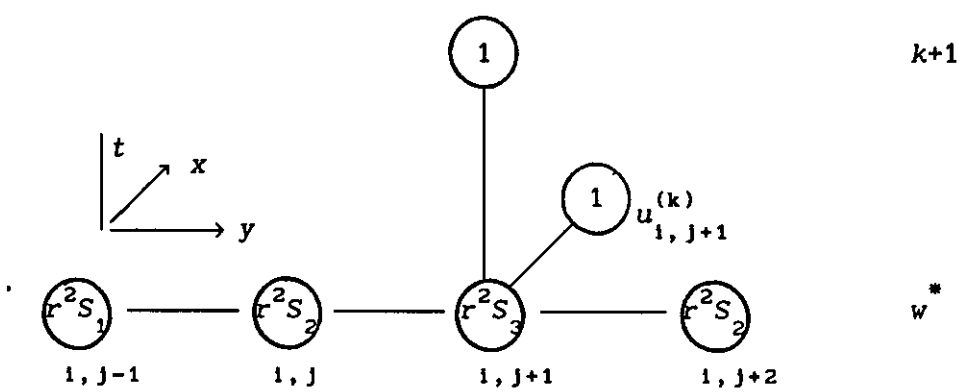


Figure 6.3.2-4: Computational molecules for computing $u_{i,j+1}^{(k+1)}$, for large N, Algorithm 6.3.2-1, the CAGE-DG-2 scheme.

We now extend the coupled form of the AGE-DG method to three dimensions, the CAGE-DG-3 scheme. Let us recall the AGE-DG-3 scheme, i.e., equations (6.1.3-21) - (6.1.3-26). In explicit form, we have

$$\mathbf{u}^{(k+1/6)} = [I - (rI + G_1)^{-1}A]\mathbf{u}^{(k)} + \omega(rI + G_1)^{-1}\mathbf{b} \tag{6.3.2-16}$$

$$\mathbf{u}^{(k+1/3)} = (rI + G_2)^{-1}[r\mathbf{u}^{(k+1/6)} + G_2\mathbf{u}^{(k)}] \tag{6.3.2-17}$$

$$\mathbf{u}_y^{(k+1/2)} = (rI + G_1)^{-1}[r\mathbf{u}_y^{(k+1/3)} + G_1\mathbf{u}_y^{(k)}] \tag{6.3.2-18}$$

$$\mathbf{u}_y^{(k+2/3)} = (rI + G_2)^{-1}[r\mathbf{u}_y^{(k+1/2)} + G_2\mathbf{u}_y^{(k)}] \tag{6.3.2-19}$$

$$\mathbf{u}_z^{(k+5/6)} = (rI + G_1)^{-1}[r\mathbf{u}_z^{(k+2/3)} + G_1\mathbf{u}_z^{(k)}] \tag{6.3.2-20}$$

$$\mathbf{u}_z^{(k+1)} = (rI + G_2)^{-1}[r\mathbf{u}_z^{(k+5/6)} + G_2\mathbf{u}_z^{(k)}] \tag{6.3.2-21}$$

In the coupled form, equations (6.3.2-16) - (6.3.2-21) can be combined explicitly to give a set of new equations, i.e., the CAGE-DG-3 scheme as

$$\mathbf{u}^{(k+1/3)} = [I - \omega r(rI + G_2)^{-1}(rI + G_1)^{-1}A]\mathbf{u}^{(k)} + \omega r(rI + G_2)^{-1}(rI + G_1)^{-1}\mathbf{b} \tag{6.3.2-22}$$

$$\mathbf{u}_y^{(k+2/3)} = r^2(rI + G_2)^{-1}(rI + G_1)^{-1}w_y^{*} + \mathbf{u}_y^{(k)} \tag{6.3.2-23}$$

$$\mathbf{u}_z^{(k+1)} = r^2(rI + G_2)^{-1}(rI + G_1)^{-1}w_z^{**} + \mathbf{u}_z^{(k)} \tag{6.3.2-24}$$

where $w_y^{*} = [\mathbf{u}_y^{(k+1/3)} - \mathbf{u}_y^{(k)}]$ and $w_z^{**} = [\mathbf{u}_z^{(k+2/3)} - \mathbf{u}_z^{(k)}]$.

Obviously, by choosing a similar pattern for the matrices G_1 and G_2 we have for the CAGE-DG-3 scheme, the computational molecules for equations (6.3.2-22) are expected to be similar to the ones derived in *Figures* 6.3.2-1 and 6.3.2-2 with the additional complexity of the z direction.

In equations (6.3.2-23) and (6.3.2-24), we would also expect its computational molecules to be given as *Figures* 6.3.2-3 and 6.3.2-4. The molecules for computing $u_{i,j}^{(k+1/3)}$ and $u_{i+1,j}^{(k+1/3)}$ can be illustrated as follows.

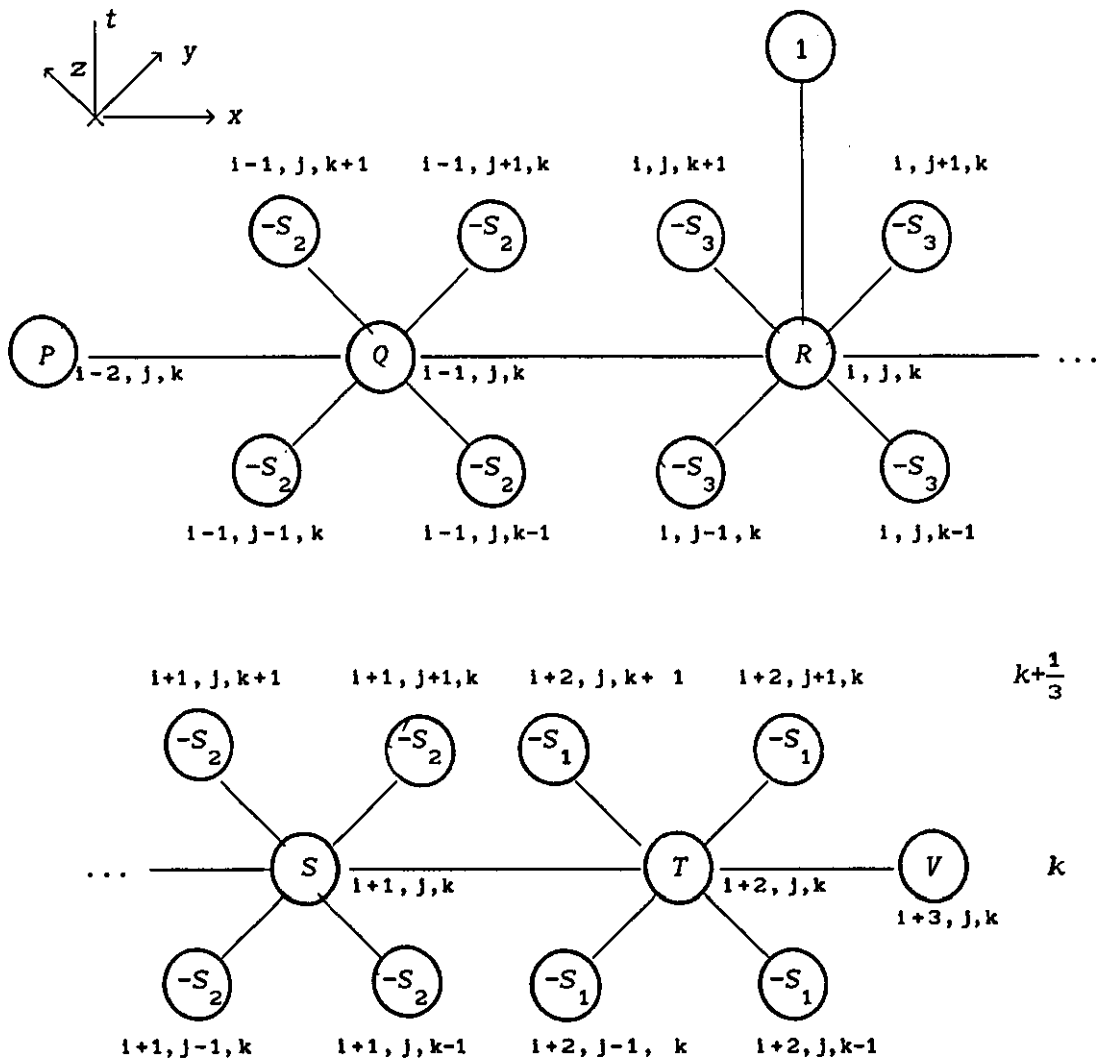
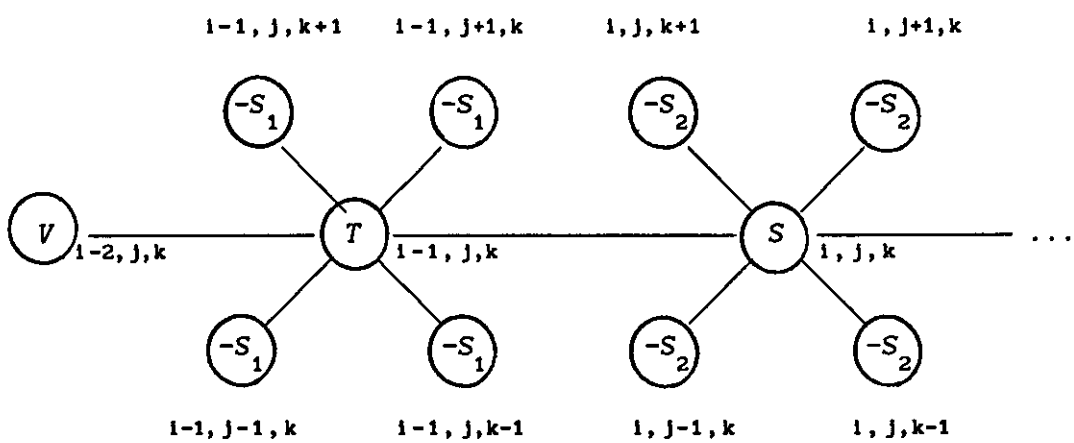


Figure 6.3.2-5: Computational molecules for computing $u_{i,j}^{(k+1/3)}$, for large N, the CAGE-DG-3 scheme.



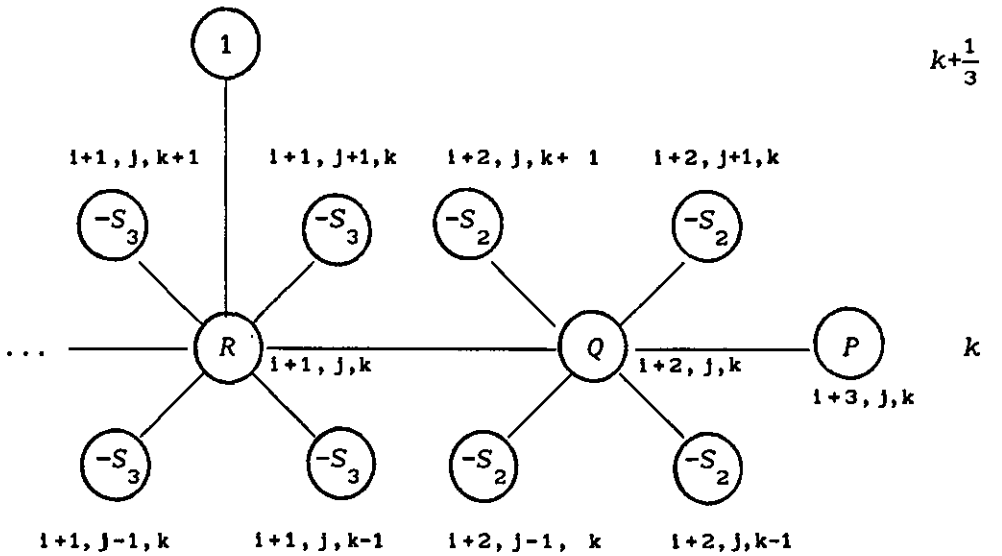


Figure 6.3.2-5: Computational molecules for computing $u_{1,j}^{(k+1/3)}$, for large N, the CAGE-DG-3 scheme.

6.3.3 The computational complexity

In this section, our concern is to find the best computational form which gives the fastest CPU time. In other words, we will determine the computational scheme that needs the least amount of mathematical operations in each iteration. *Table* 6.3.2-1 below summarises our findings from Sections 6.3.1 and 6.3.2.

Method	Multiplication	Addition	Overall
AGE-DG-2	$17(N-1)^2$	$15(N-1)^2$	$32(N-1)^2$
COMP-AGE-DG-2	$16(N-1)^2$	$18(N-1)^2$	$34(N-1)^2$
CAGE-DG-2	$12(N-3)^2$	$18(N-3)^2$	$30(N-3)^2$
AGE-DG-3	$25(N-1)^3$	$23(N-1)^3$	$48(N-1)^3$
COMP-AGE-DG-3	$24(N-1)^3$	$28(N-1)^3$	$52(N-1)^3$
CAGE-DG-3	$15(N-3)^3$	$30(N-3)^3$	$45(N-3)^3$

Table 6.3.2-1: The number of operations per iteration

The result indicates that there is a slight difference in the amount of work needed for each scheme. It appears that the COMP-AGE-DG schemes are less effective since the schemes takes more work (overall) when compared with the other schemes. Although the CAGE-DG schemes give the best performance for the computational work, symbolically, these schemes require more mathematical work to obtain. Thus, the best scheme is the standard AGE-DG-2 and AGE-DG-3 schemes.

6.4 Summary

The extension of the AGE-DG scheme for solving the two and three dimensional elliptic problems governed by different boundary conditions is the main concern in this chapter. The results obtained not only show that the extension is viable, but also, gives further clear evidence of the superiority of the AGE method over the SOR method. Although, the SOR method needs less computational effort, the simplicity of the AGE method, makes the method as powerful as SOR.

The focal point in the early sections was to ascertain the consistency of the method and that the method converges to the exact solution when solving the problems subject to Dirichlet boundary conditions. This has been shown in detail on how the convergence is achieved. The other interesting matter is the determination of the theoretical optimal parameter, r. By applying a simple assumption, this parameter is shown to be closely related to the theoretical optimal parameters for the one dimensional problem. As the theory is not available to date, this assumption can be regarded a good approximation for determining the experimental value of r.

The AGE method is also shown to perform well for the problems governed by periodic and Neumann boundary conditions. Although only one problem has been assigned to each boundary condition, the variation of the value of ρ in the Helmholtz equation in two dimensions is sufficient to study the characteristics of other similar equations. This might be seen when solving some problems governed by Dirichlet boundary conditions in the earlier sections.

The optimal single parameter r for both problems was also obtained experimentally, as no existing theory for this parameter would be determined. However, by using a rather simple assumption, the value of r can easily be obtained. This assumption is shown to have a similar pattern derived from the optimal single parameter for the one dimensional problem and can be considered as a good guide to obtain the experimental value of r.

This chapter ends with the study of other forms of presentation of the AGE method. The forms which we are concerned with is the COMP-AGE scheme and CAGE method. The CAGE method is shown to have less computational work and is suitable for parallel computation. However, if sequential computation is considered, the AGE-DG method should be competitive.

For the viability of the AGE method to solve the two dimensional problems governed by different boundary conditions, the simplicity of the method, an easy way to determine the optimal parameter r and the possibility of solution on parallel computers, indicates that the AGE method should be highly regarded as a substitute to the SOR method.

CHAPTER 7

MULTI-PARAMETER ITERATIVE METHODS FOR ELLIPTIC PDE's

7.1 The application of multi-parameters to iterative methods

The application of multi-parameters to the one dimensional problem has shown rather disappointing results as the matrix derived from the problem does not possess the commutative properties for the matrices G_1 and G_2. These setbacks, however, are compensated by the reasonably good results from a periodic boundary value problem where the properties do hold.

Young and Ehrlich [1971], have carried out some numerical experiments on the ADI method using more than one parameter for a variety of different regions for the two dimensional problem in order to test the applicability of the theoretical results in Section 5.1.1. The results obtained shows that there are some improvements in terms of the number of iterations and some regions where the results confirm that the theoretical rate of convergence $h^{-1/m}$, where m is the number of parameters.

In this chapter, we will again rely on the applicability of the theory discussed in Section 5.1.1 for these problems by using the AGE-DG scheme. We will also consider the heuristic search discussed in Section 5.1.2 and other methods (Section 5.1.4), to determine the possibility of the improvement of the rate of convergence for these problems.

The problem of interest is to solve the two dimensional elliptic partial differential equation (2D-PDE) governed by Dirichlet boundary conditions over some different regions given as follows. The regions are

(1) a unit square,

(2) a unit square with a square removed from the centre, and

(3) a right isosceles triangular region.

For the three dimensional elliptic PDE problem (3D-PDE), we confine our attention only to a problem subject to Dirichlet boundary conditions on the unit cube.

7.1.1 Convergence of the AGE-DG method for m (>1) parameters

Let us recall the AGE-DG-2 scheme in Section 6.1.3, i.e., equation (6.1.3-3) - (6.1.3-6). By combining these four equations, we have

$$\mathbf{u}^{(k+1)} = T_r \mathbf{u}^{(k)} + C \qquad (7.1.1\text{-}1)$$

where

$$T_r = I - \omega r^3 \prod_{i=4}^{1} (rI + G_i)^{-1} A \text{ and } C = \omega r^3 \prod_{i=4}^{1} (rI + G_i)^{-1} \mathbf{b} \qquad (7.1.1\text{-}2)$$

for $r > 0$, r is the iteration parameter. The spectral radius $\varphi(T_r)$ has been shown to be

$$\varphi(T_r) = \left| 1 - \frac{4\omega r^3 \lambda}{(r + \lambda)^4} \right| < 1, \qquad (7.1.1\text{-}3)$$

where λ is the largest eigenvalue, i.e., the spectral radius of the matrices G_1, G_2, G_3 and G_4. Thus, the AGE-DG-2 scheme for a single parameter is convergent.

Now, consider the nonstationary case, i.e., where the parameter r_k varies from iteration to iteration in a cyclic manner. If we now assume that the matrices G_1 and G_2 are commutative (the matrices G_3 and G_4 will take the form of the matrices G_1 and G_2 after columnwise ordering), then we have

$$\mathbf{u}^{(k+1)} = T_{k+1} \mathbf{u}^{(k)} + C \qquad (7.1.1\text{-}4)$$

where

$$T_{k+1} = I - \omega r_{k+1}^3 \prod_{i=4}^{1} (r_{k+1} I + G_i)^{-1} A \qquad (7.1.1\text{-}5)$$

and $C = \omega r_{k+1}^3 \prod_{i=4}^{1} (r_{k+1} I + G_i)^{-1} b$, for $r_{k+1} > 0$. (7.1.1-6)

Hence, the spectral radius, $\rho(T_{k+1})$ is given by

$$\rho(T_{k+1}) = \left| 1 - \frac{4\omega r_{k+1}^3 \lambda}{(r_{k+1} + \lambda)^4} \right| < 1, \qquad (7.1.1\text{-}7)$$

where λ is the largest eigenvalue, i.e. the spectral radius of the matrices G_1, G_2, G_3 and G_4. Hence, by (7.1.1-7), the AGE-DG-2 scheme for the nonstationary case is also convergent.

Similarly, recall the AGE-DG-3 scheme for solving the problem in three dimensions, i.e., equations (6.1.3-17) - (6.1.3-22). By combining these six equations, we have

$$u^{(k+1)} = T_r u^{(k)} + C \qquad (7.1.1\text{-}8)$$

where

$$T_r = I - \omega r^5 \prod_{i=6}^{1} (rI + G_i)^{-1} A \text{ and } C = \omega r^5 \prod_{i=6}^{1} (rI + G_i)^{-1} b \qquad (7.1.1\text{-}9)$$

for $r > 0$. The spectral radius $\rho(T_r)$ has been shown to be

$$\rho(T_r) = \left| 1 - \frac{4\omega r^5 \lambda}{(r + \lambda)^6} \right| < 1, \qquad (7.1.1\text{-}10)$$

where λ is the largest eigenvalue, i.e., the spectral radius of the matrices G_1, G_2, G_3, G_4, G_5 and G_6. Thus, the AGE-DG-3 scheme for a single parameter is also convergent.

It should be noticed that the matrices G_3 and G_4, G_5 and G_6 will take the form of the matrices G_1 and G_2 after reordering along the y and z axis respectively. We now apply the nonstationary case where the parameter r_k varies from iteration to iteration. If we assume that the matrices G_1 and G_2 commutative, then we have

$$u^{(k+1)} = T_{k+1}u^{(k)} + C \qquad (7.1.1\text{-}11)$$

where

$$T_{k+1} = I - \omega r_{k+1}^{5} \prod_{i=6}^{1}(r_{k+1}I + G_i)^{-1}A \qquad (7.1.1\text{-}12)$$

and $C = \omega r_{k+1}^{5} \prod_{i=6}^{1}(r_{k+1}I + G_i)^{-1}b$, for $r_{k+1} > 0$. (7.1.1-13)

Hence, the spectral radius, $\varphi(T_{k+1})$ is given by

$$\varphi(T_{k+1}) = \left| 1 - \frac{4\omega r_{k+1}^{5}\lambda}{(r_{k+1} + \lambda)^{6}} \right| < 1, \qquad (7.1.1\text{-}14)$$

where λ is the largest eigenvalue, i.e., the spectral radius of the matrices G_1, G_2, G_3, G_4, G_5 and G_6. Hence, by (7.1.1-14), the AGE-DG-3 scheme for the nonstationary case is also convergent.

7.1.2 The unit square region with a square removed from centre

A unit square region for the 2D-PDE, can be illustrated as

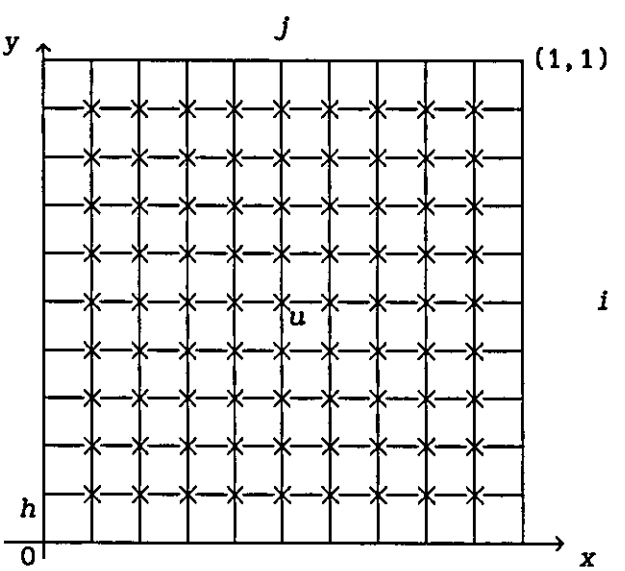


Figure 7.1.2-1: The number of nodes for $h = 1/10$.

with the grid points or nodes shown as -⋇- in *Figure* 7.1.2-1. The model problem is governed by Dirichlet boundary conditions.

Now, consider a square removed from the centre of this region, as illustrated in *Figure* 7.1.2-2.

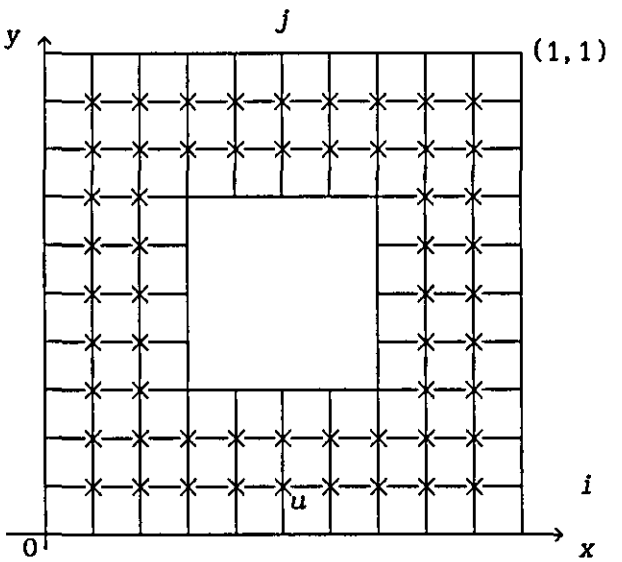


Figure 7.1.2-2: The number of nodes for h = 1/10.

The inner boundaries are also subjected to the Dirichlet boundary conditions. The grid points in the region is solved by partitioning the region as shown *Figure* 7.1.2-3,

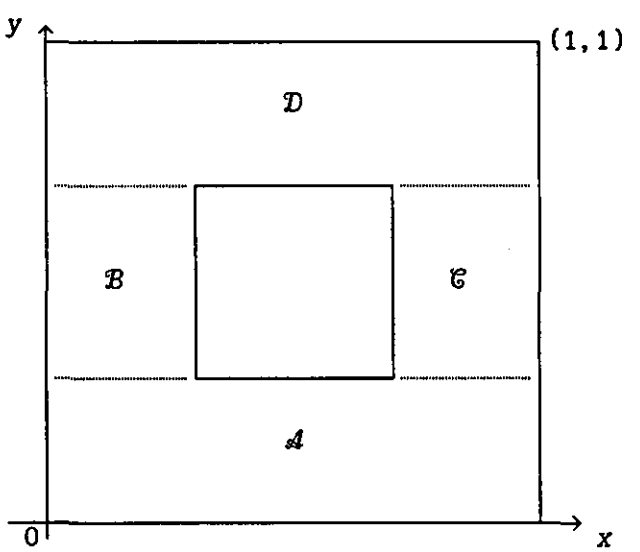


Figure 7.1.2-3: Partitioning the region.

First, we compute the points in the area A. Obviously, the points on the dotted line become the boundaries for the area A, B and C. The

values on these boundaries keep changing as the points along these boundaries have also to be evaluated. We next solve the points in the area $\mathcal{B}$ and $\mathcal{C}$ including the points along the dotted line. Finally, having calculated the value for the points along the common boundary $\mathcal{B}$, $\mathcal{C}$ and $\mathcal{D}$, all the points in the area $\mathcal{D}$ can be solved in similar way as in the area $\mathcal{A}$.

Since the square inside the region is fixed, one must be careful when choosing the mesh size, h. Let us consider the case when N is odd. For $h = 1/(N+1)$, if we choose $h = 1/10$, then the next mesh size must be 1/20, 1/40, 1/80, so on. In other words, the next interval must be half that of the previous interval. In *Figure* 7.1.2-2, we have $h = 1/10$.

It is quite difficult to derive a general matrix A which represent all values of h. But, by careful design we can derive each matrix A for each h. In this section, we will derive the matrix A for $h = 1/10$, whilst the algorithm is written for $h = 1/80$. As usual, we will apply the finite difference approximation to the Laplace equation to get the conventional five-point difference equation.

Let us recall *Figure* 7.1.2-2, where each point is denoted as $u_{i,j}$. The matrix A derived from *Figure* 7.1.2-2 can be written as follows:

$$A = \begin{bmatrix} B & -I & & & & & & & \\ -I & B & -C & & & & & & \\ & -C & D & -C & & & & & \\ & & -C & D & -C & & & & \\ & & & -C & D & -C & & & \\ & & & & -C & D & -C & & \\ & & & & & -C & D & -C & \\ & & & & & & -C & B & -I \\ & & & & & & & -I & B \end{bmatrix}_{9^2 X 9^2} \qquad (7.1.2\text{-}1)$$

where

$$B = \begin{bmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 4 & -1 \\ & & & -1 & 4 \end{bmatrix}_{9X9}, \qquad C = \begin{bmatrix} -1 & & & & \\ & -1 & & & \\ & & 0 & & \\ & & & \ddots & \\ & & & & 0 \\ & & & & & -1 \\ & & & & & & -1 \end{bmatrix}_{9X9}$$

and $$D = \begin{bmatrix} 4 & -1 & & & & \\ -1 & 4 & & & & \\ & & 0 & & & \\ & & & \ddots & & \\ & & & & 0 & \\ & & & & & 4 & -1 \\ & & & & & -1 & 4 \end{bmatrix}_{9X9} . \qquad (7.1.2\text{-}2)$$

The vector **u** is given by,

$$\mathbf{u} = [u_{1,1}, \ldots, u_{9,1}; u_{1,2}, \ldots, u_{9,2}; \ldots; u_{1,9}, \ldots, u_{9,9}]^T,$$

and the vector **b** is

$$\mathbf{b} = [b_{1,1}, \ldots, b_{9,1}; b_{1,2}, \ldots, b_{9,2}; b_{1,3}, b_{2,3}, b_{8,3}, b_{9,3}; \ldots;$$

$$b_{1,7}, b_{2,7}, b_{8,7}, b_{9,7}; b_{1,8}, \ldots, b_{9,8}; b_{1,9}, \ldots, b_{9,9}]^T.$$

By applying the AGE method, we consider the splitting of A into four submatrices, i.e.,

$$A = G_1 + G_2 + G_3 + G_4 \qquad (7.1.2\text{-}3)$$

where G_1, G_2, G_3 and G_4 are symmetric and positive definite.

Now, with A in (7.1.2-1), we have

$$(G_1 + G_2) = \begin{bmatrix} B_1 & & & & & & & & \\ & B_1 & & & & & & & \\ & & D_1 & & & & & & \\ & & & D_1 & & & & & \\ & & & & D_1 & & & & \\ & & & & & D_1 & & & \\ & & & & & & D_1 & & \\ & & & & & & & B_1 & \\ & & & & & & & & B_1 \end{bmatrix}_{9^2X9^2} \qquad (7.1.2\text{-}4)$$

where

$$B_1 = \begin{bmatrix} 2 & -1 & & & & & & & \\ -1 & 2 & -1 & & & & & & \\ & \ddots & \ddots & \ddots & & & & & \\ & & -1 & 2 & -1 & & & & \\ & & & -1 & 2 & & & & \end{bmatrix}_{9X9} \quad \text{and} \quad D_1 = \left[\begin{array}{cc|c|cc} 2 & -1 & & & \\ -1 & 2 & & & \\ \hline & & 0 & & \\ & & \ddots & & \\ \hline & & & 2 & -1 \\ & & & -1 & 2 \end{array}\right]_{9X9} . \quad (7.1.2\text{-}5)$$

By interchanging the direction from row to column, it can be shown that the matrix $(G_3 + G_4)$ has the form of $(G_1 + G_2)$ and vice-versa.

Let us consider the matrix G_1 as

$$G_1 = \begin{bmatrix} G'_1 & & & & & & & & \\ & G'_1 & & & & & & & \\ & & D'_1 & & & & & & \\ & & & D'_1 & & & & & \\ & & & & D'_1 & & & & \\ & & & & & D'_1 & & & \\ & & & & & & D'_1 & & \\ & & & & & & & G'_1 & \\ & & & & & & & & G'_1 \end{bmatrix}_{9^2X9^2}$$

where

$$G'_1 = \left[\begin{array}{cc|c|cc} 1 & -1 & & & \\ -1 & 1 & & & \\ \hline & & \ddots & & \\ \hline & & & 1 & -1 \\ & & & -1 & 1 \\ \hline & & & & 1 \end{array}\right]_{9x9} \quad D'_1 = \left[\begin{array}{cc|c|cc} 1 & -1 & & & \\ -1 & 1 & & & \\ \hline & & 0 & & \\ & & \ddots & & \\ \hline & & & 1 & -1 \\ & & & -1 & 1 \end{array}\right]_{9x9}$$

(7.1.2-6)

and the matrix G_2 as

$$G_2 = \begin{bmatrix} G'_2 & & & & & & & & \\ & G'_2 & & & & & & & \\ & & D'_2 & & & & & & \\ & & & D'_2 & & & & & \\ & & & & D'_2 & & & & \\ & & & & & D'_2 & & & \\ & & & & & & D'_2 & & \\ & & & & & & & G'_2 & \\ & & & & & & & & G'_2 \end{bmatrix}_{9^2X9^2}$$

where

$$G_2' = \begin{bmatrix} 1 & & & & & \\ & 1 & -1 & & & \\ & -1 & 1 & & & \\ & & & \ddots & & \\ & & & & 1 & -1 \\ & & & & -1 & 1 \end{bmatrix}_{9\times 9} \qquad D_2' = \begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 0 & & & \\ & & & \ddots & & \\ & & & & 1 & \\ & & & & & 1 \end{bmatrix}_{9\times 9}$$

(7.1.2-7)

From (7.1.2-6) and (7.1.2-7), it is clear that the matrices G_1 and G_2 consist of (2X2) block submatrices, symmetric and positive definite. Thus, for any iteration parameter, $r > 0$, $(rI + G_1)^{-1}$ and $(rI + G_2)^{-1}$ do exist. Hence, we can use the AGE method to solve the region with $N = 9$.

For other grid meshes, such as $h = 1/20$ or $1/40$, the linear system $A\mathbf{u} = \mathbf{b}$ can be derived in similar way. The Algorithm 7.1.2-1 is written for $h = 1/80$, i.e, $N = 79$, by using the AGE-DG scheme.

Algorithm 7.1.2-1: The square region with a square removed from the centre, Section 7.1.2.

Set $u_{i,j}^{(k)} = 0$, $i, j = 0, \ldots, N+1$, $\alpha = r + 1$, $d = 1/(\alpha^2 - 1)$,

$t = \alpha - 4\omega$, $s = \omega - 1$, $\alpha_1 = \alpha d$.

Step 1: To compute $\mathbf{u}^{(k+1/4)}$.

The area $\mathcal{A}$ in *Figure* 7.1.2-3.

Step 1.1: Set $i = 1$, $j = 1$, $M = N-2$, $K = 22$.

while $j \le K$, compute

while $i \le M$, compute

$$r_1 = \omega u_{i,j-1}^{(k)} + \omega u_{i-1,j}^{(k)} + t u_{i,j}^{(k)} + s u_{i+1,j}^{(k)} + \omega u_{i,j+1}^{(k)} + \omega b_{i,j}$$

$$r_2 = \omega u_{i+1,j-1}^{(k)} + s u_{i,j}^{(k)} + t u_{i+1,j}^{(k)} + \omega u_{i+2,j}^{(k)} + \omega u_{i+1,j+1}^{(k)}$$

$$+ \omega b_{i+1,j}$$

$$u_{i,j}^{(k+1/4)} = \alpha_1 r_1 + r_2 d, \quad u_{i+1,j}^{(k+1/4)} = r_1 d + \alpha_1 r_2$$

$i = i + 2$

$$u_{N,j}^{(k+1/4)} = (\omega u_{N,j-1}^{(k)} + \omega u_{N-1,j}^{(k)} + t u_{N,j}^{(k)} + \omega u_{N,j+1}^{(k)} + \omega b_{N,j})/\alpha$$

$j = j + 1$.

Step 1.2: Set $i = 1$, $j = 23$, $M = N-2$.

while $i \le M$, compute

$$r_1 = \omega u_{i,j-1}^{(k)} + \omega u_{i-1,j}^{(k)} + t u_{i,j}^{(k)} + s u_{i+1,j}^{(k)} + \omega u_{i,j+1}^{(k)}$$

$$+ \omega b_{i,j}$$

if $(23 < i < (N-22))$ *then* $r_1 = r_1 - \omega u_{i,j+1}^{(k)}$

$$r_2 = \omega u_{i+1,j-1}^{(k)} + s u_{i,j}^{(k)} + t u_{i+1,j}^{(k)} + \omega u_{i+2,j}^{(k)} + \omega u_{i+1,j+1}^{(k)}$$

$$+ \omega b_{i+1,j}$$

if $(23 < (i+1) < (N-22))$ *then* $r_2 = r_2 - \omega u_{i+1,j+1}^{(k)}$

$$u_{i,j}^{(k+1/4)} = \alpha_1 r_1 + r_2 d, \quad u_{i+1,j}^{(k+1/4)} = r_1 d + \alpha_1 r_2$$

$i = i + 2$

$$u_{N,j}^{(k+1/4)} = (\omega u_{N,j-1}^{(k)} + \omega u_{N-1,j}^{(k)} + t u_{N,j}^{(k)} + \omega u_{N,j+1}^{(k)} + \omega b_{N,j})/\alpha$$

The area $\mathcal{B}$ in *Figure* 7.1.2-3.

Step 1.3: Set $i = 1$, $j = 24$, $M = 21$, $K = N-23$.

while $j \le K$, compute

while $i \le M$, compute

$$r_1 = \omega u_{i,j-1}^{(k)} + \omega u_{i-1,j}^{(k)} + t u_{i,j}^{(k)} + s u_{i+1,j}^{(k)} + \omega u_{i,j+1}^{(k)}$$

$$+ \omega b_{i,j}$$

$$r_2 = \omega u_{i+1,j-1}^{(k)} + s u_{i,j}^{(k)} + t u_{i+1,j}^{(k)} + \omega u_{i+2,j}^{(k)} + \omega u_{i+1,j+1}^{(k)}$$

$$+ \omega b_{i+1,j}$$

$$u_{1,j}^{(k+1/4)} = \alpha_1 r_1 + r_2 d, \quad u_{1+1,j}^{(k+1/4)} = r_1 d + \alpha_1 r_2$$

$i = i + 2$

$$u_{N,j}^{(k+1/4)} = (\omega u_{N,j-1}^{(k)} + \omega u_{N-1,j}^{(k)} + t u_{N,j}^{(k)} + \omega u_{N,j+1}^{(k)} + \omega b_{N,j})/\alpha$$

$j = j + 2.$

The area $\mathcal{C}$ in *Figure* 7.1.2-3.

Step 1.4: Repeat *Step* 1.3 for $i = 57$, $j = 24$, $M = 77$, $K = N-23$.

The area $\mathcal{D}$ in *Figure* 7.1.2-3.

Step 1.5: Repeat *Step* 1.2 for $i = 1$, $j = N-22$, $M = N-2$.

Step 1.6: Repeat *Step* 1.1 for $i = 1$, $j = 58$, $M = N-2$. $K = N$.

Step 2: To compute $\mathbf{u}^{(k+1/2)}$. $i = 2$, $j = 1$.

The area $\mathcal{A}$ in *Figure* 7.1.2-3.

Step 2.1: Set $i = 2$, $j = 1$, $M = N-1$, $K = 23$.

while $j \le K$, compute

$$u_{i-1,j}^{(k+1/2)} = (r u_{i-1,j}^{(k+1/4)} + u_{i-1,j}^{(k)})/\alpha$$

while $i \le M$, compute

$$r_1 = r u_{i,j}^{(k+1/4)} + u_{i,j}^{(k)} - u_{i+1,j}^{(k)}$$

$$r_2 = r u_{i+1,j}^{(k+1/4)} - u_{i,j}^{(k)} + u_{i+1,j}^{(k)}$$

$$u_{i,j}^{(k+1/2)} = \alpha_1 r_1 + r_2 d, \quad u_{i+1,j}^{(k+1/2)} = r_1 d + \alpha_1 r_2$$

$i = i + 2$

$j = j + 1$

The area $\mathcal{B}$ in *Figure* 7.1.2-3.

Step 2.2: Repeat *Step* 2.1 for $i = 2$, $j = 24$, $M = N-1$, $K = N-23$.

The area $\mathcal{C}$ in *Figure* 7.1.2-3.

Step 2.3: Repeat *Step* 2.1 for $i = 58$, $j = 24$, $M = N-1$, $K = N-23$.

The area $\mathcal{D}$ in *Figure* 7.1.2-3.

Step 2.4: Repeat *Step* 2.1 for $i = 2$, $j = N-22$, $M = N-1$, $K = N$.

Step 3: To compute $\mathbf{u}^{(k+3/4)}$. (Change in direction).

The area $\mathcal{A}$ in *Figure* 7.1.2-3.

Step 3.1: Set for $i = 1$, $j = 1$, $M = 23$, $K = N-2$.

while $i \le M$, compute

while $j \le K$, compute

$$r_1 = ru_{i,j}^{(k+1/2)} + u_{i,j}^{(k)} - u_{i,j+1}^{(k)}$$

$$r_2 = ru_{i,j+1}^{(k+1/2)} - u_{i,j}^{(k)} + u_{i,j+1}^{(k)}$$

$$u_{i,j}^{(k+3/4)} = \alpha_1 r_1 + r_2 d, \quad u_{i,j+1}^{(k+3/4)} = r_1 d + \alpha_1 r_2$$

$$j = j + 2$$

$$u_{i,N}^{(k+3/4)} = (ru_{i,N}^{(k+1/2)} + u_{i,N}^{(k)})/\alpha$$

$$i = i + 1.$$

The area $\mathcal{B}$ in *Figure* 7.1.2-3.

Step 3.2: Repeat *Step* 3.1 for $i = 24$, $j = 1$, $M = N-23$, $K = 21$.

The area $\mathcal{C}$ in *Figure* 7.1.2-3.

Step 3.3: Repeat *Step* 3.1 for $i = 24$, $j = 57$, $M = N-23$, $K = N-2$.

The area $\mathcal{D}$ in *Figure* 7.1.2-3.

Step 3.4: Repeat *Step* 3.1 for $i = N-22$, $j = 1$, $M = N$, $K = N-2$.

Step 4: To compute $\mathbf{u}^{(k+1)}$. $i = 1$, $j = 2$. (change in direction).

The area $\mathcal{A}$ in *Figure* 7.1.2-3.

Step 4.1: Set for $i = 1$, $j = 2$, $M = 23$, $K = N-1$.

while $i \le M$, compute

$$u_{i,j-1}^{(k+1)} = (ru_{i,j-1}^{(k+3/4)} + u_{i,j-1}^{(k)})/\alpha$$

while $j \le K$, compute

$$r_1 = ru_{i,j}^{(k+3/4)} + u_{i,j}^{(k)} - u_{i,j+1}^{(k)}$$

$$r_2 = ru_{i,j+1}^{(k+3/4)} - u_{i,j}^{(k)} + u_{i,j+1}^{(k)}$$

$$u_{i,j}^{(k+1)} = \alpha_1 r_1 + r_2 d, \quad u_{i,j+1}^{(k+1)} = r_1 d + \alpha_1 r_2$$

$j = j + 2$

$i = i + 1.$

The area $\mathcal{B}$ in *Figure* 7.1.2-3.

Step 4.2: Repeat *Step* 4.1 for $i = 24$, $j = 2$, $M = N-23$, $K = 22$.

The area $\mathcal{C}$ in *Figure* 7.1.2-3.

Step 4.3: Repeat *Step* 4.1 for $i = 24$, $j = 58$, $M = N-23$, $K = N-1$.

The area $\mathcal{D}$ in *Figure* 7.1.2-3.

Step 4.4: Repeat *Step* 4.1 for $i = N-22$, $j = 2$, $M = N$, $K = N-1$.

Step 5: Repeat *Step* 1 to *Step* 4 until convergence is achieved.

7.1.3 The right isosceles triangular region

Let us consider the right isosceles triangular region given in *Figure* 7.1.3-1.

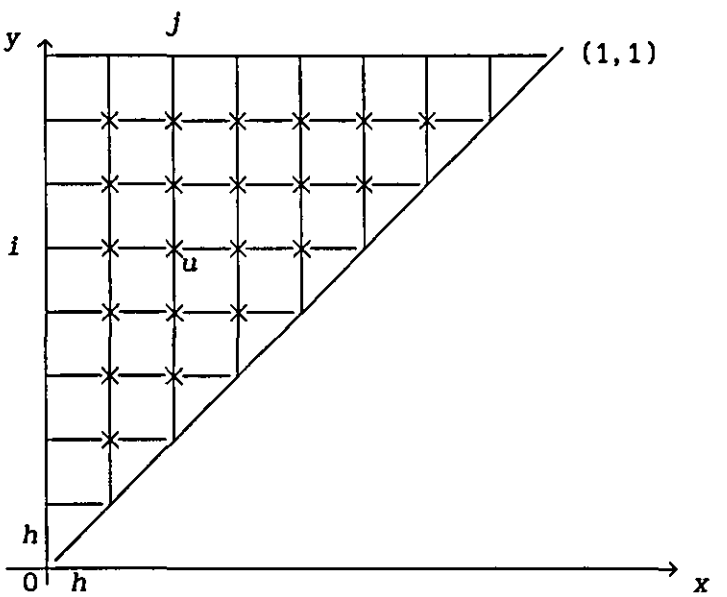


Figure 7.1.3-1 : The Right Isosceles Triangular Region, where $h = 1/8$.

This right isosceles triangular has two equal sides of length unity and the boundaries are prescribed on each side and the hypotenuse. The

grid points, as usual, are denoted as $u_{i,j}$. It is obvious that the number of grid points on the line are alternately even and odd, along the y axis, and vice versa. This characteristic gives a rather unusual matrix A compared to the one derived from a square region.

By applying the usual five-point finite difference approximation, yields the linear system $A\mathbf{u} = \mathbf{b}$, with A given as in (7.1.3-2), and the respective vector $\mathbf{u}$ and $\mathbf{b}$ are

$$\mathbf{u} = [u_{12}, \ldots, u_{17}; u_{23}, \ldots, u_{27}; u_{34}, \ldots, u_{37}; u_{45}, u_{46}, u_{47}; \ldots; u_{56}, u_{57}; u_{67}]^T$$

and $\mathbf{b} = [b_{12}, 0, \ldots, 0, b_{17}; b_{23}, 0, \ldots, 0, b_{27}; b_{34}, 0, 0, b_{37}; \ldots; b_{45}, 0, b_{47}; b_{56}, b_{17}; b_{67}]^T$.

By applying the AGE method, we consider a splitting of the matrix A in (7.1.3-2) in four submatrices, i.e,

$$A = G_1 + G_2 + G_3 + G_4 \qquad (7.1.3\text{-}1)$$

where G_1, G_2, G_3 and G_4 are symmetric and positive definite.

$A =$

4	-1						-1																	
-1	4	-1						-1																
	-1	4	-1						-1															
		-1	4	-1						-1														
			-1	4	-1						-1													
				-1	4							-1												
-1							4	-1					-1											
	-1						-1	4	-1					-1										
		-1						-1	4	-1					-1									
			-1						-1	4	-1					-1								
				-1						-1	4						-1							
							-1					4	-1											
												-1	4	-1										
													-1	4	-1									
														-1	4									
																4	-1							
																-1	4	-1						
																	-1	4	-1					
																		-1	4					
																				4	-1			
																				-1	4	-1		
																					-1	4		

(7.1.3-2)

From (7.1.3-2), we have $A_1 = G_1 + G_2$,

$$A_1 = \left[\begin{array}{c|c|c|c|c|c}
\begin{matrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & -1 & 2 & -1 & \\ & & & -1 & 2 & -1 \\ & & & & -1 & 2 \end{matrix} & & & & & \\
\hline
 & \begin{matrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & -1 & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{matrix} & & & & \\
\hline
 & & \begin{matrix} 2 & -1 & & \\ -1 & 2 & -1 & \\ & -1 & 2 & -1 \\ & & -1 & 2 \end{matrix} & & & \\
\hline
 & & & \begin{matrix} 2 & -1 & \\ -1 & 2 & -1 \\ & -1 & 2 \end{matrix} & & \\
\hline
 & & & & \begin{matrix} 2 & -1 \\ -1 & 2 \end{matrix} & \\
\hline
 & & & & & 2
\end{array}\right]$$

(7.1.3-2)

From (7.1.3-2), we can show that the matrices G_1 and G_2 consist of (2X2) block submatrices, symmetric and positive definite. The matrices G_1 and G_2 are given as follows:

$$G_1 = \begin{bmatrix} B_1 & & & & & \\ & B_2 & & & & \\ & & B_3 & & & \\ & & & B_4 & & \\ & & & & B_5 & \\ & & & & & 1 \end{bmatrix}, \quad G_2 = \begin{bmatrix} C_1 & & & & & \\ & C_2 & & & & \\ & & C_3 & & & \\ & & & C_4 & & \\ & & & & C_5 & \\ & & & & & 1 \end{bmatrix} \qquad (7.1.3\text{-}3)$$

where

$$B_1 = \left[\begin{array}{cc|cc|cc} 1 & -1 & & & & \\ -1 & 1 & & & & \\ \hline & & 1 & -1 & & \\ & & -1 & 1 & & \\ \hline & & & & 1 & -1 \\ & & & & -1 & 1 \end{array}\right], \quad B_2 = \left[\begin{array}{cc|cc|c} 1 & -1 & & & \\ -1 & 1 & & & \\ \hline & & 1 & -1 & \\ & & -1 & 1 & \\ \hline & & & & 1 \end{array}\right],$$

$$B_3 = \begin{bmatrix} 1 & -1 & & \\ -1 & 1 & & \\ & & 1 & -1 \\ & & -1 & 1 \end{bmatrix}, \quad B_4 = \begin{bmatrix} 1 & -1 & & \\ -1 & 1 & & \\ & & & \\ & & & 1 \end{bmatrix}, \quad B_5 = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix},$$

$$C_1 = \begin{bmatrix} 1 & & & & \\ & 1 & -1 & & \\ & -1 & 1 & & \\ & & & 1 & -1 \\ & & & -1 & 1 \end{bmatrix}, \quad C_2 = \begin{bmatrix} 1 & & & \\ & 1 & -1 & \\ & -1 & 1 & \\ & & & 1 & -1 \\ & & & -1 & 1 \end{bmatrix},$$

$$C_3 = \begin{bmatrix} 1 & & & \\ & 1 & -1 & \\ & -1 & 1 & \\ & & & 1 \end{bmatrix}, \quad C_3 = \begin{bmatrix} 1 & & \\ & 1 & -1 \\ & -1 & 1 \end{bmatrix} \text{ and } C_5 = \begin{bmatrix} 1 & \\ & 1 \end{bmatrix}.$$

Evidently, for any $r > 0$, the matrices $(rI + G_1)^{-1}$ and $(rI + G_2)^{-1}$ do exist, where r is the iteration parameter. Hence, we can use the AGE method, i.e., the AGE-DG scheme, to solve the right isosceles triangular region. First, to derive the matrix $C_r = (rI + G_1) - \omega A$.

For $N = 7$, we then have

$$C_r = \begin{bmatrix}
t & s & & & & & & & & & & & & & & & & & & & & & & & & \\
s & t & s & & & & & \omega & & & & & & & & & & & & & & & & & & \\
 & s & t & s & & & & & \omega & & & & & & & & & & & & & & & & & \\
 & & s & t & s & & & & & \omega & & & & & & & & & & & & & & & & \\
 & & & s & t & s & & & & & \omega & & & & & & & & & & & & & & & \\
 & & & & s & t & & & & & & \omega & & & & & & & & & & & & & & \\
\hline
 & \omega & & & & & & t & s & & & & & & & & & & & & & & & & & \\
 & & \omega & & & & & s & t & s & & & & \omega & & & & & & & & & & & & \\
 & & & \omega & & & & & s & t & s & & & & \omega & & & & & & & & & & & \\
 & & & & \omega & & & & & s & t & s & & & & \omega & & & & & & & & & & \\
 & & & & & \omega & & & & & s & t & & & & & \omega & & & & & & & & & \\
\hline
 & & & & & & & \omega & & & & & t & s & & & & & & & & & & & & \\
 & & & & & & & & \omega & & & & s & t & s & & & & & & & & & & & \\
 & & & & & & & & & \omega & & & & s & t & s & & & & & & & & & & \\
 & & & & & & & & & & \omega & & & & s & t & & & & & & & & & & \\
\hline
 & & & & & & & & & & & & \omega & & & & t & s & & & & & & & & \\
 & & & & & & & & & & & & & \omega & & & s & t & s & & & & & & & \\
 & & & & & & & & & & & & & & \omega & & & s & t & & & & & & & \\
\hline
 & & & & & & & & & & & & & & & & & \omega & & t & s & & & & & \\
 & & & & & & & & & & & & & & & & & & \omega & s & t & \omega & & & & \\
\hline
 & & & & & & & & & & & & & & & & & & & & \omega & t & & & &
\end{bmatrix}$$

where $t = \alpha - 4\omega$, $s = \omega - 1$, with $\alpha = r + 1$.

In general, for the case when N is odd, the algorithm for solving the region can be written as follows.

Algorithm 7.1.3-1: The right isosceles triangular region.

Set $u_{i,j}^{(k)} = 0,\ i,\ j = 0,\ \ldots,\ N+1,\ \alpha = r + 1,\ d = 1/(\alpha^2 - 1),$

$t = \alpha - 4\omega,\ s = \omega - 1,\ \alpha_1 = \alpha d.$

Step 1: To compute $\mathbf{u}^{(k+1/4)}$.

Even number of points on a line. Set $i = 1,\ j = i + 1.$

while $i \le N-2$, compute

while $j \le N$, compute

$$r_1 = \omega u_{i-1,j}^{(k)} + \omega u_{i,j-1}^{(k)} + t u_{i,j}^{(k)} + s u_{i,j+1}^{(k)} + \omega u_{i+1,j}^{(k)} + \omega b_{i,j}$$

$$r_2 = \omega u_{i-1,j+1}^{(k)} + s u_{i,j}^{(k)} + t u_{i,j+1}^{(k)} + \omega u_{i,j+2}^{(k)} + \omega u_{i+1,j+1}^{(k)} + \omega b_{i,j+1}$$

$$u_{i,j}^{(k+1/4)} = \alpha_1 r_1 + r_2 d, \quad u_{i,j+1}^{(k+1/4)} = r_1 d + \alpha_1 r_2$$

$i = i + 2$

$j = i + 1$

Odd number of points on a line. Set $i = 2,\ j = 3.$

while $i \le N-1$, compute

while $j \le N-2$, compute

$$r_1 = \omega u_{i-1,j}^{(k)} + \omega u_{i,j-1}^{(k)} + t u_{i,j}^{(k)} + s u_{i,j+1}^{(k)} + \omega u_{i+1,j}^{(k)} + \omega b_{i,j}$$

$$r_2 = \omega u_{i-1,j+1}^{(k)} + s u_{i,j}^{(k)} + t u_{i,j+1}^{(k)} + \omega u_{i,j+2}^{(k)} + \omega u_{i+1,j+1}^{(k)} + \omega b_{i,j+1}$$

$$u_{i,j}^{(k+1/4)} = \alpha_1 r_1 + r_2 d, \quad u_{i,j+1}^{(k+1/4)} = r_1 d + \alpha_1 r_2$$

$j = j + 2$

$$u_{i,N}^{(k+1/4)} = (\omega u_{i-1,N}^{(k)} + \omega u_{i,N-1}^{(k)} + t u_{i,N}^{(k)} + \omega u_{i+1,N}^{(k)} + \omega b_{i,N})/\alpha$$

$i = i + 2$.

Step 2: To compute $\mathbf{u}^{(k+1/2)}$. Set $i = 1$.

Even number of points on a line.

while $i \le N-2$, compute

$k = i + 1$, $j = k + 1$

$$u_{i,k}^{(k+1/2)} = (r u_{i,k}^{(k+1/4)} + u_{i,k}^{(k)})/\alpha$$

while $j \le N-2$, compute

$$r_1 = r u_{i,j}^{(k+1/4)} + u_{i,j}^{(k)} - u_{i,j+1}^{(k)}$$

$$r_2 = r u_{i,j+1}^{(k+1/4)} - u_{i,j}^{(k)} + u_{i,j+1}^{(k)}$$

$$u_{i,j}^{(k+1/2)} = \alpha_1 r_1 + r_2 d, \quad u_{i,j+1}^{(k+1/2)} = r_1 d + \alpha_1 r_2$$

$j = j + 2$

$$u_{i,N}^{(k+1/2)} = (r u_{i,N}^{(k+1/4)} + u_{i,N}^{(k)})/\alpha$$

$i = i + 1$.

Odd number of points on a line. Set $i = 2$.

while $i \le N-1$, compute

$k = i + 1$, $j = k + 1$

$$u_{i,k}^{(k+1/2)} = (r u_{i,k}^{(k+1/4)} + u_{i,k}^{(k)})/\alpha$$

while $j \le N-1$, compute

$$r_1 = r u_{i,j}^{(k+1/4)} + u_{i,j}^{(k)} - u_{i,j+1}^{(k)}$$

$$r_2 = r u_{i,j+1}^{(k+1/4)} - u_{i,j}^{(k)} + u_{i,j+1}^{(k)}$$

$$u_{i,j}^{(k+1/2)} = \alpha_1 r_1 + r_2 d, \quad u_{i,j+1}^{(k+1/2)} = r_1 d + \alpha_1 r_2$$

$j = j + 2$

$i = i + 1$.

Step 3: To compute $\mathbf{u}^{(k+3/4)}$. Set $m = 2$, $k = 1$.

Even number of points on a line.

while $m \le N-1$, compute

$i = m$, $j = m + 1$

while $k \le m/2$, compute

$$r_1 = ru_{i,j}^{(k+1/2)} + u_{i,j}^{(k)} - u_{i-1,j}^{(k)}$$

$$r_2 = ru_{i-1,j}^{(k+1/2)} - u_{i,j}^{(k)} + u_{i-1,j}^{(k)}$$

$$u_{i,j}^{(k+3/4)} = \alpha_1 r_1 + r_2 d, \quad u_{i-1,j}^{(k+3/4)} = r_1 d + \alpha_1 r_2$$

$i = i - 2$

$k = k + 1$

$m = m + 2$.

Odd number of points on a line. Set $j = 2$, $k = 1$.

while $j \le N-1$, compute

$$u_{i,j}^{(k+3/4)} = (ru_{i,j}^{(k+1/2)} + u_{i,j}^{(k)})/\alpha$$

$i = j - 1$, $m = (j - 2)/2$

while $k \le m$, compute

$$r_1 = ru_{i,j}^{(k+1/2)} + u_{i,j}^{(k)} - u_{i-1,j}^{(k)}$$

$$r_2 = ru_{i-1,j}^{(k+1/2)} - u_{i,j}^{(k)} + u_{i-1,j}^{(k)}$$

$$u_{i,j}^{(k+3/4)} = \alpha_1 r_1 + r_2 d, \quad u_{i-1,j}^{(k+3/4)} = r_1 d + \alpha_1 r_2$$

$i = i - 2$

$k = k + 1$

$j = j + 2$.

Step 4: To compute $\mathbf{u}^{(k+1)}$. Set $m = 2$, $k = 1$.

Even number of points on a line.

while $m \le N-1$, compute

$i = m$, $j = m + 1$

$$u_{i,j}^{(k+1)} = (ru_{i,j}^{(k+3/4)} + u_{i,j}^{(k)})/\alpha$$

$i = i - 1$

while $k \le (m - 2)/2$, compute

$$r_1 = ru_{i,j}^{(k+3/4)} + u_{i,j}^{(k)} - u_{i-1,j}^{(k)}$$

$$r_2 = ru_{i-1,j}^{(k+3/4)} - u_{i,j}^{(k)} + u_{i-1,j}^{(k)}$$

$$u_{i,j}^{(k+1)} = \alpha_1 r_1 + r_2 d, \quad u_{i-1,j}^{(k+1)} = r_1 d + \alpha_1 r_2$$

$i = i - 2$

$k = k + 1$

$$u_{i,j}^{(k+1)} = (ru_{i,j}^{(k+3/4)} + u_{i,j}^{(k)})/\alpha$$

$m = m + 2.$

Odd number of points on a line. Set $m = 1$, $k = 1$.

while $m \le N-2$, compute

$i = m$, $j = m + 1$

$$u_{i,j}^{(k+1)} = (ru_{i,j}^{(k+3/4)} + u_{i,j}^{(k)})/\alpha$$

$i = i - 1$

while $k \le m/2$, compute

$$r_1 = ru_{i,j}^{(k+3/4)} + u_{i,j}^{(k)} - u_{i-1,j}^{(k)}$$

$$r_2 = ru_{i-1,j}^{(k+3/4)} - u_{i,j}^{(k)} + u_{i-1,j}^{(k)}$$

$$u_{i,j}^{(k+1)} = \alpha_1 r_1 + r_2 d, \quad u_{i-1,j}^{(k+1)} = r_1 d + \alpha_1 r_2$$

$i = i - 2$

$k = k + 1$

$m = m + 2.$

Step 5: Repeat *Step* 1 t o *Step* 4 until convergence is achieved.

7.1.4 Experimental results

Numerical results presented here are concerned with the application of multi-parameters on different regions for the two dimensional problem governed by the Dirichlet boundary conditions and the three dimensional problem in a unit cube governed by the Dirichlet boundary conditions.

Again, the existing ADI multi-parameter formulae has been tested and it has been found that these parameters are not suitable for these problems where the splitting is based on the AGE method as there is no gain in terms of number of iterations. Other hypotheses also do not work well as no improvement is shown concerning the rate of convergence. Thus, the results obtained were purely based on the heuristic search for two parameters.

Problem 1 - The Poisson Equation

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = -2, \qquad 0 \le x,y \le 1,$$

The exact solution is $U(x,y) = \sinh \pi x \sin \pi y + x(1 - x)$.

(A) The unit square region, see *Figure* 7.1.4-1.

The boundary conditions are given by

$$U(0,y) = 0,\ U(1,y) = \sinh \pi x \sin \pi y, \qquad 0 \le y \le 1,$$

$$U(x,0) = U(x,1) = x(1 - x), \qquad 0 \le x \le 1.$$

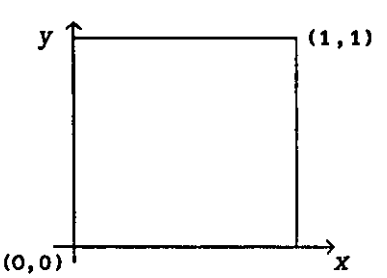


Figure 7.1.4-1: The unit square region

The results are tabulated in *Table* 7.1.4-1.

h^{-1}	Single Parameter		2 parameter - Heuristic		
	r	iter	r_1	r_2	iter
10	0.97 - 1.08	27	0.59	6.17	19
20	0.58 - 0.59	56	0.30	14.53	36
40	0.34	116	0.15	27.98	64
80	0.18	228	0.08	57.00	112

Table 7.1.4-1: Number of iterations for Problem 1 - Region (A)

(B) The unit square with a square removed from the centre, see *Figure* 7.1.4-2. The unit square removed is $\frac{4}{10} \times \frac{4}{10}$. The outer boundary conditions are as given in the unit square region.

The inner boundary conditions are given by

$$U(x,y) = \sinh \pi x \sin \pi y + x(1 - x).$$

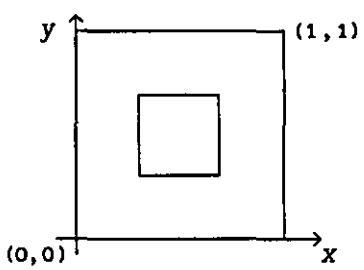


Figure 7.1.4-2: The unit square region with a square removed from the centre

The results are tabulated in *Table* 7.1.4-2.

h^{-1}	Single Parameter		2 parameter - Heuristic		
	r	iter	r_1	r_2	iter
10	1.73 - 1.08	12	0.99	3.59	10
20	1.12 - 0.59	25	0.59	4.86	18
40	0.65 - 0.69	54	0.32	10.75	34
80	0.41 - 0.42	121	0.20	33.00	78

Table 7.1.4-2: Number of iterations for Problem 1 - Region (B)

(C) The right isosceles triangular region, see *Figure* 7.1.4-3.

The boundary conditions on the hypotenuse are

$$U(x,y) = \sinh \pi x \sin \pi y + x(1 - x),$$

and on the other sides of length unity,

$$U(0,y) = 0, \quad 0 \le y \le 1, \quad U(x,1) = x(1 - x), \quad 0 \le x \le 1.$$

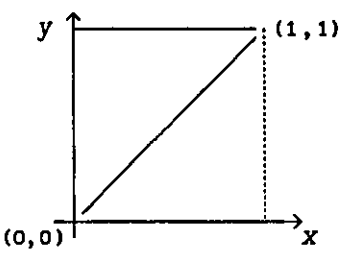


Figure 7.1.4-3: The right isosceles triangular region

The results are tabulated in *Table* 7.1.4-3.

h^{-1}	Single Parameter		2 parameter - Heuristic		
	r	iter	r_1	r_2	iter
10	1.67 - 1.71	16	0.96	3.93	13
20	1.10	36	0.53	2.15	25
40	0.79	102	0.30	1.31	60
80	0.78	373	0.28	1.28	202

Table 7.1.4-3: Number of iterations for Problem 1 - Region (C)

Problem 2 - The three dimensional problem

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} + \frac{\partial^2 U}{\partial y^2} = 0, \qquad 0 \le x,y,z \le 1,$$

governed by the Dirichlet boundary conditions

$$U(x,y,0) = U(x,y,1) = 0 \qquad 0 \le x,y \le 1,$$

$$U(x,0,z) = U(x,1,z) = \sin \pi x \sin \pi z \qquad 0 \le x,z \le 1,$$

$$U(0,y,z) = U(1,y,z) = 0 \qquad 0 \le y,z \le 1.$$

The exact solution is

$$U(x,y,z) = \operatorname{sech} \frac{\pi}{\sqrt{2}} \sin \pi x \cosh \{\sqrt{2}\pi(y - 0.5)\} \sin \pi z.$$

The results are tabulated in *Table* 7.1.4-4.

h^{-1}	Single Parameter		2 parameter - Heuristic		
	r	iter	r_1	r_2	iter
12	1.22 - 1.25	26	0.62	3.01	18
14	1.08 - 1.11	31	0.55	5.96	20
16	0.98	35	0.49	8.80	22
18	0.88 - 0.89	40	0.44	10.41	24
20	0.81	44	0.39	12.25	26

Table 7.1.4-4: Number of iterations for Problem 2

Varga [1962], has shown that in the ADI-PR method, if a certain single iteration parameter $r > 0$, is used for all iterations, then the number of iterations varies as h^{-1}. This work shows that this result holds well for the square and its related area, i.e., the region with a square removed from the centre but not for the triangular region. In this region, the error propagation waves cannot dissipate themselves as quickly over the domain as in the other regions. As a result, more iterations are needed at a larger number of points which shows that convergence is slower.

Young and Ehrlich [1955], proved that with the ADI-PR method, the number of iterations was proportional to $h^{-1/m}$, for m = 1, 2 and 3. In their work, this seems to hold fairly well for the square and its subset but not for the triangular region. Nevertheless, for this region, the number of iterations still appeared to vary like $h^{-\alpha}$, where $0 < \alpha < 1$.

Due to the similarity of the AGE-DG scheme to the ADI-PR method and from the theory with m = 2, one would expect the number of iterations to be proportional to $h^{-1/2}$. However, the results shows that the rate of convergence for the first two regions varies like $h^{-\alpha}$, where $0 < \alpha < 1$ with $\alpha = 1^-$, as shown in the *Table* 7.1.4-5 which follows:

Problem-region	Rate of Convergence	
	Single Parameter	two Parameter
Problem 1 - A	1.0285	0.8879
Problem 1 - B	1.1113	0.9808
Problem 1 - C	1.5132	1.3136
Problem 2	0.9600	0.6727

Table 7.1.4-5: Comparison of the rate of convergence

The triangular region with m = 2, however, does not fall in this category but gives a slightly better result compared with the single parameter. Although the change of the rate of convergence is only small, the gain in the number of iterations may be considered quite substantial when N is large.

For the three dimensional problem, i.e., Problem 2, the method performs slightly better when compared to the two dimensional problem. This can be seen from *Table* 7.1.4-5 and *Table* 7.1.4-6 which show a considerable gain in the number of iterations.

Since G_1 and G_2 are the (2X2) block submatrices, then it was quite sufficient to investigate only a small number of parameters, i.e., 2 or 3. However, as the slope for the two parameter case does not change according to $h^{-1/2}$, and moreover the heuristic search for the three parameters is too cumbersome, then it is appropriate at the moment to consider only the 2 parameter case.

These results obtained seem disappointing perhaps due to the following reasons. In our case, the matrices G_1 and G_2 are not commutative for all regions. Moreover, the block size (2x2) is too small compared to the large block sizes of the ADI-PR method which is often of size (20x20) or bigger. The small block size coupled with their multiplicity will cause all the eigenvalues to be clustered and

not distinctive as in the ADI-PR method. As a result, this may develop round-off error growth for the large parameter values.

Apart from the good results where the number of iterations are reduced to almost half in every case, the only setback is the doubtful stability when solving for a larger number of points. Experimentally, these results show that the first parameter is close to zero so that the second parameter will be some distance apart. Thus, if the 2 parameter case is going to be considered, one must take a full consideration when solving larger number of points.

7.2 The multi-parameter case for the semi-iterative methods

The multi-parameter case for the semi-iterative method for one dimensional problems has been discussed in detail in Chapter 5. In this section, we will now extend the application for the two and three dimensional problems. Since the results for the one dimensional problem gave no gain in terms of the number of iterations required for convergence, then we would expect a similar pattern when solving a problem with higher dimension (as in Section 7.1), for the case of more than one parameter.

7.2.1 The Richardson method

Let us recall the Richardson Method, i.e., equation (5.2.1-2)

$$\mathbf{u}^{(k+1)} = (I - \omega_{k+1}N^{-1}A)\mathbf{u}^{(k)} + \omega_{k+1}N^{-1}\mathbf{b} \qquad (7.2.1\text{-}1)$$

where ω_{k+1} is defined in (5.2.1-6).

When applying the method in one dimension, we can simply use the CAGE algorithm, i.e., Algorithm 4.2.2-2 with the substitution of $2r\omega_{k+1}$

instead of $2r$. Since the CAGE method for higher dimensions is more tedious to use, then, we will consider using the AGE-DG scheme written as a single equation for solving the two and three dimensional problem.

Thus, by combining the four equations from the AGE-DG-2 scheme in equations (6.1.3-3) - (6.1.3-6) into the single equation, we have

$$\mathbf{u}^{(k+1)} = (I - N^{-1}A)\mathbf{u}^{(k)} + N^{-1}\mathbf{b}$$

$$= \mathbf{u}^{(k)} + N^{-1}(\mathbf{b} - A\mathbf{u}^{(k)}) \qquad (7.2.1\text{-}2)$$

where

$$N^{-1} = 2r^3\omega_{k+1} \prod_{i=4}^{1}(rI + G_i)^{-1}. \qquad (7.2.1\text{-}3)$$

We will consider only the case where $r = r_k$.

By interchanging from row to column ordering or vice-versa, we can reduce the iteration to four steps involving the matrices G_1 and G_2. Thus, a step by step calculation can be performed as follows:

Step 1: Compute $\mathbf{v}^{(k)} = (rI + G_1)^{-1}(\mathbf{b} - A\mathbf{u}^{(k)})$

Step 2: Compute $\mathbf{w}^{(k)} = (rI + G_2)^{-1}\mathbf{v}^{(k)}$

Step 3: Compute $\mathbf{v}_c^{(k)} = (rI + G_1)^{-1}\mathbf{w}_c^{(k)}$

Step 4: Compute $\mathbf{u}_c^{(k+1)} = \mathbf{u}_c^{(k)} + 2r^3\omega_{k+1}(rI + G_2)^{-1}\mathbf{v}_c^{(k)}$,

where c denotes the columnwise ordering.

Now, let us consider the matrices A, G_1 and G_2 as in (6.1.1-11) and (6.1.1-15) respectively. Thus, the algorithm for the Richardson method for the two dimensional (2D) problem can be written as follows:

Algorithm 7.2.1-1: The Richardson Method for the 2D problem.

Set $u_{i,j}^{(k)} = 0$, $i, j = 0, \ldots, N+1$, $\alpha = r + g$, $d = 1/(\alpha^2 - 1)$, $\alpha_1 = \alpha d$,

ω_k, $k = 1, N$, from the relation (5.2.1-6).

Step 1. To compute $\mathbf{v}^{(k)} = (rI + G_1)^{-1}(\mathbf{b} - A\mathbf{u}^{(k)})$.

Set $i, j = 1$.

while $j \le N$, compute

while $i \le N-2$, compute

$$r_1 = u^{(k)}_{i,j-1} + u^{(k)}_{i-1,j} - 4gu^{(k)}_{i,j} + u^{(k)}_{i+1,j} + u^{(k)}_{i,j+1} + b_{i,j}$$

$$r_2 = u^{(k)}_{i+1,j-1} + u^{(k)}_{i,j} - 4gu^{(k)}_{i+1,j} + u^{(k)}_{i+2,j} + u^{(k)}_{i+1,j+1} + b_{i+1,j}$$

$$v^{(k)}_{i,j} = \alpha_1 r_1 + r_2 d, \quad v^{(k)}_{i+1,j} = r_1 d + \alpha_1 r_2$$

$i = i + 2$

$$v^{(k)}_{N,j} = (u^{(k)}_{N,j-1} + u^{(k)}_{N-1,j} - 4gu^{(k)}_{N,j} + u^{(k)}_{N,j+1} + b_{N,j})/\alpha$$

$j = j + 1$.

Step 2. To compute $\mathbf{w}^{(k)} = (rI + G_2)^{-1}\mathbf{v}^{(k)}$.

Set $i = 2$, $j = 1$.

while $j \le N$, compute

$$w^{(k)}_{1,j} = v^{(k)}_{1,j}/\alpha$$

while $i \le N-1$, compute

$$w^{(k)}_{i,j} = \alpha_1 v^{(k)}_{i,j} + dv^{(k)}_{i+1,j}$$

$$w^{(k)}_{i+1,j} = dv^{(k)}_{i,j} + \alpha_1 v^{(k)}_{i+1,j}$$

$i = i + 2$

$j = j + 1$.

Step 3. To compute $\mathbf{v}^{(k)}_c = (rI + G_1)^{-1}\mathbf{w}^{(k)}_c$.

Set $i, j = 1$. (Change in direction)

while $i \le N$, compute

while $j \le N-2$, compute

$$v^{(k)}_{i,j} = \alpha_1 w^{(k)}_{i,j} + dw^{(k)}_{i,j+1}$$

$$v^{(k)}_{i,j+1} = dw^{(k)}_{i,j} + \alpha_1 w^{(k)}_{i,j+1}$$

$j = j + 2$

$w_{i,N}^{(k)} = w_{i,N}^{(k)}/\alpha$

$i = i + 1.$

Step 4: To compute $\mathbf{u}_c^{(k+1)} = \mathbf{u}_c^{(k)} + 2r^3\omega_{k+1}(rI + G_2)^{-1}\mathbf{v}_c^{(k)}$,

Set $i = 1$, $j = 2$. (Change in direction)

while $i \leq N$, compute

$u_{i,1}^{(k+1)} = u_{i,1}^{(k)} + 2r^3\omega_{k+1}v_{i,1}^{(k)}/\alpha$

while $j \leq N-2$, compute

$u_{i,j}^{(k+1)} = u_{i,j}^{(k)} + 2r^3\omega_{k+1}d(\alpha v_{i,j}^{(k)} + v_{i,j+1}^{(k)})$

$u_{i,j+1}^{(k+1)} = u_{i,j+1}^{(k)} + 2r^3\omega_{k+1}d(v_{i,j}^{(k)} + \alpha v_{i,j+1}^{(k)})$.

$j = j + 2$

$i = i + 1.$

Step 5. Repeat *Step* 1 to *Step* 4 until convergence is achieved.

Now, for the three dimensional (3D) problem, let us recall the AGE-DG-3 scheme, i.e., equations (6.1.3-17) - (6.1.3-22). By combining these six equations, we then have the single equation,

$$\mathbf{u}^{(k+1)} = (I - N^{-1}A)\mathbf{u}^{(k)} + N^{-1}\mathbf{b}$$

$$= \mathbf{u}^{(k)} + N^{-1}(\mathbf{b} - A\mathbf{u}^{(k)}) \qquad (7.2.1\text{-}4)$$

where

$$N^{-1} = 2r^5\omega_{k+1}\prod_{i=6}^{1}(rI + G_i)^{-1}. \qquad (7.2.1\text{-}5)$$

Again, we will consider only the case where $r = r_k$.

By interchanging the order of the direction, i.e., from x to y, from x to z and vice-versa, we proceed as before and only consider the matrices G_1 and G_2. Thus, the whole calculation can be performed as follows:

Step 1: Compute $\mathbf{v}^{(k)} = (rI + G_1)^{-1}(\mathbf{b} - A\mathbf{u}^{(k)})$

Step 2: Compute $\mathbf{w}^{(k)} = (rI + G_2)^{-1}\mathbf{v}^{(k)}$

Step 3: Compute $\mathbf{v}_x^{(k)} = (rI + G_1)^{-1}\mathbf{w}_x^{(k)}$

Step 4: Compute $\mathbf{w}_x^{(k)} = (rI + G_2)^{-1}\mathbf{v}_x^{(k)}$

Step 5: Compute $\mathbf{v}_y^{(k)} = (rI + G_1)^{-1}\mathbf{w}_y^{(k)}$

Step 6: Compute $\mathbf{u}_y^{(k+1)} = \mathbf{u}_y^{(k)} + 2r^5\omega_{k+1}(rI + G_2)^{-1}\mathbf{v}_y^{(k)}$,

where x and y denote the change of direction along the x-axis and y-axis respectively.

By considering the matrices A in (6.1.2-6), G_1 and G_2 in (6.1.2-10) we can write the algorithm for the Richardson method for the 3D problem.

Algorithm 7.2.1-2: The Richardson Method for the 3D problem.

Set $u_{i,j,k}^{(k)} = 0$, $i, j, k = 0, \ldots, N+1$, $\alpha = r + g$, $d = 1/(\alpha^2 - 1)$,

$\alpha_1 = \alpha d$, ω_k, $k = 1, N$, from the relation (5.2.1-6).

Step 1: To compute $\mathbf{v}^{(k)} = (rI + G_1)^{-1}(\mathbf{b} - A\mathbf{u}^{(k)})$. Set $i, j, k = 1$.

while $k \le N$, compute

while $j \le N$, compute

while $i \le N-2$, compute

$$r_1 = u_{i,j,k-1}^{(k)} + u_{i,j-1,k}^{(k)} + u_{i-1,j,k}^{(k)} - 6gu_{i,j,k}^{(k)} + u_{i+1,j,k}^{(k)} + u_{i,j+1,k}^{(k)} + u_{i,j,k+1}^{(k)} + b_{i,j,k}$$

$$r_2 = u_{i+1,j,k-1}^{(k)} + u_{i+1,j-1,k}^{(k)} + u_{i,j,k}^{(k)} - 6gu_{i+1,j,k}^{(k)} + u_{i+2,j,k}^{(k)} + u_{i+1,j+1,k}^{(k)} + u_{i+1,j,k+1}^{(k)} + b_{i+1,j,k}$$

$$v_{i,j,k}^{(k)} = \alpha_1 r_1 + r_2 d, \quad v_{i+1,j,k}^{(k)} = r_1 d + \alpha_1 r_2$$

$i = i + 2$

$$v^{(k)}_{N,j,k} = (u^{(k)}_{N,j,k-1} + u^{(k)}_{N,j-1,k} + u^{(k)}_{N-1,j,k} - 6gu^{(k)}_{N,j,k} + u^{(k)}_{N,j+1,k} + u^{(k)}_{N,j,k+1} + b_{N,j,k})/\alpha$$

$j = j + 1$

$k = k + 1.$

Step 2: To compute $\mathbf{w}^{(k)} = (rI + G_2)^{-1}\mathbf{v}^{(k)}$. Set $i = 2$, j, $k = 1$.

while $k \leq N$, compute

while $j \leq N$, compute

$$w^{(k)}_{1,j,k} = v^{(k)}_{1,j,k}/\alpha$$

while $i \leq N-1$, compute

$$w^{(k)}_{i,j,k} = \alpha_1 v^{(k)}_{i,j,k} + dv^{(k)}_{i+1,j,k}$$

$$w^{(k)}_{i+1,j,k} = dv^{(k)}_{i,j,k} + \alpha_1 v^{(k)}_{i+1,j,k}$$

$i = i + 2$

$j = j + 1$

$k = k + 1.$

Step 3: To compute $\mathbf{v}^{(k)}_x = (rI + G_1)^{-1}\mathbf{w}^{(k)}_x$. Set i, j, $k = 1$.

(Change direction).

while $k \leq N$, compute

while $i \leq N$, compute

while $j \leq N-2$, compute

$$v^{(k)}_{i,j,k} = \alpha_1 w^{(k)}_{i,j,k} + dw^{(k)}_{i,j+1,k}$$

$$v^{(k)}_{i,j+1,k} = dw^{(k)}_{i,j,k} + \alpha_1 w^{(k)}_{i,j+1,k}$$

$j = j + 2$

$$v^{(k)}_{i,N,k} = w^{(k)}_{i,N,k}/\alpha$$

$i = i + 1$

$k = k + 1.$

Step 4: To compute $w_x^{(k)} = (rI + G_2)^{-1} v_x^{(k)}$.

Set $i = 1$, $j = 2$, $k = 1$.

(Change in direction).

while $k \le N$, compute

while $i \le N$, compute

$$w_{i,1,k}^{(k)} = v_{i,1,k}^{(k)}/\alpha$$

while $j \le N-1$, compute

$$w_{i,j,k}^{(k)} = \alpha_1 v_{i,j,k}^{(k)} + dv_{i,j+1,k}^{(k)}$$

$$w_{i,j+1,k}^{(k)} = dv_{i,j,k}^{(k)} + \alpha_1 v_{i,j+1,k}^{(k)}$$

$j = j + 2$

$i = i + 1$

$k = k + 1$.

Step 5: To compute $v_y^{(k)} = (rI + G_1)^{-1} w_y^{(k)}$.

Set i, j, $k = 1$.

(Change in direction).

while $i \le N$, compute

while $j \le N$, compute

while $k \le N-2$, compute

$$v_{i,j,k}^{(k)} = \alpha_1 w_{i,j,k}^{(k)} + dw_{i,j,k+1}^{(k)}$$

$$v_{i,j,k+1}^{(k)} = dw_{i,j,k}^{(k)} + \alpha_1 w_{i,j,k+1}^{(k)}$$

$k = k + 2$

$$v_{i,j,N}^{(k)} = w_{i,j,N}^{(k)}/\alpha$$

$j = j + 1$

$i = i + 1$.

Step 6: To compute $u_y^{(k+1)} = u_y^{(k)} + 2r^5 \omega_{k+1} (rI + G_2)^{-1} v_y^{(k)}$.

Set i, $j = 1$, $k = 2$. (Change in direction)

while $i \le N$, compute

while $j \le N$, compute

$$u_{i,j,1}^{(k+1)} = u_{i,j,1}^{(k)} + 2r^5\omega_{k+1}v_{i,j,1}^{(k)}/\alpha$$

while $k \le N-1$, compute

$$u_{i,j,k}^{(k+1)} = u_{i,j,k}^{(k)} + 2r^5\omega_{k+1}d(\alpha v_{i,j,k}^{(k)} + v_{i,j,k+1}^{(k)})$$

$$u_{i,j,k+1}^{(k+1)} = u_{i,j,k+1}^{(k)} + 2r^5\omega_{k+1}d(v_{i,j,k}^{(k)} + \alpha v_{i,j,k+1}^{(k)})$$

$k = k + 2$

$j = j + 1$

$i = i + 1$.

Step 7: Repeat *Step* 1 to *Step* 6 until convergence is achieved.

7.2.2 The Chebyshev semi-iterative method

Let us recall the Chebyshev semi-iterative method, i.e., equations (5.2.2-1) and (5.2.2-2) which are used to solve the one dimensional problem. This method can also be applied to solve 2D and 3D problems.

We rewrite equations (5.2.2-1) and (5.2.2-2) as follows:

$$\mathbf{u}^{(1)} = \omega_0[\mathbf{u}^{(0)} + N^{-1}(\mathbf{b} - A\mathbf{u}^{(0)})], \qquad (7.2.2\text{-}1)$$

and $$\mathbf{u}^{(k+1)} = \omega_k[\mathbf{u}^{(k)} + N^{-1}(\mathbf{b} - A\mathbf{u}^{(k)})] + (1 - \omega_k)\mathbf{u}^{(k-1)}$$

$$k = 1, 2, \ldots \qquad (7.2.2\text{-}2)$$

with the respective matrix N^{-1} for the 2D and 3D problems is in the form of (7.2.1-3) and (7.2.1-5).

Unlike in the one dimensional problem, the CAGE algorithm for the Chebyshev method to solve the 2D and 3D problems is quite difficult to be implemented. Instead, we will use the step by step calculation as it has been shown in the Richardson method. These calculations have to be split into two parts.

First, to compute the vector $\mathbf{u}^{(1)}$ and then the routine calculations to obtain the vector $\mathbf{u}^{(k+1)}$. By interchanging the direction from row to column, once again, we only need to consider the matrices G_1 and G_2.

Obviously, the computational work for $\mathbf{u}^{(k)} + N^{-1}(\mathbf{b} - A\mathbf{u}^{(k)})$ is similar to the ones in the Richardson method except the multiplication of $2r\omega_{k+1}$. Thus, the algorithm for the chebyshev semi-iterative method for solving the 2D and 3D problems may be written based on the Algorithm 7.2.1-1 and 7.2.1-2 respectively with some amendments to the existing algorithms.

7.2.3 The computational complexity

Obviously, the Richardson and Chebyshev semi-iterative methods need just a little more computational effort compared to the AGE-DG scheme. In the Richardson method, the only extra calculation is ω_{k+1} for each point. Thus, for a large N, we may expect an extra N^2 and N^3 work for the respective 2D and 3D problems for every iteration.

The chebyshev need a slightly more computational work. Apart from the multiplication of ω_k, the term $(1-\omega_k)\mathbf{u}^{(k-1)}$ need one addition and one multiplication for every point at each iteration. Thus, for large N, the extra work needed is N^2 additions and $2N^2$ multiplications for the 2D problem, and N^3 additions and $2N^3$ multiplications for the 2D problem.

7.2.4 Experimental results

The Problem 1 in square region and Problem 2 given in Section 7.1.5 were investigated further to obtain some improvements in terms of the number of iterations.

The theoretical ω_k discussed in Sections 5.2.1 seems to be unsuitable for problems where the splitting of matrix A is according to the AGE method. Experimentally, these ω_k do not give any improvement concerning the rate of convergence. Thus, the results for the AGE-Richardson method obtained in this section were based on the ω_k which were determined heuristically.

For the AGE-Chebyshev method, the Chebyshev parameters given in Section 5.2.2 seem to fit well and it is obvious that λ_{max} is in the interval [0,1], i.e., for the method to converge.

Problem 1 - The Poisson Equation

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = -2, \qquad 0 \le x,y \le 1,$$

The boundary conditions are given by

$$U(0,y) = 0,\ U(1,y) = \sinh \pi x \sin \pi y, \qquad 0 \le y \le 1,$$

$$U(x,0) = U(x,1) = x(1-x), \qquad 0 \le x \le 1.$$

The exact solution is

$$U(x,y) = \sinh \pi x \sin \pi y + x(1-x).$$

The results are tabulated in *Tables* 7.2.4-1 and 7.2.4-2.

h^{-1}	AGE-DG-2	AGE-Richardson						
	1 par	2 parameter (par)			3 parameter (par)			
	iter	r	ω	iter	r	ω_1	ω_2	iter
10	27	1.02-1.08	1.25	21	1.14-1.17	1.0	2.0	17
20	56	0.58-0.61	1.19	48	0.57-0.58	0.7	1.7	43
40	116	0.34	1.11	107	0.33	0.6	1.6	95
80	228	0.18	1.06	218	0.17	0.5	1.5	205

Table 7.2.4-1: Number of iterations for Problem 1

h^{-1}	AGE-DG-2		AGE-Chebyshev		
	r	iter	r	λ_{max}	iter
10	0.97 - 1.08	27	1.10 - 1.16	0.65	19
20	0.58 - 0.59	56	0.59 - 0.62	0.62	44
40	0.34	116	0.35	0.59	98
80	0.18	228	0.18	0.46	210

Table 7.2.4-2: Number of iterations for Problem 1

Problem 2 - The three dimensional problem

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} + \frac{\partial^2 U}{\partial z^2} = 0, \qquad 0 \le x, y, z \le 1,$$

governed by the Dirichlet boundary conditions

$$U(x,y,0) = U(x,y,1) = 0 \qquad 0 \le x, y \le 1,$$

$$U(x,0,z) = U(x,1,z) = \sin \pi x \sin \pi z \qquad 0 \le x, z \le 1,$$

$$U(0,y,z) = U(1,y,z) = 0 \qquad 0 \le y, z \le 1.$$

The exact solution is

$$U(x,y,z) = \operatorname{sech} \frac{\pi}{\sqrt{2}} \sin \pi x \cosh \{\sqrt{2}\pi(y - 0.5)\} \sin \pi z.$$

The results are tabulated in *Tables* 7.2.4-3 and 7.2.4-4.

h^{-1}	AGE-DG-3	AGE-Richardson						
	1 par	2 parameter (par)			3 parameter (par)			
	iter	r	ω	iter	r	ω_1	ω_2	iter
10	21	1.42-1.53	1.36	16	1.86-1.90	1.1	2.5	13
12	26	1.24-1.30	1.33	20	1.24-1.49	1.0	2.2	17
14	31	1.11-1.13	1.30	24	1.05-1.21	0.9	2.0	21
16	35	1.00-1.01	1.29	28	0.97-1.01	0.8	2.0	23
18	40	0.91	1.28	32	0.86-0.92	0.8	2.0	27
20	44	0.82-0.84	1.26	37	0.80	0.8	2.0	29

Table 7.2.4-3: Number of iterations for Problem 2

h^{-1}	AGE-DG-3		AGE-Chebyshev		
	r	iter	r	λ_{max}	iter
10	1.42	21	1.91 - 1.93	0.72	15
12	1.22 - 1.25	26	1.31 - 1.35	0.68	18
14	1.08 - 1.11	31	1.15 - 1.18	0.68	21
16	0.98	35	1.03 - 1.05	0.71	24
18	0.88 - 0.89	40	0.94 - 0.95	0.72	27
20	0.81	44	0.83 - 0.90	0.73	32

Table 7.2.4-4: Number of iterations for Problem 2

The results obtained show that a faster convergence can be achieved when using the AGE-Richardson and AGE-Chebyshev methods for solving the two and three dimensional problems. These improvements compensate the disappointing results when solving the one dimensional problem by using the same method. This gains agree with the arguments by Gourlay [1968] that the convergence can be accelerated by using these methods.

Tables 7.2.4-1 and 7.4.2-3 show that the AGE-Richardson method gives better convergence when more than one parameter is used. However, the more parameters that are involved, the more difficult is the search of the parameters which has to be performed. Although, the results for the two parameter case (ω_1 and ω_2) are better than the single parameter ω, the difficulties in obtaining these parameters, forces the single parameter results to be considered. Moreover, the results deteriorate at larger N.

We now compare the performance of the (single ω) AGE-Richardson method and the multi ω of the AGE-Chebyshev method. *Tables* 7.2.4-2 and 7.2.4-4 show the impressive results for the AGE-Chebyshev method in terms of the number of iterations. The simplicity in obtaining the parameters ω_{k+1} given in Section 5.2.2, give further arguments that this method should be regarded as viable for the multi-parameter case.

It is obvious that if the sequence of ω_{k+1} converges to 1, then the AGE-Chebyshev method will converge to the AGE-DG scheme. Moreover, for this sequence of parameters, λ_{max} can be shown to lie in the interval [0,1]. By varying this λ_{max}, the optimal number of iterations for a specified number of points can be determined quite easily.

Thus, for this simplicity and better convergence, the AGE-Chebyshev method can be considered for the application to the multi-parameter formula for the two and three dimensional problems.

7.3 The Explicit Alternating Direction (EAD) method

Let us consider the splitting of $A = H + V$. Then, the ADI-PR method for solving the two dimensional problem is given by

$$(rI + H)\mathbf{u}^{(k+1/2)} = (rI - V)\mathbf{u}^{(k)} + \mathbf{b} \qquad (7.3\text{-}1)$$

$$(rI + V)\mathbf{u}^{(k+1)} = (rI - H)\mathbf{u}^{(k+1/2)} + \mathbf{b} \qquad (7.3\text{-}2)$$

for any iteration parameter, $r > 0$ and where H and V are the symmetric and positive definite tridiagonal matrices.

By using a similar technique, we will now derive the EAD method.

7.3.1 The derivation of the EAD method

Let us consider further,

$$A_1 = (rI + H) = G_1 + G_2, \qquad (7.3.1\text{-}1)$$

and $$A_2 = (rI + V) = G_3 + G_4 \qquad (7.3.1\text{-}2)$$

which gives

$$A_1\mathbf{u}^{k+1/2} = \mathbf{b}_1, \qquad (7.3.1\text{-}3)$$

where $\mathbf{b}_1 = (rI - V)\mathbf{u}^k + \mathbf{b}$, and

$$A_2 u^{k+1} = b_2, \tag{7.3.1-4}$$

where $b_2 = (rI - H)u^k + b$.

By using the AGE-PR(1) scheme, for any $s > 0$, we can solve equation (7.3.1-3) as a pair (A)

$$(sI + G_1)u^{p+1/2} = (sI - G_2)u^p + b_1 \tag{7.3.1-5}$$

$$(sI + G_2)u^{k+1/2} = (sI - G_1)u^{p+1/2} + b_1 \tag{7.3.1-6}$$

and equation (7.3.1-4) as a pair (B)

$$(sI + G_3)u^{p+1/2} = (sI - G_4)u^p + b_2 \tag{7.3.1-7}$$

$$(sI + G_4)u^{k+1} = (sI - G_3)u^{p+1/2} + b_2. \tag{7.3.1-8}$$

This is called Explicit Alternating Direction in the PR form in two dimensions, i.e., the EAD-PR-2 scheme. After appropriate reordering on columnwise, equations (7.3.1-7) and (7.3.1-8) becomes

$$(sI + G_1)u_c^{p+1/2} = (sI - G_2)u_c^p + b_{2c} \tag{7.3.1-9}$$

$$(sI + G_2)u_c^{k+1} = (sI - G_1)u_c^{p+1/2} + b_{2c} \tag{7.3.1-10}$$

where c denotes the columnwise ordering.

We now seek to analyse the convergence of the EAD-PR-2 scheme. The respective iteration matrix of the pair (A) and pair (B) is

$$T_{(A)}^{PR} = (sI + G_2)^{-1}(sI - G_1)(sI + G_1)^{-1}(sI - G_2) \tag{7.3.1-11}$$

and $$T_{(B)}^{PR} = (sI + G_4)^{-1}(sI - G_3)(sI + G_3)^{-1}(sI - G_4). \tag{7.3.1-12}$$

This gives the iteration matrix of the EAD-PR-2 scheme as

$$\begin{aligned} T_{rs}^{PR} &= T_{(A)}^{PR} T_{(B)}^{PR} \\ &= (sI + G_2)^{-1}(sI - G_1)(sI + G_1)^{-1}(sI - G_2)(sI + G_4)^{-1} \\ &\qquad (sI - G_3)(sI + G_3)^{-1}(sI - G_4). \end{aligned} \tag{7.3.1-13}$$

Let $\alpha > 0$, $\beta > 0$, $\mu > 0$ and $\nu > 0$ be the eigenvalues of G_1, G_2, G_3 and G_4. Since these matrices have the positive eigenvalues, then

$$\|T_{rs}^{PR}\|_2 = \left|\frac{(s-r-\alpha)(s-r-\beta)(s-r-\mu)(s-r-\nu)}{(s+r+\alpha)(s+r+\beta)(s+r+\mu)(s+r+\nu)}\right| < 1. \tag{7.3.1-14}$$

If $\gamma = max\ (\alpha,\mu)$ and $\varphi = max\ (\beta,\nu)$, then

$$\|T_{rs}^{PR}\| = \left|\frac{(s-r-\gamma)(s-r-\varphi)}{(s+r+\gamma)(s+r+\varphi)}\right|^2$$

$$= \left|\frac{(s-r-\psi)}{(s+r+\psi)}\right|^4 < 1. \qquad (7.3.1\text{-}15)$$

where $\psi = max\ (\gamma,\varphi)$. Thus, the EAD-PR-2 scheme is convergent.

We now derive the EAD-GT-2 scheme for solving the two dimensional problem, i.e., by using Guittet's form. The Guittet form to solve the two dimensional problem is given by

$$(rI + H)\mathbf{u}^{k+1/2} = [(rI + H)(rI + V) - \omega rA]\mathbf{u}^k + \omega r\mathbf{b} \qquad (7.3.1\text{-}16)$$

$$(rI + V)\mathbf{u}^{k+1} = \mathbf{u}^{k+1/2} \qquad (7.3.1\text{-}17)$$

Now, let us consider further the matrices A_1 and A_2 as in (7.3.1-1) and (7.3.1-2) respectively, which give

$$A_1\mathbf{u}^{k+1/2} = \mathbf{b}_1, \qquad (7.3.1\text{-}18)$$

where $\mathbf{b}_1 = (A_1A_2 - \omega rA)]\mathbf{u}^k + \mathbf{b}$, and

$$A_2\mathbf{u}^{k+1} = \mathbf{u}^{k+1/2}. \qquad (7.3.1\text{-}19)$$

By using the Guittet form, for any $s > 0$, the EAD-GT-2 scheme can be written as pairs (C) and (D), i.e.,

$$\text{(C)}\quad (sI + G_1)\mathbf{u}^{p+1/2} = [(sI + G_1)(sI + G_2) - \omega sA_1]\mathbf{u}^p + \omega s\mathbf{b}_1$$

$$(sI + G_2)\mathbf{u}^{k+1/2} = \mathbf{u}^{p+1/2}, \qquad (7.3.1\text{-}20)$$

$$\text{(D)}\quad (sI + G_3)\mathbf{u}^{p+1/2} = [(sI + G_3)(sI + G_4) - \omega sA_2]\mathbf{u}^p + \omega s\mathbf{u}^{k+1/2}$$

$$(sI + G_4)\mathbf{u}^{k+1} = \mathbf{u}^{p+1/2}. \qquad (7.3.1\text{-}21)$$

After appropriate reordering on columnwise, the pair (D) becomes

$$\text{(D')}\quad (sI + G_1)\mathbf{u}_c^{p+1/2} = [(sI + G_1)(sI + G_2) - \omega sA_1]\mathbf{u}_c^p + \omega s\mathbf{u}_c^{k+1/2}$$

$$(sI + G_2)\mathbf{u}_c^{k+1} = \mathbf{u}_c^{p+1/2} \qquad (7.3.1\text{-}22)$$

where c denotes columnwise ordering.

The respective iteration matrix of the pairs (A) and (B) is

$$T_{(C)}^{GT} = 1 - \omega s(sI + G_2)^{-1}(sI + G_1)^{-1}A_1 \qquad (7.3.1\text{-}23)$$

and $$T^{GT}_{(D)} = 1 - \omega s(sI + G_4)^{-1}(sI + G_3)^{-1}A_2. \qquad (7.3.1\text{-}24)$$

This gives the iteration matrix of the EAD-GT-2 scheme as

$$T^{GT}_{rs} = T^{GT}_{(A)} T^{GT}_{(B)}$$
$$= [1 - \omega s(sI + G_2)^{-1}(sI + G_1)^{-1}A_1][1 - \omega s(sI + G_4)^{-1}(sI + G_3)^{-1}A_2] \qquad (7.3.1\text{-}25)$$

We now analyse the convergence of the EAD-GT-2 scheme.

Let $\alpha > 0$, $\beta > 0$, $\mu > 0$ and $\nu > 0$ be the eigenvalues of G_1, G_2, G_3 and G_4. Since these matrices have the positive eigenvalues, then

$$\|T^{GT}_{rs}\|_2 = \left|1 - \frac{\omega s(r+\alpha+\beta)}{(s+r+\alpha)(s+r+\beta)}\right| \left|1 - \frac{\omega s(r+\mu+\nu)}{(s+r+\mu)(s+r+\nu)}\right| < 1 \qquad (7.3.1\text{-}26)$$

If $\gamma = max\ (\alpha, \mu)$ and $\varphi = max\ (\beta, \nu)$, then

$$\|T^{GT}_{rs}\|_2 = \left|1 - \frac{\omega s(r+\gamma+\varphi)}{(s+r+\gamma)(s+r+\varphi)}\right|^2$$
$$= \left|\frac{(s+r+\gamma)(s+r+\varphi) - \omega s(r+\gamma+\varphi)}{(s+r+\gamma)(s+r+\varphi)}\right|^2$$
$$= \left|\frac{(s+r+\psi)^2 - \omega s(r+2\psi)}{(s+r+\psi)^2}\right|^2 < 1 \qquad (7.3.1\text{-}27)$$

where $\psi = max\ (\gamma, \varphi)$. Thus, the scheme is convergent.

It can be shown that $(s-r-\psi)^4 < [(s+r+\psi)^2 - \omega s(r+2\psi)]^2$ for any r, ψ, $s > 0$ and $1 < \omega < 2$. Thus, it can be concluded that $\|T^{PR}_{rs}\| < \|T^{GT}_{rs}\|$ which shows the EAD-PR-2 scheme converges faster than the EAD-GT-2 scheme.

For the two dimensional problem in a square region, let us consider the matrix A as in (6.1.1-11). With the splitting of $A = H + V$, we get

$$H = \begin{bmatrix} \overline{H} & & & 0 \\ & \overline{H} & & \\ & & \ddots & \\ 0 & & & \overline{H} \end{bmatrix}_{N^2 X N^2}, \text{ with } \overline{H} = \begin{bmatrix} 2g & -1 & & 0 \\ -1 & 2g & -1 & \\ & \ddots & \ddots & \ddots \\ & & -1 & 2g & -1 \\ 0 & & & -1 & 2g \end{bmatrix}$$

(7.3.1-28)

and

$$V = \begin{bmatrix} \overline{V} & -I & & 0 \\ -I & \overline{V} & -I & \\ & \ddots & \ddots & \ddots \\ & & -I & \overline{V} & -I \\ 0 & & & -I & \overline{V} \end{bmatrix}_{N^2 X N^2}, \text{ with } \overline{V} = \begin{bmatrix} 2g & & & 0 \\ & 2g & & \\ & & \ddots & \\ & & & 2g \\ 0 & & & & 2g \end{bmatrix}.$$

(7.3.1-29)

For any iteration parameter, $r > 0$, we have

$$rI + H = \begin{bmatrix} \overline{H}_1 & & & 0 \\ & \overline{H}_1 & & \\ & & \ddots & \\ & & & \overline{H}_1 \\ 0 & & & & \overline{H}_1 \end{bmatrix}_{N^2 X N^2}, \text{ with } \overline{H}_1 = \begin{bmatrix} \alpha & -1 & & 0 \\ -1 & \alpha & -1 & \\ & \ddots & \ddots & \ddots \\ & & -1 & \alpha & -1 \\ 0 & & & -1 & \alpha \end{bmatrix}$$

(7.3.1-30)

and

$$rI + V = \begin{bmatrix} \overline{V}_1 & -I & & 0 \\ -I & \overline{V}_1 & -I & \\ & \ddots & \ddots & \ddots \\ & & -I & \overline{V}_1 & -I \\ 0 & & & -I & \overline{V}_1 \end{bmatrix}_{N^2 X N^2}, \text{ with } \overline{V}_1 = \begin{bmatrix} \alpha & & & 0 \\ & \alpha & & \\ & & \ddots & \\ & & & \alpha \\ 0 & & & & \alpha \end{bmatrix}$$

$$rI - V = \begin{bmatrix} \overline{V}_2 & I & & 0 \\ I & \overline{V}_2 & I & \\ & \ddots & \ddots & \ddots \\ & & I & \overline{V}_2 & I \\ 0 & & & I & \overline{V}_2 \end{bmatrix}_{N^2 X N^2}, \text{ with } \overline{V}_2 = \begin{bmatrix} \beta & & & 0 \\ & \beta & & \\ & & \ddots & \\ & & & \beta \\ 0 & & & & \beta \end{bmatrix}$$

(7.3.1-31)

where $\alpha = r + 2g$ and $\beta = r - 2g$.

Now, if $rI + H = G_1 + G_2$, we have

$$G_1 = \begin{bmatrix} \bar{G}_1 & & 0 \\ & \bar{G}_1 & \\ & & \ddots \\ & & & \bar{G}_1 \\ 0 & & & & \bar{G}_1 \end{bmatrix}_{N^2 X N^2}, \quad G_2 = \begin{bmatrix} \bar{G}_2 & & 0 \\ & \bar{G}_2 & \\ & & \ddots \\ & & & \bar{G}_2 \\ 0 & & & & \bar{G}_2 \end{bmatrix}_{N^2 X N^2},$$

$$\bar{G}_1 = \begin{bmatrix} w & -1 & & & \\ -1 & w & & & \\ & & \ddots & & \\ & & & w & -1 \\ & & & -1 & w \end{bmatrix} \quad \text{and} \quad \bar{G}_2 = \begin{bmatrix} w & & & \\ & w & -1 & \\ & -1 & w & \\ & & & \ddots \\ & & & & w \end{bmatrix}$$

(7.3.1-32)

where $w = \alpha/2$.

Evidently by interchanging the order of direction we have $(G_1 + G_2)$ in the form of $(G_3 + G_4)$ and vice-versa.

For any iteration parameter $s > 0$, we have

$$sI + G_1 = \begin{bmatrix} G_1' & & 0 \\ & G_1' & \\ & & \ddots \\ & & & G_1' \\ 0 & & & & G_1' \end{bmatrix}_{N^2 X N^2}, \quad sI + G_2 = \begin{bmatrix} G_2' & & 0 \\ & G_2' & \\ & & \ddots \\ & & & G_2' \\ 0 & & & & G_2' \end{bmatrix}_{N^2 X N^2},$$

$$G_1' = \begin{bmatrix} \gamma & -1 & & & \\ -1 & \gamma & & & \\ & & \ddots & & \\ & & & \gamma & -1 \\ & & & -1 & \gamma \end{bmatrix} \quad \text{and} \quad G_2' = \begin{bmatrix} \gamma & & & \\ & \gamma & -1 & \\ & -1 & \gamma & \\ & & & \ddots \\ & & & & \gamma \end{bmatrix},$$

$$sI - G_1 = \begin{bmatrix} G_1'' & & 0 \\ & G_1'' & \\ & \ddots & \\ 0 & & G_1'' \\ & & G_1' \end{bmatrix}_{N^2 \times N^2}, \quad sI - G_2 = \begin{bmatrix} G_2'' & & 0 \\ & G_2'' & \\ & \ddots & \\ 0 & & G_2'' \\ & & G_2'' \end{bmatrix}_{N^2 \times N^2},$$

$$G_1' = \begin{bmatrix} \delta & 1 & & & & \\ 1 & \delta & & & & \\ & & \ddots & & & \\ & & & \delta & 1 & \\ & & & 1 & \delta & \\ & & & & & \delta \end{bmatrix} \quad \text{and} \quad G_2' = \begin{bmatrix} \delta & & & & & \\ & \delta & 1 & & & \\ & 1 & \delta & & & \\ & & & \ddots & & \\ & & & & \delta & 1 \\ & & & & 1 & \delta \end{bmatrix}$$

(7.3.1-33)

where $\gamma = s + w$ and $\delta = s - w$.

Since r, $s > 0$, then the matrices $(sI + G_1)^{-1}$ and $(sI + G_2)^{-1}$ do exist. It is obvious that these matrices consist of the (2x2) block submatrices, $\hat{G}$, i.e.,

$$\hat{G} = \begin{bmatrix} \gamma & -1 \\ -1 & \gamma \end{bmatrix} \text{ and its inverse is } \hat{G}^{-1} = d \begin{bmatrix} \gamma & 1 \\ 1 & \gamma \end{bmatrix},$$

where $d = 1/(\gamma^2 - 1)$.

We now write the algorithm for the EAD-PR-2 and EAD-GT-2 schemes.

Algorithm 7.3.1-1: The EAD-PR-2 scheme.

set $u_{i,j}^{(k)} = 0$, i, j, $= 0$, $N+1$, $\alpha = r + 2g$, $\beta = r - 2g$, $w = \alpha/2$,

$\gamma = s + w$, $\delta = s - w$, $d = 1/(\gamma^2 - 1)$, $\gamma_1 = \gamma d$.

Step 1. Set $u_{i,j}^{(p+1/2)} = 0$, i, $j = 0$, $N+1$.

Step 2. Compute $\mathbf{b1} = (rI - V)\mathbf{u}^{(k)} + \mathbf{b}$.

for $j = 1$, N, compute

for $i = 1$, N, compute

$$b1_{i,j} = u_{i,j-1}^{(k)} + \beta u_{i,j}^{(k)} + u_{i,j+1}^{(k)} + b_{i,j}.$$

Step 3. Set $u_{i,j}^{(p)} = u_{i,j}^{(k+1/2)}$, $i, j = 0, N+1$.

Step 4. Compute $\mathbf{u}^{(p+1/2)} = (sI + G_1)^{-1}[(sI - G_2)\mathbf{u}^{(p)} + \mathbf{b1}]$.

Set $i, j = 1$.

while $j \le N$, compute

while $i \le N-2$, compute

$$r_1 = u_{i-1,j}^{(p)} + \delta u_{i,j}^{(p)} + b1_{i,j}$$

$$r_2 = \delta u_{i+1,j}^{(p)} + u_{i+2,j}^{(p)} + b1_{i+1,j}$$

$$u_{i,j}^{(p+1/2)} = \gamma_1 r_1 + r_2 d, \qquad u_{i+1,j}^{(p+1/2)} = r d_1 + \gamma_1 r_2$$

$i = i + 1$

$$u_{N,j}^{(p+1/2)} = (u_{N-1,j}^{(p)} + \delta u_{N,j}^{(p)} + b1_{N,j})/\gamma$$

$j = j + 1$.

Step 5. Compute $\mathbf{u}^{(k+1/2)} = (sI + G_2)^{-1}[(sI - G_1)\mathbf{u}^{(p+1/2)} + \mathbf{b1}]$.

Set $i = 2, j = 1$.

while $j \le N$, compute

$$u_{1,j}^{(k+1/2)} = (\delta u_{1,j}^{(p+1/2)} + \delta u_{2,j}^{(p+1/2)} + b1_{1,j})/\gamma$$

while $i \le N-1$, compute

$$r_1 = u_{i-1,j}^{(p+1/2)} + \delta u_{i,j}^{(p+1/2)} + b1_{i,j}$$

$$r_2 = \delta u_{i+1,j}^{(p+1/2)} + u_{i+2,j}^{(p+1/2)} + b1_{i+1,j}$$

$$u_{i,j}^{(k+1/2)} = \gamma_1 r_1 + r_2 d, \qquad u_{i+1,j}^{(k+1/2)} = r d_1 + \gamma_1 r_2$$

$i = i + 1$

$j = j + 1$.

Step 6. Set $u_{i,j}^{(q+1/2)} = 0$, $i, j = 0, N+1$.

Step 7. Compute $\mathbf{b2}_c = (rI - V)\mathbf{u}_c^{(k+1/2)} + \mathbf{b}_c$.

for $i = 1, N$, compute

for $j = 1, N$, compute

$$b2_{i,j} = u_{i-1,j}^{(k+1/2)} + \beta u_{i,j}^{(k+1/2)} + u_{i+1,j}^{(k+1/2)} + b_{i,j}.$$

Step 8. Set $u_{i,j}^{(q)} = u_{i,j}^{(k+1)}$, $i, j = 0, N+1$.

Step 9. Compute $\mathbf{u}_c^{(q+1/2)} = (sI + G_1)^{-1}[(sI - G_2)\mathbf{u}_c^{(q)} + \mathbf{b2}_c]$.

Set $i, j = 1$.

while $i \le N$, compute

while $j \le N-2$, compute

$$r_1 = u_{i,j-1}^{(q)} + \delta u_{i,j}^{(q)} + b2_{i,j}$$

$$r_2 = \delta u_{i,j+1}^{(q)} + u_{i,j+2}^{(q)} + b2_{i,j+1}$$

$$u_{i,j}^{(q+1/2)} = \gamma_1 r_1 + r_2 d, \qquad u_{i,j+1}^{(q+1/2)} = r d_1 + \gamma_1 r_2$$

$j = j + 1$

$$u_{i,N}^{(q+1/2)} = (u_{i,N-q}^{(q)} + \delta u_{i,N}^{(q)} + b2_{i,N})/\gamma$$

$i = i + 1$.

Step 10. Compute $\mathbf{u}_c^{(k+1)} = (sI + G_2)^{-1}[(sI - G_1)\mathbf{u}_c^{(q+1/2)} + \mathbf{b2}_c]$.

Set $i = 1$, $j = 2$.

while $i \le N$, compute

$$u_{i,1}^{(k+1)} = (\delta u_{i,1}^{(q+1/2)} + \delta u_{i,2}^{(q+1/2)} + b2_{i,1})/\gamma$$

while $j \le N-1$, compute

$$r_1 = u_{i,j-1}^{(q+1/2)} + \delta u_{i,j}^{(q+1/2)} + b2_{i,j}$$

$$r_2 = \delta u_{i,j+1}^{(q+1/2)} + u_{i,j+2}^{(q+1/2)} + b2_{i,j+1}$$

$$u_{i,j}^{(k+1)} = \gamma_1 r_1 + r_2 d, \qquad u_{i,j+1}^{(k+1)} = r d_1 + \gamma_1 r_2$$

$j = j + 1$

$i = i + 1$.

Step 11. Repeat *Step* 1 to *Step* 10 until convergence is achieved.

In the EAD-GT-2 scheme, we need to compute the matrices

$$C_r = (rI + H)(rI + V) - \omega r A \qquad (7.3.1\text{-}34)$$

and

$$D_s = (sI + G_1)(sI + G_2) - \omega s A_1 \qquad (7.3.1\text{-}35)$$

in order to determine the vectors **b1** and $\mathbf{u}^{(p+1/2)}$ respectively. The elements of C_r and D_s are shown in Algorithm 7.3.1-2, where they become the coefficients for $\mathbf{u}^{(k)}$ (*Step* 1) and $\mathbf{u}^{(p)}$ (*Step* 3 and 6) respectively.

Algorithm 7.3.1-2: The EAD-GT-2 scheme.

set $u_{i,j}^{(k)} = 0,\ i,\ j,\ = 0,\ N+1,\ \alpha = r + 2g,\ w = \alpha/2,\ \gamma = s + w,$

$d = 1/(\gamma^2 - 1),\ \gamma_1 = \gamma d,\ p_1 = \alpha^2 - 4g\omega r,\ q_1 = \omega r - \alpha,$

$p_2 = \gamma^2 - \alpha\omega s$ and $q_2 = \omega s - \gamma.$

Step 1. Compute $\mathbf{b1} = C_r\mathbf{u}^{(k)} + \omega r\mathbf{b}.$

for $j = 1,\ N$

for $i = 1,\ N$

$$b1_{i,j} = q_1(u_{i,j-1}^{(k)} + u_{i+1,j}^{(k)} + u_{i,j+1}^{(k)} + u_{i-1,j}^{(k)}) + u_{i-1,j-1}^{(k)} + u_{i+1,j-1}^{(k)} + p_1 u_{i,j}^{(k)} + u_{i-1,j+1}^{(k)} + u_{i+1,j+1}^{(k)} + \omega r b_{i,j}^{(k)}.$$

Step 2. Set $u_{i,j}^{(p)} = u_{i,j}^{(k+1/2)},\ i,\ j = 0,\ N+1.$

Step 3. Compute $\mathbf{u}^{(p+1/2)} = (sI + G_1)^{-1}D_s\mathbf{u}^{(p)} + \omega r\mathbf{b1}.$ Set $i,\ j = 1.$

while $j \le N$, compute

while $i \le N-2$, compute

$$r_1 = q_2(u_{i-1,j}^{(p)} + u_{i+1,j}^{(p)}) + p_2 u_{i,j}^{(p)} + u_{i+2,j}^{(p)} + \omega s b1_{i,j}$$

$$r_2 = u_{i-1,j}^{(p)} + p_2 u_{i+1,j}^{(p)}) + q_2(u_{i,j}^{(p)} + u_{i+2,j}^{(p)}) + \omega s b1_{i+1,j}$$

$$u_{i,j}^{(p+1/2)} = \gamma_1 r_1 + r_2 d, \qquad u_{i+1,j}^{(p+1/2)} = r d_1 + \gamma_1 r_2$$

$i = i + 1$

$$u_{N,j}^{(p+1/2)} = (q_2 u_{N-1,j}^{(p)} + p_2 u_{N,j}^{(p)} + \omega s b1_{N,j})/\gamma$$

$j = j + 1.$

Step 4. Compute $\mathbf{u}^{(k+1/2)} = (sI + G_2)^{-1}\mathbf{u}^{(p+1/2)}.$ Set $i = 2,\ j = 1.$

while $j \le N$, compute

$$u_{1,j}^{(k+1/2)} = u_{1,j}^{(p+1/2)}/\gamma$$

while $i \le N-1$, compute

$$u_{i,j}^{(k+1/2)} = \gamma_1 u_{i,j}^{(p+1/2)} + du_{i+1,j}^{(p+1/2)}$$

$$u_{i+1,j}^{(k+1/2)} = du_{i,j}^{(p+1/2)} + \gamma_1 u_{i+1,j}^{(p+1/2)}$$

$i = i + 1$

$j = j + 1$.

Step 5. Set $u_{i,j}^{(p)} = u_{i,j}^{(k+1)}$, $i, j = 0, N+1$.

Step 6. Compute $\mathbf{u}_c^{(p+1/2)} = (sI + G_1)^{-1} D_s \mathbf{u}_c^{(p)} + \omega s \mathbf{u}_c^{(k+1/2)}$.

Set $i, j = 1$.

while $i \le N$, compute

while $j \le N-2$, compute

$$r_1 = q_2(u_{i,j-1}^{(p)} + u_{i,j+1}^{(p)}) + p_2 u_{i,j}^{(p)} + u_{i,j+2}^{(p)} + \omega s u_{i,j}^{(k+1/2)}$$

$$r_2 = u_{i,j-1}^{(p)} + p_2 u_{i,j+1}^{(p)}) + q_2(u_{i,j}^{(p)} + u_{i,j+2}^{(p)}) + \omega s u_{i,j+1}^{(k+1/2)}$$

$$u_{i,j}^{(p+1/2)} = \gamma_1 r_1 + r_2 d, \qquad u_{i,j+1}^{(p+1/2)} = r d_1 + \gamma_1 r_2$$

$j = j + 1$

$$u_{i,N}^{(p+1/2)} = (q_2 u_{i,N-1}^{(p)} + p_2 u_{i,N}^{(p)} + \omega s u_{i,N}^{(k+1/2)})/\gamma$$

$i = i + 1$.

Step 7. Compute $\mathbf{u}_c^{(k+1)} = (sI + G_2)^{-1} \mathbf{u}_c^{(p+1/2)}$. Set $i = 2$, $j = 1$.

while $i \le N$, compute

$$u_{i,1}^{(k+1/2)} = u_{i,1}^{(p+1/2)}/\gamma$$

while $j \le N-1$, compute

$$u_{i,j}^{(k+1)} = \gamma_1 u_{i,j}^{(p+1/2)} + du_{i,j+1}^{(p+1/2)}$$

$$u_{i,j+1}^{(k+1)} = du_{i,j}^{(p+1/2)} + \gamma_1 u_{i,j+1}^{(p+1/2)}$$

$j = j + 1$

$i = i + 1$.

Step 8. Repeat *Step* 1 to *Step* 7 until convergence is achieved.

The EAD method for solving the three dimensional problem can be derived as follows . Let us consider the splitting of the matrix $A = X + Y + Z$, where the matrices X, Y and Z representing the directions along the axis x, y and z respectively. By using the PR form, for any $r > 0$, we have the set of equations as

$$(rI + X)\mathbf{u}^{(k+1/3)} = (rI - Y - Z)\mathbf{u}^{(k)} + \mathbf{b} \qquad (7.3.1\text{-}36)$$

$$(rI + Y)\mathbf{u}^{(k+2/3)} = (rI - X - Z)\mathbf{u}^{(k+1/3)} + \mathbf{b} \qquad (7.3.1\text{-}37)$$

$$(rI + Z)\mathbf{u}^{(k+1)} = (rI - X - Y)\mathbf{u}^{(k+2/3)} + \mathbf{b} \qquad (7.3.1\text{-}38)$$

with the iteration matrix, T_r,

$$T_r = (rI + Z)^{-1}(rI - X - Y)(rI + Y)^{-1}(rI - X - Z) \times (rI + X)^{-1}(rI - Y - Z) \qquad (7.3.1\text{-}39)$$

or

$$(rI + X + Y)\mathbf{u}^{(k+1/3)} = (rI - Z)\mathbf{u}^{(k)} + \mathbf{b} \qquad (7.3.1\text{-}40)$$

$$(rI + Y + Z)\mathbf{u}^{(k+2/3)} = (rI - X)\mathbf{u}^{(k+1/3)} + \mathbf{b} \qquad (7.3.1\text{-}41)$$

$$(rI + X + Z)\mathbf{u}^{(k+1)} = (rI - Y)\mathbf{u}^{(k+2/3)} + \mathbf{b}, \qquad (7.3.1\text{-}42)$$

with the iteration matrix, T_r,

$$T_r = (rI + X + Z)^{-1}(rI - Y)(rI + Y + Z)^{-1}(rI - X) \times (rI + X + Y)^{-1}(rI - Z). \qquad (7.3.1\text{-}43)$$

Equations (7.3.1-36) - (7.3.1-38) are unstable, since the norm of the iteration matrix for this scheme can be greater than one. Equations (7.3.1-40) - (7.3.1-42) are stable, but it is quite difficult to implement. Instead, we will use the Guittet form which is given as follows:

$$(rI + X)\mathbf{u}^{(k+1/3)} = C_r\mathbf{u}^{(k)} + \omega r^2\mathbf{b} \qquad (7.3.1\text{-}44)$$

$$(rI + Y)\mathbf{u}^{(k+2/3)} = \mathbf{u}^{(k+1/3)} \qquad (7.3.1\text{-}45)$$

$$(rI + Z)\mathbf{u}^{(k+1)} = \mathbf{u}^{(k+2/3)} \qquad (7.3.1\text{-}46)$$

where $C_r = (rI + X)(rI + Y)(rI + Y) - \omega r^2 A$.

Let $A_1 = rI + X = G_1 + G_2$, $A_2 = rI + Y = G_3 + G_4$,

$A_3 = rI + Z = G_5 + G_6$, and $\mathbf{b1} = C_r\mathbf{u}^{(k)} + \omega r^2\mathbf{b}$,

which give

$$(G_1 + G_2)\mathbf{u}^{(k+1/3)} = \mathbf{b1} \tag{7.3.1-47}$$

$$(G_3 + G_4)\mathbf{u}^{(k+2/3)} = \mathbf{u}^{(k+1/3)} \tag{7.3.1-48}$$

$$(G_5 + G_6)\mathbf{u}^{(k+1)} = \mathbf{u}^{(k+2/3)}. \tag{7.3.1-49}$$

Then, for any $s > 0$, the EAD method in Guittet's form for solving the three dimensional problem, i.e., the EAD-GT-3 scheme is given by

$$(sI + G_1)\mathbf{u}^{(p+1/2)} = P_s\mathbf{u}^{(p)} + \omega s\mathbf{b1} \tag{7.3.1-50}$$

$$(sI + G_2)\mathbf{u}^{(k+1/3)} = \mathbf{u}^{(p+1/2)} \tag{7.3.1-51}$$

$$(sI + G_3)\mathbf{u}^{(p+1/2)} = Q_s\mathbf{u}^{(p)} + \omega s\mathbf{u}^{(k+1/3)} \tag{7.3.1-52}$$

$$(sI + G_4)\mathbf{u}^{(k+2/3)} = \mathbf{u}^{(p+1/2)} \tag{7.3.1-53}$$

$$(sI + G_5)\mathbf{u}^{(p+1/2)} = R_s\mathbf{u}^{(p)} + \omega s\mathbf{u}^{(k+2/3)} \tag{7.3.1-54}$$

$$(sI + G_6)\mathbf{u}^{(k+1)} = \mathbf{u}^{(p+1/2)} \tag{7.3.1-55}$$

where $P_s = (sI + G_1)(sI + G_2) - \omega sA_1$, $Q_s = (sI + G_3)(sI + G_4) - \omega sA_2$, and $R_s = (sI + G_5)(sI + G_6) - \omega sA_3$.

The iteration matrix for the EAD-GT-3 scheme is given by

$$\begin{aligned} T_{rs}^{GT-3} = {} & [I - \omega s(sI + G_2)^{-1}(sI + G_1)^{-1}A_1] \times \\ & [I - \omega s(sI + G_4)^{-1}(sI + G_3)^{-1}A_2] \times \\ & [I - \omega s(sI + G_6)^{-1}(sI + G_5)^{-1}A_3]. \end{aligned} \tag{7.3.1-56}$$

We now analyse the convergence of the EAD-GT-3 scheme.

Let $\alpha > 0$, $\beta > 0$, $\delta > 0$, $\gamma > 0$, $\mu > 0$ and $\nu > 0$ be the eigenvalues of G_1, G_2, G_3, G_4, G_5 and G_6. Since these matrices have the positive eigenvalues, then

$$\begin{aligned} \|T_{rs}^{GT-3}\|_2 = {} & \left|1 - \frac{\omega s(r+\alpha+\beta)}{(s+r+\alpha)(s+r+\beta)}\right| \left|1 - \frac{\omega s(r+\delta+\gamma)}{(s+r+\delta)(s+r+\gamma)}\right| \times \\ & \left|1 - \frac{\omega s(r+\mu+\nu)}{(s+r+\mu)(s+r+\nu)}\right| < 1. \end{aligned} \tag{7.3.1-57}$$

If $\psi = max\ (\alpha, \delta, \mu)$ and $\varphi = max\ (\beta, \gamma, \nu)$, then

$$\|T_{rs}^{GT-3}\|_2 = \left|1 - \frac{\omega s(r+\psi+\varphi)}{(s+r+\psi)(s+r+\varphi)}\right|^3$$

$$= \left|\frac{(s+r+\psi)(s+r+\varphi) - \omega s(r+\psi+\varphi)}{(s+r+\psi)(s+r+\varphi)}\right|^3$$

$$= \left|\frac{(s+r+\phi)^2 - \omega s(r+2\phi)}{(s+r+\phi)^2}\right|^3 < 1 \qquad (7.3.1\text{-}58)$$

where $\phi = max\ (\psi, \varphi)$. Thus, the EAD-GT-3 scheme is convergent.

In programming, we only need to consider the matrices G_1 and G_2 since after reordering the directions, equations (7.3.1-52) - (7.3.1-55) becomes

$$(sI + G_1)\mathbf{u}_y^{(p+1/2)} = P_s\mathbf{u}_y^{(p)} + \omega s\mathbf{u}_y^{(k+1/3)} \qquad (7.3.1\text{-}59)$$

$$(sI + G_2)\mathbf{u}_y^{(k+2/3)} = \mathbf{u}_y^{(p+1/2)} \qquad (7.3.1\text{-}60)$$

$$(sI + G_1)\mathbf{u}_z^{(p+1/2)} = P_s\mathbf{u}_z^{(p)} + \omega s\mathbf{u}_z^{(k+2/3)} \qquad (7.3.1\text{-}61)$$

$$(sI + G_2)\mathbf{u}_z^{(k+1)} = \mathbf{u}_z^{(p+1/2)}. \qquad (7.3.1\text{-}62)$$

When solving the three dimensional problem in a unit cube, we would then have the matrix A as in (6.1.2-6), and by applying the EAD method, the matrices G_1 and G_2 would be similar to the one given in (7.3.1-32).

In the EAD-GT-3 scheme, we need to compute the matrices

$$C_r = (rI + X)(rI + Y)(rI + Z) - \omega r^2 A \qquad (7.3.1\text{-}64)$$

and

$$P_s = (sI + G_1)(sI + G_2) - \omega s A_1 \qquad (7.3.1\text{-}65)$$

in order to determine the vectors **b1** and $\mathbf{u}^{(p+1/2)}$ respectively.

The elements of C_r and P_s are shown in Algorithm 7.3.1-3, where they become the coefficients for $\mathbf{u}^{(k)}$ and $\mathbf{u}^{(p)}$ respectively.

We now write the algorithm for the EAD-GT-3 scheme.

Algorithm 7.3.1-3: The EAD-GT-3 scheme.

Set $u^{(k)}_{i,j,k} = 0$, i, j, $k = 0$, $N+1$, $\alpha = r + 2g$, $w = \alpha/2$, $\gamma = s + w$,

$d = 1/(\gamma^2 - 1)$, $\gamma_1 = \gamma d$, $p_1 = \alpha^3 - 6g\omega r^2$, $q_1 = \omega r^2 - \alpha^2$,

$p_2 = \gamma^2 - \alpha\omega s$ and $q_2 = \omega s - \gamma$.

Step 1. Compute $\mathbf{b1} = C_r \mathbf{u}^{(k)} + \omega r \mathbf{b}$.

for $k = 1$, N

for $j = 1$, N

for $i = 1$, N

$$
\begin{aligned}
b1_{i,j,k} = &- u^{(k)}_{i-1,j-1,k-1} - u^{(k)}_{i+1,j-1,k-1} - u^{(k)}_{i-1,j+1,k-1} \\
&- u^{(k)}_{i+1,j+1,k-1} - u^{(k)}_{i-1,j-1,k+1} - u^{(k)}_{i+1,j-1,k+1} \\
&- u^{(k)}_{i-1,j+1,k+1} - u^{(k)}_{i+1,j+1,k+1} + \alpha(u^{(k)}_{i,j-1,k-1} \\
&+ u^{(k)}_{i-1,j,k-1} + u^{(k)}_{i+1,j,k-1} + u^{(k)}_{i,j+1,k-1} \\
&+ u^{(k)}_{i-1,j-1,k} + u^{(k)}_{i+1,j-1,k} + u^{(k)}_{i-1,j+1,k} \\
&+ u^{(k)}_{i+1,j+1,k} + u^{(k)}_{i,j-1,k+1} + u^{(k)}_{i-1,j,k+1} \\
&+ u^{(k)}_{i+1,j,k+1} + u^{(k)}_{i,j+1,k+1}) + q_1(u^{(k)}_{i,j,k-1} \\
&+ u^{(k)}_{i,j-1,k} + u^{(k)}_{i-1,j,k} + u^{(k)}_{i+1,j,k} + u^{(k)}_{i,j+1,k} \\
&+ u^{(k)}_{i,j,k+1}) + p_1 u^{(k)}_{i,j,k} + \omega r^2 b_{i,j,k}
\end{aligned}
$$

$i = i + 1$

$j = j + 1$

$k = k + 1$.

Step 2. Set $u^{(p)}_{i,j,k} = u^{(k+1/3)}_{i,j,k}$, i, j, $k = 0$, $N+1$

Step 3. Compute $\mathbf{u}^{(p+1/2)} = (sI + G_1)^{-1} P_s \mathbf{u}^{(p)} + \omega s \mathbf{b1}$. Set i, j, $k = 1$.

while $k \le N$, compute

while $j \le N$, compute

while $i \le N-2$, compute

$$r_1 = q_2(u^{(p)}_{i-1,j,k} + u^{(p)}_{i+1,j,k}) + p_2 u^{(p)}_{i,j,k} + u^{(p)}_{i+2,j,k} + \omega sb1_{i,j,k}$$

$$r_2 = u^{(p)}_{i-1,j,k} + p_2 u^{(p)}_{i+1,j,k} + q_2(u^{(p)}_{i,j,k} + u^{(p)}_{i+2,j,k}) + \omega sb1_{i+1,j,k}$$

$$u^{(p+1/2)}_{i,j,k} = \gamma_1 r_1 + r_2 d, \qquad u^{(p+1/2)}_{i+1,j,k} = r d_1 + \gamma_1 r_2$$

$i = i + 2$

$$u^{(p+1/2)}_{N,j,k} = (q_2 u^{(p)}_{N-1,j,k} + p_2 u^{(p)}_{N,j,k} + \omega sb1_{N,j,k})/\gamma$$

$j = j + 1.$

$k = k + 1.$

Step 4. Compute $\mathbf{u}^{(k+1/3)} = (sI + G_2)^{-1}\mathbf{u}^{(p+1/2)}$. Set $i = 2$, j, $k = 1$.

while $k \le N$, compute

while $j \le N$, compute

$$u^{(k+1/3)}_{i,j,k} = u^{(p+1/2)}_{i,j,k}/\gamma$$

while $i \le N$-1, compute

$$u^{(k+1/3)}_{i,j,k} = \gamma_1 u^{(p+1/2)}_{i,j,k} + d u^{(p+1/2)}_{i+1,j,k}$$

$$u^{(k+1/3)}_{i+1,j,k} = d u^{(p+1/2)}_{i,j,k} + \gamma_1 u^{(p+1/2)}_{i+1,j,k}$$

$i = i + 2$

$j = j + 1$

$k = k + 1.$

Step 5. Set $u^{(p)}_{i,j,k} = u^{(k+2/3)}_{i,j,k}$, i, j, $k = 0$, N+1

Step 6. Compute $\mathbf{u}^{(p+1/2)}_y = (sI + G_1)^{-1}P_s\mathbf{u}^{(p)}_y + \omega s\mathbf{u}^{(k+1/3)}_y$.

Set i, j, $k = 1$.

while $i \le N$, compute

while $k \le N$, compute

while $j \le N$-2, compute

$$r_1 = q_2(u^{(p)}_{i,j-1,k} + u^{(p)}_{i,j+1,k}) + p_2 u^{(p)}_{i,j,k} + u^{(p)}_{i,j+2,k}$$

$$+ \omega s u_{i,j,k}^{(k+1/3)}$$

$$r_2 = u_{i,j-1,k}^{(p)} + p_2 u_{i,j+1,k}^{(p)} + q_2 (u_{i,j,k}^{(p)} + u_{i,j+2,k}^{(p)})$$

$$+ \omega s u_{i,j+1,k}^{(k+1/3)}$$

$$u_{i,j,k}^{(p+1/2)} = \gamma_1 r_1 + r_2 d, \qquad u_{i,j+1,k}^{(p+1/2)} = r d_1 + \gamma_1 r_2$$

$j = j + 2$

$$u_{i,N,k}^{(p+1/2)} = (q_2 u_{i,N-1,k}^{(p)} + p_2 u_{i,N,k}^{(p)} + \omega s u_{i,N,k}^{(k+1/3)})/\gamma$$

$k = k + 1.$

$i = i + 1.$

Step 7. Compute $\mathbf{u}_y^{(k+2/3)} = (sI + G_2)^{-1}\mathbf{u}_y^{(p+1/2)}$. Set i, $k = 1$, $j = 2$.

while $i \le N$, compute

while $k \le N$, compute

$$u_{i,1,k}^{(k+2/3)} = u_{i,1,k}^{(p+1/2)}/\gamma$$

while $j \le N-1$, compute

$$u_{i,j,k}^{(k+2/3)} = \gamma_1 u_{i,j,k}^{(p+1/2)} + d u_{i,j+1,k}^{(p+1/2)}$$

$$u_{i,j+1,k}^{(k+2/3)} = d u_{i,j,k}^{(p+1/2)} + \gamma_1 u_{i,j+1,k}^{(p+1/2)}$$

$j = j + 2$

$k = k + 1$

$i = i + 1.$

Step 8. Set $u_{i,j,k}^{(p)} = u_{i,j,k}^{(k+1)}$, i, j, $k = 0$, $N+1$

Step 9. Compute $\mathbf{u}_z^{(p+1/2)} = (sI + G_1)^{-1}P_s\mathbf{u}_z^{(p)} + \omega s\mathbf{u}_z^{(k+2/3)}$.

Set i, j, $k = 1$.

while $j \le N$, compute

while $i \le N$, compute

while $k \le N-2$, compute

$$r_1 = q_2 (u_{i,j,k-1}^{(p)} + u_{i,j,k+1}^{(p)}) + p_2 u_{i,j,k}^{(p)} + u_{i,j,k+2}^{(p)}$$

$$+ \omega s u_{i,j,k}^{(k+2/3)}$$

$$r_2 = u^{(p)}_{i,j,k-1} + p_2 u^{(p)}_{i,j,k+1} + q_2(u^{(p)}_{i,j,k} + u^{(p)}_{i,j,k+2}) + \omega s u^{(k+2/3)}_{i,j,k+1}$$

$$u^{(p+1/2)}_{i,j,k} = \gamma_1 r_1 + r_2 d, \qquad u^{(p+1/2)}_{i,j,k+1} = r d_1 + \gamma_1 r_2$$

$k = k + 2$

$$u^{(p+1/2)}_{i,j,N} = (q_2 u^{(p)}_{i,j,N-1} + p_2 u^{(p)}_{i,j,N} + \omega s u^{(k+2/3)}_{i,j,N})/\gamma$$

$i = i + 1.$

$j = j + 1.$

Step 10. Compute $\mathbf{u}_z^{(k+1)} = (sI + G_2)^{-1}\mathbf{u}_z^{(p+1/2)}$. Set i, $j = 1$, $k = 2$.

while $j \le N$, compute

while $i \le N$, compute

$$u^{(k+1)}_{i,j,1} = u^{(p+1/2)}_{i,j,1}/\gamma$$

while $k \le N-1$, compute

$$u^{(k+1)}_{i,j,k} = \gamma_1 u^{(p+1/2)}_{i,j,k} + d u^{(p+1/2)}_{i,j,k+1}$$

$$u^{(k+1)}_{i,j,k+1} = d u^{(p+1/2)}_{i,j,k} + \gamma_1 u^{(p+1/2)}_{i,j,k+1}$$

$k = k + 2$

$i = i + 1$

$j = j + 1.$

Step 11. Repeat *Step* 1 to *Step* 10 until convergence is achieved.

7.3.2 Experimental results

Numerical results presented in this section are concerned with the application of the EAD method. Four problems were tested, with the first two are in two dimensions and the other are in three dimensions.

For the two dimensional problems, both results for the EAD-PR and EAD-GT-2 schemes are presented, whilst for three dimensional problems, only the EAD-GT-3 scheme is considered.

Problem 1 - The Laplace Equation in two dimensions

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = 0, \qquad 0 \le x, y \le 1,$$

subject to the boundary conditions

$$U(x,0) = U(x,1) = \sin \pi x, \qquad 0 \le x \le 1,$$

$$U(0,y) = U(1,y) = 0, \qquad 0 \le y \le 1.$$

The exact solution is given by $U(x,y) = \operatorname{sech} \frac{\pi}{2} \cosh \pi(y - \frac{1}{2}) \sin \pi x$.

The results are tabulated in *Table* 7.3.2-1.

h^{-1}	AGE-DG-2 scheme		EAD-PR scheme			EAD-GT-2 scheme		
	r	iter	r	s	iter	r	s	iter
10	0.96 - 1.00	22	0.84	0.40	15	0.81	0.47	16
20	0.57 - 0.58	47	0.43	0.23	29	0.45	0.26	33
40	0.34	102	0.24	0.12	65	0.25	0.13	69
40	0.19	209	0.13	0.06	137	0.13	0.06	137

Table 7.3.2-1: Number of iterations for Problem 1

Problem 2 - The Poisson Equation

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = -2, \qquad 0 \le x, y \le 1,$$

The boundary conditions are given by

$$U(0,y) = 0,\ U(1,y) = \sinh \pi x \sin \pi y, \qquad 0 \le y \le 1,$$

$$U(x,0) = U(x,1) = x(1 - x), \qquad 0 \le x \le 1.$$

The exact solution is given by $U(x,y) = \sinh \pi x \sin \pi y + x(1 - x)$.

The results are tabulated in *Table* 7.3.2-2.

h^{-1}	AGE-DG-2 scheme		EAD-PR scheme			EAD-GT-2 scheme		
	r	iter	r	s	iter	r	s	iter
10	0.97 - 1.08	27	0.80	0.48	17	0.82	0.53	19
20	0.58 - 0.59	56	0.44	0.26	38	0.44	0.30	41
40	0.34	116	0.25	0.14	82	0.26	0.16	90
80	0.18	228	0.14	0.07	176	0.15	0.08	198

Table 7.3.2-2: Number of iterations for Problem 2

Problem 3 - The Laplace Equation in three dimensions

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} + \frac{\partial^2 U}{\partial z^2} = 0, \qquad 0 \le x,y,z \le 1,$$

subject to the boundary conditions

$$U(x,y,0) = U(x,y,1) = 0, \qquad 0 \le x,y \le 1,$$

$$U(0,y,z) = U(1,y,z) = 0, \qquad 0 \le y,z \le 1,$$

$$U(x,0,z) = 0, \; U(x,1,z) = \sin \pi x \sin \pi z, \qquad 0 \le x,z \le 1.$$

The exact solution is given by

$$U(x,y,z) = \operatorname{sech} \frac{\pi}{\sqrt{2}} \sin \pi x \cosh [\sqrt{2}\pi(y - \tfrac{1}{2}) \sin \pi z.$$

The result is tabulated in *Table* 7.3.2-3.

h^{-1}	AGE-DG-3 (ω = 2)		EAD-GT-3 scheme		
	r	iter	r	s	iter
12	1.22 - 1.25	26	0.98	0.49	18
14	1.08 - 1.11	31	0.82	0.44	20
16	0.98	35	0.71	0.40	23
18	0.88 - 0.89	40	0.61	0.36	25
20	0.81	44	0.53	0.32	26

Table 7.3.2-3: Number of iterations for Problem 3

Problem 4 - The Laplace Equation in three dimensions

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} + \frac{\partial^2 U}{\partial z^2} = 0, \qquad 0 \le x,y,z \le 1,$$

subject to the boundary condition

$$U(x,y,0) = 0, \; U(x,y,1) = 400xy, \qquad 0 \le x,y \le 1.$$

$$U(0,y,z) = 0, \; U(1,y,z) = 400yz, \qquad 0 \le y,z \le 1.$$

$$U(x,0,z) = 0, \; U(x,1,z) = 400xz, \qquad 0 \le x,z \le 1.$$

The exact solution is given by

$$U(x,y,z) = 400xyz.$$

The results are tabulated in *Table* 7.3.2-4.

h^{-1}	AGE-DG-3 (ω = 2)		EAD-GT-3 scheme		
	r	iter	r	s	iter
12	1.42 - 1.50	42	1.10	0.65	30
14	1.25 - 1.33	50	0.99	0.57	36
16	1.13 - 1.17	57	0.90	0.52	42
18	1.03 - 1.06	64	0.83	0.47	49
20	0.95	70	0.78	0.43	56

Table 7.3.2-4: Number of iterations for Problem 4

The results outline the performance of the EAD method compared to the AGE-DG scheme in terms of the number of iterations. The results for the two dimensional problems agree with the theory as the EAD-PR scheme shows a smaller number of iterations than the EAD-GT-2 scheme. Both schemes are also shown to converge faster than the AGE-DG-2 scheme.

A better performance for the three dimensional problems is also obtained where the EAD-GT-3 scheme converges much faster than the AGE-DG-3 scheme. For the two dimensional problem, however, the rate of convergence becomes slower at a larger number of points. This could be due the rounding error growth as the value of s becomes small.

The improvements shown in terms of the number of iterations for the given problems are at the expense of one more parameter, i.e., s. Experimentally, this parameter is found to be given by the relation $s \approx 0.5r$. Both parameters r and s are determined heuristically but it is not too easy. However, from the experiments (for N = 9), we may expect that r is closed to 1. This approximation can be taken as a guide to determine the experimental parameters for other problems as the theory regarding these parameters has yet to be developed.

Although there are some setbacks with these schemes such as the determination of parameter r and the intensive computations involved, nevertheless it produces good results especially when solving the three

dimensional problem.

Other good points to recommend the method such as the parameters r and s are within the interval [0,1] and there is a simple relation between them, makes the search for parameters not too difficult. Thus, the EAD-GT-3 scheme might be regarded as successful for the application of multi-parameters to solve the three dimensional problem.

7.4 Summary

This chapter is a continuation of the application of multi-parameters to accelerate the AGE method as discussed earlier in Chapter 5. The problems considered in this chapter are of higher dimensions, i.e., two and three dimensional problems.

The early section is concerned with the solution of the different regions of a two dimensional problem and a unit cube elliptic equation by using the existing ADI(AGE) parameters. However, none of these are suitable for the AGE method. Instead, the heuristic search is performed to find a better convergence. The results obtained are remarkable, but the difficulties in determining the correct parameters forces the strategy to be considered very carefully.

The application of the AGE-Richardson and AGE-Chebyshev methods discussed in Chapter 5 is enlarged in the next section for the solution of problems with higher dimensions. The results are not as good as obtained in the earlier section especially for the two dimensional problem, but the simplicity to determine the parameter ω_{k+1} in the AGE-Chebyshev method, makes the method viable for the application of multi-parameters.

The newly developed technique, called the EAD method is discussed in Section 5.3. The method is based on the the PR and Guittet's form. The slight problem with these schemes is the lengthy computational algorithm involved, but the results obtained are competitive with others in the previous sections, especially for the solution for three dimensional problem.

The results for two dimensional problems, again, show a little improvements when larger numbers of points are involved. However, all the parameters are well within the interval [0,1] which shows that the schemes are stable. The other difficulty is that the parameters r and s has to be determined heuristically. However, this is compensated by the simple relation $s = 0.5r$, which certainly will ease the parameter search.

By careful considerations the AGE-Chebyshev method and the EAD-GT-3 scheme might be recommended for solving the three dimensional problems. For the two dimensional problems, though the AGE-Chebyshev method seems viable, it is worthwhile to focus only on the application of a single parameter.

PART IV

CONCLUSIONS

CHAPTER 8

CONCLUSIONS AND SUGGESTIONS FOR FURTHER RESEARCH

8.1 Summary and conclusions

Throughout the course of this thesis, a new explicit method, the AGE method has been proposed to solve ordinary and partial differential equations. The emphasis is on the convergence, stability, consistency, accuracy and efficiency of the various schemes derived from the AGE method.

The first observation is in Chapter 3, where a comparative study of the existing methods, i.e., SOR, with the AGE method is given to solve the two point boundary value problem. The AGE method, based on the ADI method in PR form, has been shown to perform better than the SOR method especially when solving the linear problems when matrix A is strongly diagonally dominant, irrespective of the boundary conditions. The method has also been shown to be stable for a higher order accuracy, i.e., on using the Numerov method for approximation.

The solution of the two point boundary problem by the AGE method is enlarged in Chapter 4. In this chapter, the AGE method has been rewritten in various alternative forms in order to achieve the possibility of time saving, its expansion to solve problems in higher dimension, and more important to make full use of the explicitness of the method so that, it may well be suitable to use in parallel computation. The SMAGE scheme is found to be the fastest especially for solving a nonlinear problem, the AGE-DG scheme can also be extended for solving the multi-dimensional problem and the CAGE method can also be considered when parallelism is concerned.

The solution for the two point boundary value problems are investigated further in Chapter 5, concerning the application of multi-parameters. The results appear to be disappointing and only show improvement in terms of the number of iterations, when solving the linear problem governed by periodic boundary conditions. The commutativity properties of the matrices G_1 and G_2 in this problem make the AGE method converge faster. It has also been shown that for this problem, the application of 2 parameters is better than the 3 parameter case. Thus, the 2 parameter case might well be considered when applying the multi-parameter formulae. However, it is suspected that better results can be obtained from larger orders of matrices.

Chapter 6 begins with the extension of the AGE-DG scheme for solving the two and three dimensional problems subject to different boundary conditions. As shown, the AGE method is capable for solving these problems and is better than the SOR method. The difficulty in determining the theoretical optimal parameter has been overcome by using some rather simple heuristic assumptions.

The AGE-DG scheme is also shown to be viable when solving the two and three dimensional problems governed by periodic and Neumann boundary conditions. On comparison with the SOR method, again, the AGE-DG scheme has the edge. Although the AGE-DG scheme needs more computational effort, the simplicity of the scheme and the facile way to obtain of the parameter r, give the scheme the advantage.

The alternative computational forms for reducing the CPU time are also tested in this chapter. However, for sequential computation, the best form is the AGE-DG scheme. Nevertheless, the CAGE method for the two dimensional problems can be applied on parallel computers.

The thesis is concluded in Chapter 7, by extending the discussion of the multi-parameter case in elliptic problems. The application of the existing ADI(AGE) parameters, the AGE-Richardson and AGE-Chebyshev methods are extended for two and three dimensional problem. The new technique based on the PR and Guittet's form, called the EAD method, for solving the problem with higher dimensions are added to this chapter as another strategy for the use of multi-parameters for accelerating the AGE method.

The conclusions of this section is that the EAD method is well ahead of other strategies when the solution with two parameters is concerned. The second choice is the AGE-Chebyshev method. The heuristic search cannot be regarded as a good strategy since it may develop a rounding error growth which will then affect the stability of the method.

The disappointing results for the multi-parameter case might be explained as follows. The bigger block (40x40) for the ADI-PR method will definitely be suitable for a sequence of parameters. If we reduce the problem to the (2x2) block ADI-PR method, it will automatically become the AGE method. Thus, we would not expect to use the same sequence of parameters for this particular problem. In other words, this can be attributed to a scaling factor.

To conclude, the AGE method is viable for solving any two point boundary value problems and elliptic equations regardless of boundary conditions by using a single parameter. The method is also stable for a problem with higher order accuracy.

For the application of multi-parameters, only the two parameter case can be considered for the AGE method. This is due to the fact

that, the AGE method is comprised only of small (2x2) blocks, which naturally will only have two eigenvalues with higher order multiplicity.

8.2 Suggestions for further research

The AGE method to date has been investigated for a limited number of problems and clearly further numerical investigations into a wider class of problems, especially nonlinear, would be beneficial for the wider understanding of the method.

Further applications of the method to non self-adjoint partial differential equations, would also be advantageous for now the matrix A will become unsymmetric and have complex eigenvalues.

Extension of the AGE method can also be considered by the use of alternative approximations for the differential operators, i.e., spline and finite elements methods, and the use of variational methods such as Galerkin and Conjugate Gradient methods.

Recently, the use of the preconditioning method has become a standard strategy for the acceleration of iterative methods. Consequently, a suitable preconditioner for the AGE method would be of great importance.

Finally, for any implicit method, often we are not able to exploit the method to the full when the issue of parallelism is concerned. Thus, although the AGE method appear to require slightly more computational work, i.e., in developing the CAGE method, however, with its principal advantage, i.e., explicitness, could serve as an efficient technique for achieving parallel computation.

REFERENCES

ABDULLAH, A.R.B. [1983], *The Study of Some Numerical Methods for Solving Parabolic Partial Differential Equations*, Ph.D. Thesis, Loughborough University of Technology.

AHMAD, A.R. [1985], *Acceleration Strategies for the Alternating Group Explicit (AGE) Method for Solving Linear Equations*, M.Sc. Dissertation, Dept. of Mathematics, Loughborough University of Technology.

AHMAD, A.R. [1990], *Kaedah Berangka Permulaan: Pengaturcaraan dengan Turbo Pascal*, Dewan Bahasa dan Pustaka, Kuala Lumpur, Malaysia.

AHMAD, A.R. and **EVANS, D.J.** [1991], *Convergence Rate of the Alternating Group Explicit (AGE) Method for m (>1) Iteration Parameter*, Computer Studies 652, Loughborough University of Technology.

AL-WALI, A.A. *Private Communication.*

AMES, W.F. [1977], *Numerical Methods for Partial Differential Equations*, Second Edition, Academic Press, New York.

BIRKHOFF, G., VARGA, R.S., and **YOUNG, D.M.** [1962], *Alternating Direction Implicit methods*, Advances in Computers, Vol. 3, Academic Press, New York.

BROWN, W.S and **HEARN, A.C** [1978], *Application of Symbolic Algebraic Computation*, Proceedings, Computational Atomic and Molecular Physics Conference, Nottingham, England.

BROYDEN, C.G. [1975], *Basic Matrices*, The MacMillan Press Ltd, London.

COLLATZ, L. [1960], *The Numerical Treatment of Differential Equations*, Springer-Verlag, Berlin.

CONTE, S.D. and **de BOOR, C.** [1972], *Elementary Numerical Analysis, An Algorithmic Approach*, Second Edition, McGraw-Hill Kogakusha Ltd., Tokyo.

DOUGLAS, J. [1962], *Alternating Direction Methods for Three Space Variables*, Numer. Math., Vol. 4, pp. 41 - 63.

DOUGLAS, J. and **PEARCY, C.M.** [1963], *On Convergence of Alternating Direction Procedures in the Presence of Singular Operators*, Numer. Math. Vol. 5, pp. 175-184.

DOUGLAS, J. and **RACHFORD, H.H.** [1956], *On the Numerical Solution of Heat Conduction Problems in Two and Three Space Variables*, Trans. Amer. Maths. Soc., Vol. 82, pp. 421-439.

EVANS, D.J. [1972], *A New Iterative Procedure for the Solution of Sparse Systems of Linear Difference Equations*, in "Sparse Matrices and Their Application", eds., Rose, D.J. and Willoughby, R.A., Plenum Press, New York, pp. 89-100.

EVANS, D.J. [1974], *Iterative Sparse Matrix Algorithms, in Software for Numerical Mathematics*, (Evans, D.J. Ed.) Academic Press, pp. 49-83.

EVANS, D.J. [1985], *Group Explicit Iterative Methods for Solving Large Linear Systems*, Inter. J. Computer Math., Vol 17, pp. 81-108.

EVANS, D.J. [1990], *The solution of Periodic Parabolic Equations by The Coupled Alternating Group Explicit (CAGE) Iterative Method*, Inter. J. Computer Math., Vol 34, pp. 227-235.

EVANS, D.J. and **AHMAD A.R.** [1990], *On Improving the Accuracy of The Alternating Group Explicit (AGE) Method*, Computer Studies 572, Loughborough University of Technology.

EVANS, D.J. and **SAHIMI, M.S.** [1988], *The Alternating Group Explicit (AGE) Iterative Method for Solving Parabolic Equation I: 2 Space Dimensional Problems*, Inter. J. Computer Math., Vol 24, pp. 117-142.

EVANS, D.J. and **SAHIMI, M.S.** [1988], *The Alternating Group Explicit (AGE) Iterative Method for Solving Parabolic Equation II: 3 Space Dimensional Problems*, Inter. J. Computer Math., Vol 26, pp. 117-142.

EVANS, D.J. and **YOUSIF, W.S.** [1988], *The Solution of Two-point Boundary Value Problems by the Alternating Group Explicit (AGE) Method*, SIAM J. Sci. Stat. Comput., Vol. 9, No. 3, pp. 474-484.

FADEEVA, D.K. and **FADEEVA, V.N** [1963], *Computational Methods of Linear Algebra*, W.H. Freeman and Company, San Francisco.

FORSYTHE, G.E , MALCOLM, M.A. and **MOLER, C.B.** [1977], *Computer Methods for Mathematical Computations,* Prentice-Hall, New Jersey.

FORSYTHE, G.E and **MOLER, C.B.** [1967], *Computer Solution of Linear Algebraic Systems*, Prentice-Hall, New Jersey.

FORSYTHE, G.E and **WASOW, W.R.** [1960], *Finite Difference methods for Partial Differential Equations*, John Wiley and Sons, New York.

FOX, L. [1950], *The Numerical Solution of Elliptic Differential Equations When Boundary Conditions Involve a Derivatives*, Philos. Trans. Roy. Soc. London, Ser. A 242, pp. 345 - 378.

FOX, L. [1957], *The Numerical Solution of Two-Point Boundary Problems in Ordinary Differential Equations*, Clarendon Press, Oxford.

FOX, L. [1962], *Numerical Solution of Ordinary and Partial Differential Equations*, Pergamon Press, New York.

FOX, L. [1964], *An Introduction to Numerical Linear Algebra*, Clarendon Press, Oxford.

FOX, L. [1977], *Finite-Difference Methods for Elliptic Boundary-Value Problems*, in The State of the Art in Numerical Analysis, ed. Jacobs, D.A.H., Academic Press, London, pp. 799 - 881.

FRIED, I. [1979], *Numerical Solution of Differential Equations*, Academic Press, London.

FROBERG, C.E. [1979], *Introduction to Numerical Analysis*, 2nd Ed., Addison-Wesley Publishing Company, Reading, Mass.

GANE, C.R. [1974], *Computational Techniques for the Numerical Solution of Parabolic and Elliptic Partial Differential Equations*, Ph.D. Thesis, Loughborough University of Technology.

GERALD, C.F. and **WHEATLEY, P.O.** [1984], *Applied Numerical Analysis*, 3rd Ed., Addison-Wesley Publishing Company, Reading, Mass.

GLADWELL, I and **WAIT, R** [1979], *A Survey of Numerical Methods for Partial Differential Equations*, Clarendon Press, Oxford.

GOLUB, G.H. and **VARGA, R.S** [1961], *Chebshev Semi-iterative Methods, Successive Overrelaxation Iterative Methods, and Second Order Richardson Iterative Methods*, Numer. Math., Vol 3, Part I and II, pp. 147 - 168.

GOULT, R.J. et al [1974], *Computational Methods in Linear Algebra*, Stanley Thornes (Publishers) Ltd., London.

GOURLAY, A.R. [1968], *The Acceleration of the Peaceman-Rachford Method by Chebyshev Polynomials*, Comput.Jour, Vol 10, pp. 378 - 382.

GOURLAY, A.R. and **WATSON, G.A.** [1973], *Computational Methods for Matrix Eigenproblems*, John Wiley, London.

GRAHAM, A. [1979], *Matrix Theory and Applications for Engineers and Mathematicians*, John Wiley, New York.

GREENSPAN, D. [1974], *Discrete Numerical Methods in Physics and Engineering*, Academic Press.

GUITTET, J. [1967], *'Une Nouvelle Methode de Directions Alternees a q Variables'*, Journ. Math. Anal. and App., Vol. 17, pp. 199-213.

HAGEMAN, L.A. and **YOUNG, D.M.** [1981], *Applied Iterative Methods*, Academic Press, New York.

HANNA, J.R. [1982], *Fourier Series and Integrals of Boundary Value Problems*, John Wiley and Sons, New York.

HEARN, A.C. [1983], *REDUCE User's Manual*, Version 3.0, The Rand Corporation, Santa Monica.

HILDERBRAND, F.B. [1974], *Introduction to Numerical Analysis*, 2nd Ed., McGraw-Hill, New Delhi.

ISAACSON, E. and **KELLER, H.B.** [1966], *Analysis of Numerical Methods*, John Wiley and Sons, New York.

JAIN, M.K. [1979], *Numerical Solution of Differential Equations*, Wiley Eastern Limited, New Delhi.

JENNINGS, A. [1977], *Matrix Computation for Engineers and Scientists*, John Wiley, Chichester.

JOHNSON, L.W. and **RIESS, R.D.** [1982], *Numerical Analysis*, 2nd Ed., Addison-Wesley Publishing Company, Reading, Mass.

KAHAN, W. [1958], *Gauss-Seidel Methods for Solving Large Systems of Linear Equations*, Doctoral Thesis, University of Toronto, Toronto, Canada.

KAPLAN, W. [1958], *Ordinary Differential Equations*, Addison-Wesley Publishing Company, Reading, Mass.

KELLOG, R.B. [1964], *An Alternating Direction Implicit for Operator Equations*, J. Soc. Indus. Appl. Math., Vol 12, No 4.

KELLOG, R.B. and **Spanier, J.** [1965], *On Optimal Alternating Direction parameter for Singular Matrices*, Maths. Comput., Vol. 19, pp. 448 - 452.

LAMBERT, J.D. [1973], *Computatioanl Methods in Ordinary Differential Equations*, John Wiley and Sons, London.

MARCHUK, G.I. [1975], *Methods of Numerical Mathematics*, Springer-Verlag, New York.

MARCHUK, G.I. and **SHAIDUROV, V.V.** [1983], *Difference Methods and Their Applications*, Springer-Verlag, New York.

MILLER, K.S. [1959], *Partial Differential Equations in Engineering Problems*, Prentice-Hall, New Jersey.

MILNE, W.E. [1970], *Numerical Solution of Differential Equations*, 2nd Ed., Peter Smith Publisher, Gloucester, Mass.

MITCHELL, A.R. [1969], *Computational Methods in Partial Differential Equations*, John Wiley and Sons, London.

MITCHELL, A.R. and **FAIRWEATHER, G.** [1964], *Improved forms of the Alternating Direction methods for Douglas, Peaceman and Rachford for Solving Parabolic and Elliptic Equations*, Numer. Math., Vol. 6, pp. 285-292.

MITCHELL, A.R. and **GRIFFFITH, D.F.** [1980], *The Finite Difference Method in Partial Differential Equations*, John Wiley, New York.

MITCHELL, A.R. and **WAIT, R.** [1977], *The Finite Elements Method in Partial Differential Equations*, John Wiley, New York.

MURPHY, C.P. [1978], *Numerical Methods for Solving Ordinary and Partial Differential Equations*, Ph.D. Thesis, Loughborough University of Technology.

PEACEMAN, D.W. and **RACHFORD, H.H.** [1955], *The Numerical Solution of Parabolic and Elliptic Differential Equations*, J. Soc. Indus. Appl. Math., Vol 3, pp. 28-41.

PEARCY, C. [1962], *On convergence of Alternating Direction Procedures*, Numer. Math., Vol. 4, pp. 172-176.

PRESS, W.H., et. al., [1986], *Numerical Recipes*, The Art of Scientific Computing, Cambridge University Press, Cambridge.

RALSTON, A. [1965], *A First Course in Numerical Analysis*, McGraw-Hill Book Company, New York.

RICHARDSON, L.F. [1910], *The Approximate Arithemetical Solution by Finite Difference of Physical Problems Involving Differential Equations with Application to the Stress in a Masonry Dam*, Philos. Trans. Roy. Soc. London, Ser. A 210, pp. 307-357.

SAHIMI, M.S. [1986], *Numerical Methods for Solving Hyperbolic and Parabolic Differential Equations*, Ph.D. Thesis, Loughborough University of Technology.

SCHEID, F. [1968], *Theory and Problems of Numerical Analysis*, Schaum's Outline Series, McGraw-Hill Book Company, New York.

SHELDON, J. [1955], *On the Numerical Solution of Elliptic Differential Equations*, Math. Tables Aids Comput., Vol. 9, pp. 101-112.

SHORTLEY, G.H. and **WELLER, R.** [1938], *The Numerical Solution of Laplace's Equations*, J. Appl. Phys., Vol 9, pp. 334-348.

SHORTLEY, G.H. [1953], *Use of Chebyshev Polynomial Operators in the Numerical Solution of Boundary Value Problem*, J. Appl. Phys., Vol 24, pp. 392-396.

SMITH, G.D. [1978], *Numerical Solution for Partial Differential Equations: Finite Difference Methods*, 2nd Ed., Oxford University Press, Oxford.

STEPHENSON, G. [1978], *An Introduction to Partial Differential Equations for Science Students*, 2nd Ed., Longman, London.

STEWART, G.W. [1973], *Introduction to Matrix Computation*, Academic Press.

TODD, J. [1962], *Survey of Numerical Analysis*, McGraw-Hill Book Company, New York.

VARGA, R.S. [1962], *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, N.J.

VICHNEVETSKY, R. [1981], *Computer Methods for Partial Differential Equations*, Vol. 1, Prentice-Hall, Englewood Cliffs, N.J.

WACHSPRESS, E.L. [1957], *CURE: A Generalised Two-Space-Dimension, Multi-Group Coding for the IBM-704*, Report KAPL-1724, Knolls Atomic Power Lab., Schenectady, New York.

WACHSPRESS, E.L. and **HABETLER, G.J.** [1960], *An Alternating direction Implicit Iteration Technique*, J. Soc. Indus. Appl. Math., Vol 8, pp. 403-424.

WACHSPRESS, E.L. [1962], *Optimum Alternating Direction Implicit Iteration Parameters for a Model Problem*, J. Soc. Indus. Appl. Math., Vol 10, No. 2, pp. 339-350.

WACHSPRESS, E.L. [1966], *Iterative Solution of Elliptic Systems*, Prentice-HaLL, Englewood Cliffs, N.J.

WAIT, R. [1979], *The Numerical Solution of Algebraic Equations*, John Wiley and Sons, New York.

WILKINSON, J.H. [1965], *The Algebraic Eigenvalue Problem*, Oxford University Press, Oxford.

WOLFRAM, S. [1991], *Mathematica: A System for Doing Mathematics by Computer*, 2nd. Ed., Addison-Wesley, Redwood City, California.

YANENKO, N.N. [1971], *The Method of Fractional Steps*, Springer-Verlag, New York.

YATES, R.C. [1952], *Differential Equations*, 1st Ed., McGraw-Hill Book Company, New York.

YOUNG, D.M. [1954], *Iterative Methods for Solving Partial Differential Equation of Elliptic Type*, Trans. Amer. Math. Soc., Vol. 76 pp. 92-111.

YOUNG, D.M. [1954a], *On Richardson's Method for Solving Linear Systems with Positive Definite Matrices*, J. Math. Phys., Vol XXXII, pp. 243-255.

YOUNG, D.M. [1971], *Iterative Solution for Large Linear System*, Academic Press, New York.

YOUNG, D.M. and **FRANK, T.G.** [1963], *On A Survey of Computer Methods for Solving Elliptic and Parabolic Partial Differential Equations*, Rome Centre Int. de Calcul., Vol 2, No. 1.

YOUNG, D.M. and **GREGORY, R.T.** [1972], *A Survey of Numerical Mathematics*, Vol. I and II, Addison-Wesley, Philippines.

YOUNG, D.M. and **WARLICK, C.H.** [1953], *On the Use of Richardson's Method for the Numerical Solution of Laplace's Equation on the ORDVAC*, Ballistic Research Labs., Memorandum Report No. 707, Aberdeen Proving Ground, Maryland.

YOUNG, D.M. and **EHRLICH, L.W.** [1955], *Some Numerical Studies of Iterative Methods for Solving Elliptic Difference Equations, Boundary Problems in Differential Equations*, Ed. Rudolph E. Langer, pp. 143-157.

YOUSIF, W.S. [1984], *New Block Iterative Methods for the Solution of Boundary Value Problems*, Ph.D. Thesis, Loughborough University of Technology.

WACHSPRESS, E.L. [1963], Extended Application of Alternating Direction Implicit Iteration Model Problem Theory, J. Soc. Indust. Appl. Math., Vol 11, No. 4, pp. 994-1016.

WACHSPRESS, E.L. [1968], Solution of the ADI minimax problems, doctoral thesis, Rensselaer Polytechnic Inst., Troy, New York: p548

APPENDICES

```
c     program AGE for accuracy with F.D. formula
c     Problem 2: u" = 3u2/2,
c     u(0) = 4, u(1) = 1
c     The acceleration parameter is rho
      implicit real*8 (a-h,o-z)
      parameter (kk=502)
      dimension uk1(0:kk), uhalf(0:kk), uk(0:kk), b(1:kk),
     1          g(1:kk), beta(1:kk), alfa(1:kk)
      print *, 'n(even), nrho, strho, rhoinc'
      read *, n, nrho, strho, rhoinc
      eps = 1.0d-08
      h = 1.0/(n+1)
      do 555 i = 1, n
555   b(i) = 0.0
      b(1) = 4.0
      b(n) = 1.0
      write (*, 122) n
122   format (/, 'Number of points', i5 ,/)
      do 55 mk = 1, nrho
      iter = 0
      rho = strho + rhoinc*mk
      do 10 i = 0, n+1
10    uk1(i) = 0.0
c
c     start iterate
c
15    iter = iter + 1
      do 28 i = 0, n+1
28    uk(i) = uk1(i)
      do 556 i = 1, n
      xi = i*h
      g(i) = 1.0 + 3.0*uk(i)*h*h/4.0
      alfa(i) = rho + g(i)
      beta(i) = rho - g(i)
556   continue
c
c     first sweep
c
      do 30 i = 1, n-1, 2
      r1 = b(i) + uk(i-1) + beta(i)*uk(i)
      r2 = b(i+1) + beta(i+1)*uk(i+1) + uk(i+2)
      d = 1.0/(alfa(i)*alfa(i+1) - 1.0)
      uhalf(i) = (alfa(i+1)*r1 + r2)*d
      uhalf(i+1) = (r1 + alfa(i)*r2)*d
30    continue
c
c     second sweep
c
      uk1(1) = (b(1) + beta(1)*uhalf(1) + uhalf(2))/alfa(1)
      do 45 i = 2, n-2, 2
      r1 = b(i) + uhalf(i-1) + beta(i)*uhalf(i)
      r2 = b(i+1) + beta(i+1)*uhalf(i+1) + uhalf(i+2)
      d = 1.0/(alfa(i)*alfa(i+1) - 1.0)
      uk1(i) = (alfa(i+1)*r1 + r2)*d
      uk1(i+1) = (r1 + alfa(i)*r2)*d
45    continue
      uk1(n) = (b(n) + uhalf(n-1) + beta(n)*uhalf(n))/alfa(n)
```

```
c
c     test for convergence
c
      do 50 i = 1, n
      if (dabs(uk1(i) - uk(i))/(1.0 + dabs(uk(i))).gt.eps)
     *                            go to 15
50    continue
c
c     results
c
      print*, 'rho         no. of iter'
      print*, ' '
      write (*, 125) rho, iter
125   format (f7.4, i10)
c     go to 55
      print*, ' '
      print*, ' Problem 2: -u" + 3u**2/2 = 0, u(0) = 4, u(1) = 1'
      print*, ' '
      print*, ' double-precision, tol = 1.0D-08'
      print*, ' usual F.D. formula'
      write(*,112)
112   format (//,'    x       Exact             Computed',
     1        '          Abs error       Prc Trc Err',/)
      do 123 i = 1, n
      xi = i*h
      ftrue = 4.0/(1 + xi)**2
      abserr = dabs(ftrue - uk1(i))
      prcerr = 5*h*h*uk1(i)**3/8.0
      write (*,124) xi, ftrue, uk1(i), abserr, prcerr
124   format (f4.2, 4d17.8)
123   continue
55    continue
      end
```

```
c     Program AGE for accuracy - Numerov formula
c     Problem 2: u" = 3u2/2
c     u(0) = 4, u(1) = 1
c     The acceleration parameter is rho
      implicit real*8 (a-h,o-z)
      parameter (kk=502)
      dimension uk1(0:kk), uhalf(0:kk), uk(0:kk), b(1:kk),
     *     g(1:kk), beta(1:kk), alfa(1:kk), a(1:kk), c(1:kk)
      print *, 'n(even), nrho, strho, rhoinc'
      read *, n, nrho, strho, rhoinc
      eps = 1.0d-08
      h = 1.0/(n+1)
      hh8 = h*h/8.0
      do 555 i = 1, n
555   b(i) = 0.0
      b(1) = b(1) + 4.0*(1.0 - 4.0*hh8)
      b(n) = b(n) + 1.0 - hh8
      write (*, 122) n
122   format (/, 'Number of points', i5 ,/)
      do 55 mk = 1, nrho
      iter = 0
      rho = strho + rhoinc*mk
      do 10 i = 0, n+1
10    uk1(i) = 0.0
c
c     start iterate
c
15    iter = iter + 1
      do 28 i = 0, n+1
28    uk(i) = uk1(i)
      do 556 i = 1, n
      g(i) = 1.0 + 5.0*uk(i)*hh8
      alfa(i) = rho + g(i)
      beta(i) = rho - g(i)
556   continue
      a(1) = 0.0
      do 601 i = 2, n
601   a(i) = hh8*uk(i-1) - 1.0
      do 602 i = 1, n-1
602   c(i) = hh8*uk(i+1) - 1.0
      c(n) = 0.0
c
c     first sweep
c
      do 30 i = 1, n-1, 2
      r1 = b(i) - a(i)*uk(i-1) + beta(i)*uk(i)
      r2 = b(i+1) + beta(i+1)*uk(i+1) - c(i+1)*uk(i+2)
      d = 1.0/(alfa(i)*alfa(i+1) - a(i+1)*c(i))
      uhalf(i) = (alfa(i+1)*r1 - c(i)*r2)*d
      uhalf(i+1) = (-a(i+1)*r1 + alfa(i)*r2)*d
30    continue
c
c     second sweep
c
      uk1(1) = (b(1) + beta(1)*uhalf(1) - c(1)*uhalf(2))/alfa(1)
```

```
      do 45 i = 2, n-2, 2
      r1 = b(i) - a(i)*uhalf(i-1) + beta(i)*uhalf(i)
      r2 = b(i+1) + beta(i+1)*uhalf(i+1) - c(i+1)*uhalf(i+2)
      d = 1.0/(alfa(i)*alfa(i+1) - a(i+1)*c(i))
      uk1(i) = (alfa(i+1)*r1 - c(i)*r2)*d
      uk1(i+1) = (-a(i+1)*r1 + alfa(i)*r2)*d
45    continue
      uk1(n) = (b(n) - a(n)*uhalf(n-1)
     *                        + beta(n)*uhalf(n))/alfa(n)
c
c     test of convergence
c
      do 50 i = 1, n
      if (dabs(uk1(i) - uk(i))/(1.0 + dabs(uk(i))).gt.eps)
     *                       go to 15
50    continue
c
c     results
c
      print*, 'rho         no. of iter'
      print*, ' '
      write (*, 125) rho, iter
125   format (f7.4, i10)
      print*, ' '
      print*, 'Problem 2: -u" + 3u**2/2 = 0, u(0) = 4, u(1) = 1'
      print*, ' '
      print*, 'double-precision, eps = 1.0D-08'
      print*, 'Numerov formula'
      write(*,112)
112   format (//,'    x       Exact         Computed',
     1        '      Abs error     Prc Trc Err',/)
      do 123 i = 1, n
      xi = i*h
      ftrue = 4.0/(1 + xi)**2
      abserr = dabs(ftrue - uk1(i))
      prcerr = 21*h**4*uk1(i)**4/64
      write (*,124) xi, ftrue, uk1(i), abserr, prcerr
124   format (f4.2, 4d17.8)
123   continue
55    continue
      end
```

```
c     program AGE - two point boundary value problem
c     with Douglas-Rachford (D-R) and Douglas Scheme
c     omg = 1 for D-R, omg = 2 for Douglas
c     The acceleration parameter is rho
      implicit real*8 (a-h,o-z)
      parameter (kk=502)
      dimension uk1(0:kk), uhalf(0:kk), uk(0:kk), b(1:kk)
239   print *, ' '
      print *, 'n(even), coeff(real), nrho, strho, rhoinc'
      read *, n, coeff, nrho, strho, rhoinc
      eps = 1.0d-05
      phi = 3.1415927
      halfpi = phi/2
      h = halfpi/(n+1)
      do 555 i = 1, n
      xi = i*h
555   b(i) = (coeff+1)*h*h*(dsin(xi) + dcos(xi))
      b(1) = b(1) + 1.0
      b(n) = b(n) + 1.0
      g = 1.0 + coeff*h*h*0.5
      aa = g - 1
      bb = g + 1
      rhoth = dsqrt(aa*bb)
      write (*, 122) n,coeff
122   format (/, '   Number of points =', i5 ,'    ',
     *           '   Coefficient = ', f7.2,//,
     *           '   Dirichlet Boundary condition',//,
     *           '   Enter Method:', /,
     *           '     1 - Douglas-Rachford Method',/,
     *           '     2 - Douglas Method')
      read (*,*) method
      go to (118,129) method
118   print*, ' '
      print*, '   Douglas-Rachford Method'
      omg = 1.0
      go to 221
129   print*, ' '
      print*, '   Douglas Method'
      omg = 2.0
221   write(*, 218)
218   format (/, '   rho(exp)   rho(th)    a       b     iter',
     *           '      exact(3)       computed(3)',/)
      do 55 mk = 1, nrho
      iter = 0
      rho = strho + rhoinc*mk
      do 10 i = 0, n+1
10    uk1(i) = 0.0
      alfa = rho + g
      d = 1.0/(alfa*alfa - 1.0)
      alfa1 = d*alfa
      t = alfa - 2*g*omg
      ss = omg - 1
15    iter = iter + 1
      do 28 i = 0, n+1
28    uk(i) = uk1(i)
```

```
c
c     first sweep
c
      do 30 i = 1, n-1, 2
      r1 = omg*uk(i-1) + t*uk(i) + ss*uk(i+1) + omg*b(i)
      r2 = ss*uk(i) + t*uk(i+1) + omg*uk(i+2) + omg*b(i+1)
      uhalf(i) = alfa1*r1 + r2*d
      uhalf(i+1) = r1*d + alfa1*r2
30    continue
c
c     second sweep
c
      uk1(1) = (rho*uhalf(1) + g*uk(1))/alfa
      do 45 i = 2, n-2, 2
      r1 = rho*uhalf(i) + g*uk(i) - uk(i+1)
      r2 = rho*uhalf(i+1) - uk(i) + g*uk(i+1)
      uk1(i) = alfa1*r1 + r2*d
      uk1(i+1) = r1*d + alfa1*r2
45    continue
      uk1(n) = (rho*uhalf(n) + g*uk(n))/alfa
c
c     convergence test
c
      do 50 i = 1, n
      if (abs(uk1(i) - uk(i))/(1.0 + abs(uk(i))).gt.eps)
     *                          go to 15
50    continue
c
c     results
c
      uk3 = dsin(3*h) + dcos(3*h)
      write (*, 125) rho, rhoth, aa, bb, iter, uk3, uk(3)
125   format (4f9.3, i6, 2d17.8)
55    continue
      go to 239
      end
```

```
c     program The CAGE Method
c     Problem 3: u" - 3u2/2 = 0
c     u(0)  = 4, u(1) = 1
c     The acceleration parameter is rho
      implicit real*8 (a-h,o-z)
      parameter (kk=502)
      dimension uk1(0:kk), uk(0:kk), b(1:kk), g(1:kk),
     *          alfa(1:kk), res(1:kk), beta(1:kk)
102   print*, ' '
      print *, 'n(even), nrho, strho, rhoinc'
      read *, n, nrho, strho, rhoinc
      eps = 1.0d-05
      h = 1.0/(n+1)
      h75 = 0.75*h*h
      do 555 i = 1, n
555   b(i) = 0.0
      b(1) = 4.0
      b(n) = 1.0
234   write (*, 122) n
122   format (/, '   Number of points =', i5 ,//,
     *           '   The CAGE method - problem 3',/,
     *           '   Dirichlet Boundary condition',//,
     *           '   Enter Scheme:', /,
     *           '     1 - Peaceman-Rachford-(1) Scheme',/,
     *           '     2 - Peaceman-Rachford-(2) Scheme',/,
     *           '     3 - Douglas/Guittet scheme',/,
     *           '     4 - Next point',/,
     *           '     5 - Quit')
      read (*,*) method
      go to (118,129,159,102,169) method
118   print*, ' '
      print*, '  CAGE-Peaceman-Rachford-(1) Scheme'
      write(*,1010)
1010  format(/,'    rho     no of iter     exact(3)',
     *         '          computed(3)',/)
      do 558 mk = 1, nrho
      iter = 0
      rho = strho + rhoinc*mk
      do 1000 i = 0, n+1
1000  uk1(i) = 0.0
1555  iter = iter + 1
      if (iter.gt.200) go to 552
      do 2811 i = 0, n+1
2811  uk(i) = uk1(i)
      do 2581 i = 1, n
      g(i) = 1 + h75*uk(i)
      alfa(i) = rho + g(i)
      beta(i) = rho - g(i)
2581  continue
c
c     The sweep
c
      d = 1.0/(alfa(1)*alfa(2) - 1.0)
      a1 = d*beta(1)*(alfa(2)*beta(1) + 1)/alfa(1)
      b1 = d*beta(2)*(alfa(1) + beta(1))/alfa(1)
```

```
      c1 = d*(alfa(1) + beta(1))/alfa(1)
      d1 = (d*alfa(2)*beta(1) + d + 1)/alfa(1)
      e1 = d1*b(1) + c1*b(2)
      uk1(1) = a1*uk(1) + b1*uk(2) + c1*uk(3) + e1
c
      do 2555 i = 2, n-2, 2
      d1 = 1.0/(alfa(i-1)*alfa(i) - 1.0)
      d2 = 1.0/(alfa(i)*alfa(i+1) - 1.0)
      d3 = 1.0/(alfa(i+1)*alfa(i+2) - 1.0)
      w1 = d1*d2
      w2 = d2*d3
      ak = w1*alfa(i+1)*(alfa(i) + beta(i))
      bk = ak*beta(i-1)
      ck1 = alfa(i-1)*beta(i) + 1
      ck2 = alfa(i+2)*beta(i+1) + 1
      ck = w1*alfa(i+1)*beta(i)*ck1 + w2*ck2
      dk = w1*alfa(i+1)*ck1 + w2*beta(i+1)*ck2
      ek = w2*beta(i+2)*(alfa(i+1) + beta(i+1))
      fk = w2*(alfa(i+1) + beta(i+1))
      pk = w1*alfa(i+1)*(alfa(i) + beta(i))
      qk = alfa(i+1)*(d2 + w1*ck1)
      rk = d2 + w2*ck2
      sk = w2*(alfa(i+1) + beta(i+1))
      gk = pk*b(i-1) + qk*b(i) + rk*b(i+1) + sk*b(i+2)
      uk1(i) = ak*uk(i-2) + bk*uk(i-1) + ck*uk(i)
     *                    + dk*uk(i+1) + ek*uk(i+2)
     *                    + fk*uk(i+3) + gk

      ak = w1*(alfa(i) + beta(i))
      bk = ak*beta(i-1)
      ck = w2*alfa(i)*ck2 + w1*beta(i)*ck1
      dk = w2*alfa(i)*beta(i+1)*ck2 + w1*ck1
      ek = w2*alfa(i)*beta(i+2)*(alfa(i+1) + beta(i+1))
      fk = w2*alfa(i)*(alfa(i+1) + beta(i+1))
      pk = w1*(alfa(i) + beta(i))
      qk = d2 + w1*ck1
      rk = alfa(i)*(d2 + w2*ck2)
      sk = w2*alfa(i)*(alfa(i+1) + beta(i+1))
      gk = pk*b(i-1) + qk*b(i) + rk*b(i+1) + sk*b(i+2)
      uk1(i+1) = ak*uk(i-2) + bk*uk(i-1) + ck*uk(i)
     *                      + dk*uk(i+1) + ek*uk(i+2)
     *                      + fk*uk(i+3) + gk
2555  continue
      d = 1.0/(alfa(n-1)*alfa(n) - 1.0)
      an = d*(alfa(n) + beta(n))/alfa(n)
      bn = d*beta(n-1)*(alfa(n) + beta(n))/alfa(n)
      cn = d*beta(n)*(alfa(n-1)*beta(n) + 1)/alfa(n)
      dn = (d*alfa(n-1)*beta(n) + d + 1)/alfa(n)
      en = an*b(n-1) + dn*b(n)
      uk1(n) = an*uk(n-2) + bn*uk(n-1) + cn*uk(n) + en
c
c     convergence test
c
      do 5250 i = 1, n
      if (abs(uk1(i) - uk(i))/(1.0 + abs(uk(i))).gt.eps)
     *                  go to 1555
5250  continue
```

```
c
c     results
c
      uk13 = 4.0/(1 + 3*h)**2
552   write (*, 125) rho, iter, uk13, uk(3)
125   format (f8.3, i10, 2d20.8)
558   continue
      go to 221
129   print*, ' '
      print*, '  CAGE-Peaceman-Rachford-(2) scheme'
      write(*,2020)
2020  format(/,'    rho     no of iter     exact(3)',
     *         '          computed(3)',/)
      do 5595 mk = 1, nrho
      iter = 0
      rho = strho + rhoinc*mk
      do 100 i = 0, n+1
100   uk1(i) = 0.0
      twor = 2*rho
155   iter = iter + 1
      if (iter.gt.200) go to 5595
      do 281 i = 0, n+1
281   uk(i) = uk1(i)
      do 2881 i = 1, n
      g(i) = 1 + h75*uk(i)
      alfa(i) = rho + g(i)
      beta(i) = rho - g(i)
2881  continue
c
c     The sweep
c
      d = 1.0/(alfa(1)*alfa(2) - 1.0)
      w1 = twor*d
      c1 = w1/alfa(1)
      a1 = beta(1)*(w1*alfa(2) - 1)/alfa(1)
      b1 = c1*beta(2)
      d1 = c1*alfa(2)
      e1 = d1*b(1) + c1*b(2)
      uk1(1) = a1*uk(1) + b1*uk(2) + c1*uk(3) + e1
c
      do 255 i = 2, n-2, 2
      d1 = 1.0/(alfa(i-1)*alfa(i) - 1.0)
      d2 = 1.0/(alfa(i)*alfa(i+1) - 1.0)
      d3 = 1.0/(alfa(i+1)*alfa(i+2) - 1.0)
      w1 = twor*d1*d2
      w2 = twor*d2*d3
      ak = w1*alfa(i+1)
      bk = ak*beta(i-1)
      ck1 = w1*alfa(i-1) - d2
      ck2 = w2*alfa(i+2) - d2
      ck = alfa(i+1)*beta(i)*ck1 + ck2
      dk = alfa(i+1)*ck1 + beta(i+1)*ck2
      ek = w2*beta(i+2)
      qk = ak*alfa(i-1)
```

```
      rk = w2*alfa(i+2)
      fk = ak*b(i-1) + qk*b(i) + rk*b(i+1) + w2*b(i+2)
      uk1(i) = ak*uk(i-2) + bk*uk(i-1) + ck*uk(i)
     *                    + dk*uk(i+1) + ek*uk(i+2)
     *                    + w2*uk(i+3) + fk
      qk = w1*beta(i-1)
      rk = alfa(i)*ck2 + beta(i)*ck1
      sk = ck1 + alfa(i)*beta(i+1)*ck2
      tk = w2*alfa(i)*beta(i+2)
      vk = w2*alfa(i)
      bk = w1*alfa(i-1)
      ck = w2*alfa(i)*alfa(i+2)
      dk = w2*alfa(i)
      fk = w1*b(i-1) + bk*b(i) + ck*b(i+1) + dk*b(i+2)
      uk1(i+1) = w1*uk(i-2) + qk*uk(i-1) + rk*uk(i)
     *                      + sk*uk(i+1) + tk*uk(i+2)
     *                      + vk*uk(i+3) + fk
255   continue
      d = 1.0/(alfa(n-1)*alfa(n) - 1.0)
      wn = twor*d
      an = wn/alfa(n)
      bn = an*beta(n-1)
      cn = beta(n)*(wn*alfa(n-1) - 1)/alfa(n)
      dn = an*alfa(n-1)
      en = an*b(n-1) + dn*b(n)
      uk1(n) = an*uk(n-2) + bn*uk(n-1) + cn*uk(n) + en
c
c     convergence test
c
      do 5350 i = 1, n
      if (abs(uk1(i) - uk(i))/(1.0 + abs(uk(i))).gt.eps)
     *                  go to 155
5350  continue
c
c     results
c
      uk13 = 4.0/(1 + 3*h)**2
      write (*, 125) rho, iter, uk13, uk(3)
5595  continue
      go to 221
159   print*, ' '
      print*, '   CAGE-Douglas/Guittet Scheme'
      write(*,3030)
3030  format(/,'    rho     no of iter     exact(3)',
     *       '          computed(3)',/)
      do 55 mk = 1, nrho
      iter = 0
      rho = strho + rhoinc*mk
      twor = 2*rho
      do 10 i = 0, n+1
10    uk1(i) = 0.0
15    iter = iter + 1
      if (iter.gt.200) go to 515
```

```
      do 7815 i = 0, n+1
7815  uk(i) = uk1(i)
      do 956 i = 1, n
      g(i) = 1 + h75*uk(i)
      alfa(i) = rho + g(i)
956   continue
c
c     The sweep
c
      d = 1.0/(alfa(1)*alfa(2) - 1.0)
      c1 = twor*d/alfa(1)
      a1 = 1 + c1*(1 - 2*alfa(2)*g(1))
      b1 = c1*(alfa(2) - 2*g(2))
      d1 = c1*alfa(2)
      e1 = d1*b(1) + c1*b(2)
      uk1(1) = a1*uk(1) + b1*uk(2) + c1*uk(3) + e1
c
      do 45 i = 2, n-2, 2
      d1 = 1.0/(alfa(i-1)*alfa(i) - 1.0)
      d2 = 1.0/(alfa(i)*alfa(i+1) - 1.0)
      d3 = 1.0/(alfa(i+1)*alfa(i+2) - 1.0)
      w1 = twor*d1*d2
      w2 = twor*d2*d3
      ak = w1*alfa(i+1)
      rk = w2*alfa(i+2)
      qk = ak*alfa(i-1)
      bk = ak*(alfa(i-1) - 2*g(i-1))
      ck = 1 + rk + ak*(1 - 2*alfa(i-1)*g(i))
      dk = qk - 2*rk*g(i+1) + w2
      ek = rk - 2*w2*g(i+2)
      fk = ak*b(i-1) + qk*b(i) + rk*b(i+1) + w2*b(i+2)
c
      uk1(i) = ak*uk(i-2) + bk*uk(i-1) + ck*uk(i)
     *                    + dk*uk(i+1) + ek*uk(i+2)
     *                    + w2*uk(i+3) + fk
c
      qk = w1*(alfa(i-1) - 2*g(i-1))
      rk = w2*alfa(i)*alfa(i+2) - 2*w1*alfa(i-1)*g(i) + w1
      sk = 1 + w1*alfa(i-1)
     *           + w2*alfa(i)*(1 - 2*alfa(i+2)*g(i+1))
      tk = w2*alfa(i)*(alfa(i+2) - 2*g(i+2))
      u1 = w2*alfa(i)
      bk = w1*alfa(i-1)
      ck = w2*alfa(i)*alfa(i+2)
      dk = w2*alfa(i)
      fk = w1*b(i-1) + bk*b(i) + ck*b(i+1) + dk*b(i+2)
c
      uk1(i+1) = w1*uk(i-2) + qk*uk(i-1) + rk*uk(i)
     *                      + sk*uk(i+1) + tk*uk(i+2)
     *                      + u1*uk(i+3) + fk
45    continue
      d = 1.0/(alfa(n-1)*alfa(n) - 1.0)
      an = twor*d/alfa(n)
      bn = an*(alfa(n-1) - 2*g(n-1))
      cn =  1 + an*(1 - 2*alfa(n-1)*g(n))
```

```
      dn = an*alfa(n-1)
      en = an*b(n-1) + dn*b(n)
      uk1(n) = an*uk(n-2) + bn*uk(n-1) + cn*uk(n) + en
c
c     convergence test
c
      do 5359 i = 1, n
      if (abs(uk1(i) - uk(i))/(1.0 + abs(uk(i))).gt.eps)
     *                  go to 15
5359  continue
c
c     results
c
      uk3 = 4.0/(1 + 3*h)**2
515   write (*, 125) rho, iter, uk3, uk(3)
55    continue
221   continue
      go to 234
169   continue
      end
```

```
c     program smartAGE (SMAGE-NONLINEAR)
c     in Peaceman-Rachford form
c     Problem 3: u" = 3u2/2
c     u(0)  = 4, u(1) = 1
c     to measure the CPU time
c     The acceleration parameter is rho
      implicit real*8 (a-h,o-z)
      parameter (kk=502)
      dimension uk1(0:kk), uhalf(0:kk), uk(0:kk), b(1:kk),
     *          psi(0:kk), alfa(1:kk), beta(1:kk), g(1:kk)
921   print *, 'n(even), nrho, strho, rhoinc'
      read *, n, nrho, strho, rhoinc
      eps = 1.0d-05
      h = 1.0/(n+1)
      h75 = 0.75*h*h
      do 555 i = 1, n
555   b(i) = 0.0
      b(1) = 4.0
      b(n) = 1.0
      write (*, 122) n
122   format (/, '   Number of points =', i5 ,'   ',//,
     *           '   Dirichlet Boundary condition',/,
     *           '   SMAGE in Peaceman-Rachford form',//,
     *           '     rho    no of iter     exact(3)',
     *           '          computed(3)      CPU time',/)
      do 55 mk = 1, nrho
      iter = 0
      rho = strho + rhoinc*mk
      twor = 2*rho
      do 10 i = 0, n+1
10    uk1(i) = 0.0
      call_clock_time(itime1)
15    iter = iter + 1
      if (iter.gt.650) go to 1111
      do 28 i = 0, n+1
28    uk(i) = uk1(i)
      do 2581 i = 1, n
      g(i) = 1 + h75*uk(i)
      alfa(i) = rho + g(i)
      beta(i) = rho - g(i)
2581  continue
      do 30 i = 1, n-1, 2
      psi(i) = uk(i-1) + beta(i)*uk(i)
      psi(i+1) = beta(i+1)*uk(i+1) + uk(i+2)
30    continue
c
c     first sweep
c
      do 300 i = 1, n-1, 2
      r1 = psi(i) + b(i)
      r2 = psi(i+1) + b(i+1)
      d = 1.0/(alfa(i)*alfa(i+1) - 1.0)
      uhalf(i) = (alfa(i+1)*r1 + r2)*d
      uhalf(i+1) = (r1 + r2*alfa(i))*d
300   continue
```

```
c
c     second sweep
c
      do 304 i = 1, n
304   psi(i) = - psi(i) + twor*uhalf(i)
      uk1(1) = psi(1)/alfa(1)
      do 45 i = 2, n-2, 2
      d = 1.0/(alfa(i)*alfa(i+1) - 1.0)
      uk1(i) = (alfa(i+1)*psi(i) + psi(i+1))*d
      uk1(i+1) = (psi(i) + psi(i+1)*alfa(i))*d
45    continue
      uk1(n) = psi(n)/alfa(n)
c
c     convergence test
c
      do 50 i = 1, n
      if (abs(uk1(i) - uk(i))/(1.0 + abs(uk(i))).gt.eps)
     *                    go to 15
50    continue
      call_clock_time(itime2)
      time3 = real(itime2 - itime1)/100.00
c
c     results
c
1111  uk3 = 4.0/(1 + 3*h)**2
      write (*, 125) rho, iter, uk(3), uk3, time3
125   format (f8.2, i10, 2d19.8,f10.2)
55    continue
      go to 921
      end
```

```
c     The AGE-DG-2 scheme for Helmholtz's (2D) equation
c     the acceleration parameter rho
      implicit real*8 (a-h,o-z)
      parameter (kk=99)
      dimension uk1(0:kk,0:kk), uk34(0:kk,0:kk),
     *          uk12(0:kk,0:kk), uk14(0:kk,0:kk),
     *          uk(0:kk,0:kk), exact(0:kk,0:kk),
     *          b(1:kk,1:kk)
1111  print *, 'n(odd), coeff(real)'
      read *, n, coeff
      eps = 1.0d-05
      h = 1.0/(n+1)
      nn = n*n
c
c     the vector b
c
      do 3555 j = 1, n
      yj = j*h
      do 3555 i = 1, n
      xi = i*h
      terms = coeff*(2*xi*xi + yj*yj) - 6
3555  b(i,j) = h*h*terms
      do 3155 i = 1, n
      xi = i*h
      b(i,1) = b(i,1) + 2*xi*xi
      b(i,n) = b(i,n) + 1 + 2*xi*xi
3155  continue
      do 3255 j = 1, n
      yj = j*h
      b(1,j) = b(1,j) + yj*yj
      b(n,j) = b(n,j) + 2 + yj*yj
3255  continue
c
c     the exact value for the problem
c
      do 314 j = 1, n
      yj = j*h
      do 314 i = 1, n
      xi = i*h
      exact(i,j) = 2*xi*xi + yj*yj
314   continue
c
c     choose method
c
1113  write (*, 122) nn, coeff
122   format (/, '   Solving PDE (2D) - Helmholtz equation',/,
     *           '   Elliptic type using:',/,
     *           '   Number of points =',i5,/,
     *           '   Coefficient = 'f6.2,/,
     *           '   Enter option:',/,
     *           '      1- The AGE-DG-2 Scheme:',/,
     *           '      2- Change point:',/,
     *           '      3- Quit',/)
```

```
      read*, nopt
      go to (1112,1111,1119) nopt
1112  omg = 2.0
      write (*, 222) nn, coeff
222   format (/, '    Number of points =',i5,
     *          '    Coefficient = 'f6.2,/)
      g = 1.0 + coeff*h*h*0.25
      aa = g - 1.0
      bb = g + 1.0
      cc = (aa+bb)/2
      rhoth = dsqrt(aa*bb)
      print *, 'nrho, strho, rhoinc'
      read *, nrho, strho, rhoinc
      write (*, 256)
256   format(/,  '    rho    iter',
     *          '     exact          computed',
     *          '          a     b   (a+b)/2  sqrt(ab)',/)
      do 55 mk = 1, nrho
      iter = 0
      rho = strho + rhoinc*mk
      do 11 j = 0, n+1
      do 11 i = 0, n+1
11    uk1(i,j) = 0.0
      alfa = rho + g
      d = 1.0/(alfa*alfa - 1.0)
      t = alfa - 4*g*omg
      alfa1 = alfa*d
      ss = omg-1
15    iter = iter + 1
      if (iter.gt.250) go to 1211
      do 28 j = 0, n+1
      do 28 i = 0, n+1
28    uk(i,j) = uk1(i,j)
c
c     level k+1/4
c
      do 10 j = 1, n
      do 20 i = 1, n-2, 2
      r1 = omg*uk(i,j-1) + omg*uk(i-1,j) + t*uk(i,j)
     *        + ss*uk(i+1,j) + omg*uk(i,j+1) + omg*b(i,j)
      r2 = omg*uk(i+1,j-1) + ss*uk(i,j) + t*uk(i+1,j)
     *        + omg*uk(i+2,j) + omg*uk(i+1,j+1) + omg*b(i+1,j)
      uk14(i,j) = alfa1*r1 + r2*d
      uk14(i+1,j) = r1*d + alfa1*r2
20    continue
      uk14(i,j) = (omg*uk(i,j-1) + omg*uk(i-1,j) + t*uk(i,j)
     *               + omg*uk(i,j+1) + omg*b(i,j))/alfa
10    continue
c
c     level k+1/2
c
      do 90 j = 1, n
      uk12(1,j) = (rho*uk14(1,j) + g*uk(1,j))/alfa
```

```
      do 90 i = 2, n-1, 2
      r1 = rho*uk14(i,j) + g*uk(i,j) - uk(i+1,j)
      r2 = rho*uk14(i+1,j) - uk(i,j) + g*uk(i+1,j)
      uk12(i,j)  = alfa1*r1 + r2*d
      uk12(i+1,j) = r1*d + alfa1*r2
90    continue
c
c     level k+3/4
c
      do 900 i = 1, n
      do 910 j = 1, n-2, 2
      r1 = rho*uk12(i,j) + g*uk(i,j) - uk(i,j+1)
      r2 = rho*uk12(i,j+1) - uk(i,j) + g*uk(i,j+1)
      uk34(i,j) = alfa1*r1 + r2*d
      uk34(i,j+1) = r1*d + alfa1*r2
910   continue
      uk34(i,j) = (rho*uk12(i,j) + g*uk(i,j))/alfa
900   continue
c
c     level k+1
c
      do 920 i = 1, n
      uk1(i,1) = (rho*uk34(i,1) + g*uk(i,1))/alfa
      do 920 j = 2, n-1, 2
      r1 = rho*uk34(i,j) + g*uk(i,j) - uk(i,j+1)
      r2 = rho*uk34(i,j+1) - uk(i,j) + g*uk(i,j+1)
      uk1(i,j) = alfa1*r1 + r2*d
      uk1(i,j+1) = r1*d + alfa1*r2
920   continue
c
c     test for convergence
c
      do 550 j = 1, n
      do 550 i = 1, n
      if (dabs(uk1(i,j) - uk(i,j))/(1.0 + dabs(uk(i,j)))
     *                    .gt.eps) go to 15
550   continue
1211  write (*,192) rho,iter,exact(9,9), uk1(9,9),
     *              aa, bb, cc, rhoth
192   format (f7.3, i6, 2d16.8, 4f8.3)
55    continue
      go to 1113
1119  continue
      end
```

```
c     AGE for PDE Elliptic (2D)
c     nonstationary - 2 parameter case, Hueristic search
c     The acceleration parameters are called rho-1 and rho-2
      implicit real*8 (a-h,o-z)
      parameter (kk=99)
      dimension uk1(0:kk,0:kk), uk34(0:kk,0:kk),
     *          uk12(0:kk,0:kk), uk14(0:kk,0:kk),
     *          uk(0:kk,0:kk), exact(0:kk,0:kk),
     *          b(1:kk,1:kk)
      print *, 'Enter n(odd)'
      read *, n
      eps = 1.0d-05
      h = 1.0/(n+1)
      phi = 3.1415927
      halfpi = phi/2
      chlpi = dcosh(halfpi)
      nn = n*n
      do 555 j = 1, n
      do 555 i = 1, n
555   b(i,j) = 0.0
      print *, ' '
      print *, 'Enter Problem: 1 - 3'
      read (*,*) iprob
      go to (1,2,3) iprob
c
c     Problem 1
c
1     do 1555 i = 1, n
      xi = i*h
      b(i,1) = dsin(phi*xi)
      b(i,n) = dsin(phi*xi)
1555  continue
c
c     the exact value - Problem 1
c
      do 114 j = 1, n
      yj = j*h
      do 114 i = 1, n
      xi = i*h
      exact(i,j) = dcosh(phi*(yj-0.5))*dsin(phi*xi)/chlpi
114   continue
      go to 6
c
c     Problem 2
c
2     do 2555 i = 1, n
      xi = i*h
      b(i,n) = 400*xi
2555  continue
      do 2255 j = 1, n-1
      yj = j*h
      b(n,j) = 400*yj
2255  continue
      yn = n*h
      b(n,n) = b(n,n) + 400*yn
```

```
c
c     the exact value - Problem 2
c
      do 214 j = 1, n
      yj = j*h
      do 214 i = 1, n
      xi = i*h
      exact(i,j) = 400*xi*yj
214   continue
      go to 6
c
c     Problem 3
c
3     do 3555 j = 1, n
      do 3555 i = 1, n
3555  b(i,j) = 2*h*h
      do 3155 i = 1, n
      xi = i*h
      b(i,1) = b(i,1) + xi*(1-xi)
      b(i,n) = b(i,n) + xi*(1-xi)
3155  continue
      do 3255 j = 1, n
      yj = j*h
      b(n,j) = b(n,j) + dsinh(phi)*dsin(phi*yj)
3255  continue
c
c     the exact value - Problem 3
c
      do 314 j = 1, n
      yj = j*h
      do 314 i = 1, n
      xi = i*h
      exact(i,j) = dsinh(phi*xi)*dsin(phi*yj) + xi*(1-xi)
314   continue
c
c     choose method
c
6     write (*, 122) nn, iprob
122   format (/, '   Number of points =', i5,//,
     *           '   Solving PDE (2D) - Problem',i2,/,
     *           '   2 - parameter, Heuristic search',/,
     *           '   Elliptic type using:',/,
     *           '   The AGE-DG-2 Scheme:',)
      omg = 2.0
221   print*,' '
      print*,'   Enter rho-1, rho-2, rinc'
      read (*,*) rho1, rho2, rinc
      write (*,229)
229   format (/,'    Increment: 1 - rho1, 2 - rho2',/)
      read (*,*) irho
      write (*,329)
329   format (/,'     rho-1     rho-2       iters',
     *          '        exact              computed',/)
```

```
105   go to (101,102) irho
101   rho1 = rho1 + rinc
      go to 108
102   rho2 = rho2 + rinc
108   iter = 0
      do 11 j = 0, n+1
      do 11 i = 0, n+1
11    uk1(i,j) = 0.0
      icyc = 0
15    iter = iter + 1
      if (iter.gt.200) go to 105
      icyc = icyc + 1
      go to (201,202) icyc
201   rho = rho1
      go to 991
202   rho = rho2
991   alfa = 1.0 + rho
      d = 1.0/(alfa*alfa - 1.0)
      t = alfa - 4*omg
      alfa1 = alfa*d
      ss = omg - 1
      do 28 j = 0, n+1
      do 28 i = 0, n+1
28    uk(i,j) = uk1(i,j)
c
c     level k+1/4
c
      do 10 j = 1, n
      do 20 i = 1, n-2, 2
      r1 = omg*uk(i,j-1) + omg*uk(i-1,j) + t*uk(i,j)
     *        + ss*uk(i+1,j) + omg*uk(i,j+1) + omg*b(i,j)
      r2 = omg*uk(i+1,j-1) + ss*uk(i,j) + t*uk(i+1,j)
     *        + omg*uk(i+2,j) + omg*uk(i+1,j+1) + omg*b(i+1,j)
      uk14(i,j) = alfa1*r1 + r2*d
      uk14(i+1,j) = r1*d + alfa1*r2
20    continue
      uk14(i,j) = (omg*uk(i,j-1) + omg*uk(i-1,j) + t*uk(i,j)
     *                + omg*uk(i,j+1) + omg*b(i,j))/alfa
10    continue
c
c     level k+1/2
c
      do 90 j = 1, n
      uk12(1,j) = (rho*uk14(1,j) + uk(1,j))/alfa
      do 90 i = 2, n-1, 2
      r1 = rho*uk14(i,j) + uk(i,j) - uk(i+1,j)
      r2 = rho*uk14(i+1,j) - uk(i,j) + uk(i+1,j)
      uk12(i,j) = alfa1*r1 + r2*d
      uk12(i+1,j) = r1*d + alfa1*r2
90    continue
```

```
c
c     level k+3/4
c
      do 900 i = 1, n
      do 910 j = 1, n-2, 2
      r1 = rho*uk12(i,j) + uk(i,j) - uk(i,j+1)
      r2 = rho*uk12(i,j+1) - uk(i,j) + uk(i,j+1)
      uk34(i,j) = alfa1*r1 + r2*d
      uk34(i,j+1) = r1*d + alfa1*r2
910   continue
      uk34(i,j) = (rho*uk12(i,j) + uk(i,j))/alfa
900   continue
c
c     level k+1
c
      do 920 i = 1, n
      uk1(i,1) = (rho*uk34(i,1) + uk(i,1))/alfa
      do 920 j = 2, n-1, 2
      r1 = rho*uk34(i,j) + uk(i,j) - uk(i,j+1)
      r2 = rho*uk34(i,j+1) - uk(i,j) + uk(i,j+1)
      uk1(i,j) = alfa1*r1 + r2*d
      uk1(i,j+1) = r1*d + alfa1*r2
920   continue
      if (icyc.eq.2) icyc = 0
c
c     test for convergence
c
      do 550 j = 1, n
      do 550 i = 1, n
      if (dabs(uk1(i,j) - uk(i,j))/(1.0 + dabs(uk(i,j)))
     *                    .gt.eps) go to 15
550   continue
      write (*,129) rho1,rho2,iter,exact(9,9),uk1(9,9)
129   format (2f10.3, i10, 2d19.8)
      go to 105
      end
```

```
c     AGE for PDE Elliptic (3D) - 1st problem
c     second order - Chebyshev extrapolation
c     The acceleration parameters are rho and wk1
      implicit real*8 (a-h,o-z)
      parameter (kk=20)
      dimension uk1(0:kk,0:kk,0:kk), vk(0:kk,0:kk,0:kk),
     *          wk(0:kk,0:kk,0:kk), b(1:kk,1:kk,1:kk),
     *          exact(0:kk,0:kk,0:kk), uk(0:kk,0:kk,0:kk),
     *          uk0(0:kk,0:kk,0:kk)
      print *, 'n(odd), nrho, strho, rhoinc'
      read *, n, nrho, strho, rhoinc
      eps = 1.0d-05
      phi = 3.1415927
      h = 1.0/(n+1)
      nnn = n*n*n
c
c     the vector b
c
      do 555 k = 1, n
      do 555 j = 1, n
      do 555 i = 1, n
555   b(i,j,k) = 0.0
      do 2555 k = 1, n
      zk = k*h
      do 2555 i = 1, n
      xi = i*h
      b(i,1,k) = dsin(phi*xi)*dsin(phi*zk)
      b(i,n,k) = dsin(phi*xi)*dsin(phi*zk)
2555  continue
c
c     the exact value
c
      tr1 = dcosh(phi/sqrt(2.0))
      tr2 = phi*sqrt(2.0)
      do 114 k = 1, n
      zk = k*h
      do 114 j = 1, n
      yj = j*h
      do 114 i = 1, n
      xi = i*h
      exy = dsin(phi*xi)*dcosh(tr2*(yj - 0.5))
      exact(i,j,k) = exy*dsin(phi*zk)/tr1
114   continue
      write (*, 122), nnn
122   format (/, '   Solving PDE (3D) - 1st problem',/,
     *           '   Number of points =',i5,//,
     *           '   Elliptic type using:',/,
     *           '   The Douglas Method:',/,
     *           '       Rho            No of Iter',/)
1000  print*,'   maxeig'
      read*, eigmax
      kmin = 1000
      do 55 mk = 1, nrho
      iter = 1
      rho = strho + rhoinc*mk
```

```
      do 11 k = 0, n+1
      do 11 j = 0, n+1
      do 11 i = 0, n+1
11    uk(i,j,k) = 0.0
      alfa = rho + 1.0
      d = 1.0/(alfa*alfa - 1.0)
      alfa1 = alfa*d
      rho5 = rho**5
      coeff1 = 2*rho5
c
c     to compute uk1 = Muk + g
c
c     the calculation vk = (rI+G1)-1(b-Au)
c
      do 107 k = 1, n
      do 107 j = 1, n
      do 207 i = 1, n-2, 2
      r1 = uk(i,j,k-1) + uk(i,j-1,k) + uk(i-1,j,k)
     *             - 6*uk(i,j,k) + uk(i+1,j,k) + uk(i,j+1,k)
     *             + uk(i,j,k+1) + b(i,j,k)
      r2 = uk(i+1,j,k-1) + uk(i+1,j-1,k) + uk(i,j,k)
     *             - 6*uk(i+1,j,k) + uk(i+2,j,k) + uk(i+1,j+1,k)
     *             + uk(i+1,j,k+1) + b(i+1,j,k)
      vk(i,j,k) = alfa1*r1 + r2*d
      vk(i+1,j,k) = r1*d + alfa1*r2
207   continue
      vk(n,j,k) = (uk(n,j,k-1) + uk(n,j-1,k) + uk(n-1,j,k)
     *                     - 6*uk(n,j,k) + uk(n,j+1,k)
     *                     + uk(n,j,k+1) + b(n,j,k))/alfa
107   continue
c
c     wk = (rI+G2)-1vk
c
      do 907 k = 1, n
      do 907 j = 1, n
      wk(1,j,k) = vk(1,j,k)/alfa
      do 907 i = 2, n-1, 2
      wk(i,j,k) = alfa1*vk(i,j,k) + vk(i+1,j,k)*d
      wk(i+1,j,k) = vk(i,j,k)*d + alfa1*vk(i+1,j,k)
907   continue
c
c     vk(y) = (rI+G1)-1wk(y)
c
      do 9007 k = 1, n
      do 9007 i = 1, n
      do 9107 j = 1, n-2, 2
      vk(i,j,k) = alfa1*wk(i,j,k) + wk(i,j+1,k)*d
      vk(i,j+1,k) = wk(i,j,k)*d + alfa1*wk(i,j+1,k)
9107  continue
      vk(i,n,k) = wk(i,n,k)/alfa
9007  continue
```

```
c
c     wk(y) = (rI+G2)-1vk(y)
c
      do 9207 k = 1, n
      do 9207 i = 1, n
      wk(i,1,k) = vk(i,1,k)/alfa
      do 9207 j = 2, n-1, 2
      wk(i,j,k) = alfa1*vk(i,j,k) + vk(i,j+1,k)*d
      wk(i,j+1,k) = vk(i,j,k)*d + alfa1*vk(i,j+1,k)
9207  continue
c
c     vk = (rI+G1)-1wk(z)
c
      do 9507 i = 1, n
      do 9507 j = 1, n
      do 9407 k = 1, n-2, 2
      vk(i,j,k) = alfa1*wk(i,j,k) + wk(i,j,k+1)*d
      vk(i,j,k+1) = wk(i,j,k)*d + alfa1*wk(i,j,k+1)
9407  continue
      vk(i,j,n) = wk(i,j,n)/alfa
9507  continue
c
c     uk1 = uk + dm*rho**5*(rI+G2)-1vk(z)
c
      do 9307 i = 1, n
      do 9307 j = 1, n
      uk1(i,j,1) = uk(i,j,1) + coeff1*vk(i,j,1)/alfa
      do 9307 k = 2, n-1, 2
      uk1(i,j,k) = uk(i,j,k) + coeff1*(alfa1*vk(i,j,k)
     *                               + d*vk(i,j,k+1))
      uk1(i,j,k+1) = uk(i,j,k+1) + coeff1*(d*vk(i,j,k)
     *                               + alfa1*vk(i,j,k+1))
9307  continue
      wk1 = 1.0/(1 - eigmax*eigmax/2)
15    iter = iter + 1
      if (iter.gt.58) go to 553
      do 28 k = 0, n+1
      do 28 j = 0, n+1
      do 28 i = 0, n+1
      uk0(i,j,k) = uk(i,j,k)
      uk(i,j,k) = uk1(i,j,k)
28    continue
c
c     to compute uk2 = Muk1 + Cuk0
c
c     the calculation vk = (rI+G1)-1(b-Au)
c
      do 10 k = 1, n
      do 10 j = 1, n
      do 20 i = 1, n-2, 2
      r1 = uk(i,j,k-1) + uk(i,j-1,k) + uk(i-1,j,k)
     *           - 6*uk(i,j,k) + uk(i+1,j,k) + uk(i,j+1,k)
     *           + uk(i,j,k+1) + b(i,j,k)
      r2 = uk(i+1,j,k-1) + uk(i+1,j-1,k) + uk(i,j,k)
     *           - 6*uk(i+1,j,k) + uk(i+2,j,k) + uk(i+1,j+1,k)
     *           + uk(i+1,j,k+1) + b(i+1,j,k)
```

```
      vk(i,j,k) = alfa1*r1 + r2*d
      vk(i+1,j,k) = r1*d + alfa1*r2
20    continue
      vk(n,j,k) = (uk(n,j,k-1) + uk(n,j-1,k) + uk(n-1,j,k)
     *                     - 6*uk(n,j,k) + uk(n,j+1,k)
     *                     + uk(n,j,k+1) + b(n,j,k))/alfa
10    continue
c
c     wk = (rI+G2)-1vk
c
      do 90 k = 1, n
      do 90 j = 1, n
      wk(1,j,k) = vk(1,j,k)/alfa
      do 90 i = 2, n-1, 2
      wk(i,j,k) = alfa1*vk(i,j,k) + vk(i+1,j,k)*d
      wk(i+1,j,k) = vk(i,j,k)*d + alfa1*vk(i+1,j,k)
90    continue
c
c     vk(y) = (rI+G1)-1wk(y)
c
      do 900 k = 1, n
      do 900 i = 1, n
      do 910 j = 1, n-2, 2
      vk(i,j,k) = alfa1*wk(i,j,k) + wk(i,j+1,k)*d
      vk(i,j+1,k) = wk(i,j,k)*d + alfa1*wk(i,j+1,k)
910   continue
      vk(i,n,k) = wk(i,n,k)/alfa
900   continue
c
c     wk(y) = (rI+G2)-1vk(y)
c
      do 920 k = 1, n
      do 920 i = 1, n
      wk(i,1,k) = vk(i,1,k)/alfa
      do 920 j = 2, n-1, 2
      wk(i,j,k) = alfa1*vk(i,j,k) + vk(i,j+1,k)*d
      wk(i,j+1,k) = vk(i,j,k)*d + alfa1*vk(i,j+1,k)
920   continue
c
c     vk(z) = (rI+G1)-1wk(z)
c
      do 950 i = 1, n
      do 950 j = 1, n
      do 940 k = 1, n-1, 2
      vk(i,j,k) = alfa1*wk(i,j,k) + wk(i,j,k+1)*d
      vk(i,j,k+1) = wk(i,j,k)*d + alfa1*wk(i,j,k+1)
940   continue
      vk(i,j,n) = wk(i,j,n)/alfa
950   continue
c
c     uk1(z) = wk1*(uk(z) + dm*rho**5*(rI+G6)-1vk(z))
c                                       + (1-wk1)*uk0(z)
      wk2 = 1-wk1
      do 930 i = 1, n
      do 930 j = 1, n
      uk1(i,j,1) = wk1*(uk(i,j,1) + coeff1*vk(i,j,1)/alfa)
     *                             + wk2*uk0(i,j,1)
```

```
      do 930 k = 2, n-1, 2
      uk1(i,j,k) = wk1*(uk(i,j,k) + coeff1*(alfa1*vk(i,j,k)
     *                + d*vk(i,j,k+1))) + wk2*uk0(i,j,k)
      uk1(i,j,k+1) = wk1*(uk(i,j,k+1) + coeff1*(d*vk(i,j,k)
     *                + alfa1*vk(i,j,k+1))) + wk2*uk0(i,j,k+1)
930   continue
      wk1 = 1.0/(1 - eigmax*eigmax*wk1/4)
c
c     test for convergence
c
      do 550 k = 1, n
      do 550 j = 1, n
      do 550 i = 1, n
      if (dabs(uk1(i,j,k) - uk(i,j,k))/(1.0 + dabs(uk(i,j,k)))
     *              .gt.eps) go to 15
550   continue
553   write (*,129) rho,eigmax,wk1,iter,uk1(9,9,9),exact(9,9,9)
129   format (3f8.3, i6, 2d18.8)
      if (kmin.lt.iter) go to 1000
      kmin = iter
55    continue
      go to 1000
      end
```

```
c     The EAD-GT-3 scheme for PDE Elliptic (3D)
c     with Guittet formula for omg = 2
c     The acceleration parameters are called rho and s
      implicit real*8 (a-h,o-z)
      parameter (kk=29)
      dimension uk(0:kk,0:kk,0:kk), uk13(0:kk,0:kk,0:kk),
     *          uk23(0:kk,0:kk,0:kk), uk1(0:kk,0:kk,0:kk),
     *          up(0:kk,0:kk,0:kk), up12(0:kk,0:kk,0:kk),
     *          exact(0:kk,0:kk,0:kk), b(1:kk,1:kk,1:kk),
     *          b1(1:kk,1:kk,1:kk)
      print *, 'n(odd), nrho, strho, rhoinc'
      read *, n, nrho, strho, rhoinc
      eps = 1.0d-05
      h = 1.0/(n+1)
      phi = 3.1415927
      halfpi = phi/2
      nnn = n*n*n
      do 555 k = 1, n
      do 555 j = 1, n
      do 555 i = 1, n
555   b(i,j,k) = 0.0
      print *,' '
      print *,' Enter problem: 1 - 2'
      read (*,*) iprob
      go to (1,2) iprob
c
c     Problem 1
c
1     do 1555 k = 1, n
      zk = k*h
      do 1555 i = 1, n
      xi = i*h
      b(i,1,k) = dsin(phi*xi)*dsin(phi*zk)
      b(i,n,k) = dsin(phi*xi)*dsin(phi*zk)
1555  continue
c
c     the exact value - Problem 1
c
      tr1 = dcosh(phi/sqrt(2.0))
      tr2 = phi*sqrt(2.0)
      do 114 k = 1, n
      zk = k*h
      do 114 j = 1, n
      yj = j*h
      do 114 i = 1, n
      xi = i*h
      exy = dsin(phi*xi)*dcosh(tr2*(yj-0.5))
      exact(i,j,k) = exy*dsin(phi*zk)/tr1
114   continue
      go to 6
c
c     Problem 2
c
2     do 2555 j = 1, n
      yj = j*h
```

```
      do 2555 i = 1, n
      xi = i*h
      b(i,j,n) = 400*xi*yj + b(i,j,n)
2555  continue
      do 2255 k = 1, n
      zk = k*h
      do 2255 i = 1, n
      xi = i*h
      b(i,n,k) = 400*xi*zk + b(i,n,k)
2255  continue
      do 2355 k = 1, n
      zk = k*h
      do 2355 j = 1, n
      yj = j*h
      b(n,j,k) = 400*yj*zk + b(n,j,k)
2355  continue
c
c     the exact value - Problem 2
c
      do 214 k = 1, n
      zk = k*h
      do 214 j = 1, n
      yj = j*h
      do 214 i = 1, n
      xi = i*h
      exact(i,j,k) = 400*xi*yj*zk
214   continue
c
c     choose method
c
6     write (*, 122) iprob
122   format (/, '   Solving PDE (3D) - Problem',i2,/,
     *           '   Elliptic type using:',/,
     *           '   Guittet formula for omg = 2:',/,
     *           '   The EAD-GT-3 Scheme:',/)
888   print*, 'sl'
      read*, sl
      write (*, 222) nnn
222   format (/, '    Number of points =',i5,//,
     *           '       rho        iter',
     *           '        exact              computed',/)
      do 55 mk = 1, nrho
      iter = 0
      rho = strho + rhoinc*mk
      do 11 k = 0, n+1
      do 11 j = 0, n+1
      do 11 i = 0, n+1
      up12(i,j,k) = 0.0
      uk13(i,j,k) = 0.0
      uk23(i,j,k) = 0.0
11    uk1(i,j,k) = 0.0
```

```
c
c     coefficients
c
      omg = 2.0
      alfa = rho + 2
      dmr2 = omg*rho*rho
      dms = 2*s1
      beta = s1 + alfa/2
      d = 1.0/(beta*beta - 1.0)
      btd = beta*d
      p1 = alfa*alfa*alfa - 6*dmr2
      q1 = dmr2 - alfa*alfa
      p2 = beta*beta - alfa*dms
      q2 = dms - beta
      p2b = p2/beta
      q2b = q2/beta
      dmsb = dms/beta
15    iter = iter + 1
      if (iter.gt.58) go to 789
      do 28 k = 0, n+1
      do 28 j = 0, n+1
      do 28 i = 0, n+1
28    uk(i,j,k) = uk1(i,j,k)
c
c     rhs vector b1 = [(rI+X)*(rI+Y)*(rI+Z) - mr2A]uk + mr2b
c
      do 900 k = 1, n
      do 900 j = 1, n
      do 900 i = 1, n
      b1(i,j,k) = - uk(i-1,j-1,k-1) - uk(i+1,j-1,k-1)
     *            - uk(i-1,j+1,k-1) - uk(i+1,j+1,k-1)
     *            - uk(i-1,j-1,k+1) - uk(i+1,j-1,k+1)
     *            - uk(i-1,j+1,k+1) - uk(i+1,j+1,k+1)
     *            + alfa*(uk(i,j-1,k-1) + uk(i-1,j,k-1)
     *            + uk(i+1,j,k-1) + uk(i,j+1,k-1)
     *            + uk(i-1,j-1,k) + uk(i+1,j-1,k)
     *            + uk(i-1,j+1,k) + uk(i+1,j+1,k)
     *            + uk(i,j-1,k+1) + uk(i-1,j,k+1)
     *            + uk(i+1,j,k+1) + uk(i,j+1,k+1))
     *            + q1*(uk(i,j,k-1) + uk(i,j-1,k)
     *            + uk(i-1,j,k) + uk(i+1,j,k)
     *            + uk(i,j+1,k) + uk(i,j,k+1))
     *            + p1*uk(i,j,k) + dmr2*b(i,j,k)
900   continue
c
c     level k+1/3
c
      do 281 k = 0, n+1
      do 281 j = 0, n+1
      do 281 i = 0, n+1
281   up(i,j,k) = uk13(i,j,k)
c
c     up12 = (sI+G1)-1[(sI+G1)*(sI+G2) - msX]up + msb1]
c
```

```
      do 90 k = 1, n
      do 90 j = 1, n
      do 901 i = 1, n-2, 2
      r1 = q2*(up(i-1,j,k) + up(i+1,j,k)) + p2*up(i,j,k)
     *                     + up(i+2,j,k) + dms*b1(i,j,k)
      r2 = up(i-1,j,k) + q2*(up(i,j,k) + up(i+2,j,k))
     *                 + p2*up(i+1,j,k) + dms*b1(i+1,j,k)
      up12(i,j,k) = btd*r1 + r2*d
      up12(i+1,j,k) = r1*d + btd*r2
901   continue
      up12(n,j,k) = q2b*up(n-1,j,k) + p2b*up(n,j,k)
     *                               + dmsb*b1(n,j,k)
90    continue
c
c     uk13 = (sI+G2)-1(up12)
c
      do 80 k = 1, n
      do 80 j = 1, n
      uk13(1,j,k) = up12(1,j,k)/beta
      do 80 i = 2, n-1, 2
      uk13(i,j,k) = btd*up12(i,j,k) + up12(i+1,j,k)*d
      uk13(i+1,j,k) = up12(i,j,k)*d + btd*up12(i+1,j,k)
80    continue
c
c     level k+2/3
c
      do 2811 k = 0, n+1
      do 2811 j = 0, n+1
      do 2811 i = 0, n+1
2811  up(i,j,k) = uk23(i,j,k)
c
c     up12 = (sI+G1)-1[(sI+G1)*(sI+G2) - msX]up(y) + ms*uk13
c
      do 98 i = 1, n
      do 98 k = 1, n
      do 981 j = 1, n-2, 2
      r1 = q2*(up(i,j-1,k) + up(i,j+1,k)) + p2*up(i,j,k)
     *                     + up(i,j+2,k) + dms*uk13(i,j,k)
      r2 = up(i,j-1,k) + q2*(up(i,j,k) + up(i,j+2,k))
     *                 + p2*up(i,j+1,k) + dms*uk13(i,j+1,k)
      up12(i,j,k) = btd*r1 + r2*d
      up12(i,j+1,k) = r1*d + btd*r2
981   continue
      up12(i,n,k) = q2b*up(i,n-1,k) + p2b*up(i,n,k)
     *                               + dmsb*uk13(i,n,k)
98    continue
c
c     uk13 = (sI+G2)-1(up12(y))
c
      do 800 i = 1, n
      do 800 k = 1, n
      uk23(i,1,k) = up12(i,1,k)/beta
      do 800 j = 2, n-1, 2
      uk23(i,j,k) = btd*up12(i,j,k) + up12(i,j+1,k)*d
      uk23(i,j+1,k) = up12(i,j,k)*d + btd*up12(i,j+1,k)
800   continue
```

```
c
c     level k+1
c
      do 3811 k = 0, n+1
      do 3811 j = 0, n+1
      do 3811 i = 0, n+1
3811  up(i,j,k) = uk1(i,j,k)
c
c     up12 = (sI+G1)-1[(sI+G1)*(rI+G2) - msX]up(z) + ms*uk23
c
      do 99 j = 1, n
      do 99 i = 1, n
      do 999 k = 1, n-2, 2
      r1 = q2*(up(i,j,k-1) + up(i,j,k+1)) + p2*up(i,j,k)
     *                     + up(i,j,k+2) + dms*uk23(i,j,k)
      r2 = up(i,j,k-1) + q2*(up(i,j,k) + up(i,j,k+2))
     *                 + p2*up(i,j,k+1) + dms*uk23(i,j,k+1)
      up12(i,j,k) = btd*r1 + r2*d
      up12(i,j,k+1) = r1*d + btd*r2
999   continue
      up12(i,j,n) = q2b*up(i,j,n-1) + p2b*up(i,j,n)
     *                              + dmsb*uk23(i,j,n)
99    continue
c
c     uk1 = (sI+G2)-1(up12(z))
c
      do 8000 j = 1, n
      do 8000 i = 1, n
      uk1(i,j,1) = up12(i,j,1)/beta
      do 8000 k = 2, n-1, 2
      uk1(i,j,k) = btd*up12(i,j,k) + up12(i,j,k+1)*d
      uk1(i,j,k+1) = up12(i,j,k)*d + btd*up12(i,j,k+1)
8000  continue
c
c     test for convergence
c
      do 550 k = 1, n
      do 550 j = 1, n
      do 550 i = 1, n
      if (dabs(uk1(i,j,k) - uk(i,j,k))/(1.0
     *      + dabs(uk(i,j,k))).gt.eps) go to 15
550   continue
789   write (*,192) rho, iter, exact(9,9,9), uk1(9,9,9)
192   format (f10.3, i10, 3d19.8)
55    continue
      go to 888
      end
```