Computational and algorithmic techniques for the solution of elliptic and parabolic partial differential equations in two and three space dimensions

This item was submitted to Loughborough University's Institutional Repository by the/an author.

Additional Information:

- A Doctoral Thesis. Submitted in partial fulfillment of the requirements for the award of Doctor of Philosophy of Loughborough University.

Metadata Record: [https://dspace.lboro.ac.uk/2134/7935](https://dspace.lboro.ac.uk/2134/7935)

Publisher: © Elias Anastasios Lipitakis

Please cite the published version.
This item is held in Loughborough University’s Institutional Repository (https://dspace.lboro.ac.uk/) and was harvested from the British Library’s EThOS service (http://www.ethos.bl.uk/). It is made available under the following Creative Commons Licence conditions.

For the full text of this licence, please go to:
http://creativecommons.org/licenses/by-nc-nd/2.5/

Disclaimer: ©
COMPUTATIONAL AND ALGORITHMIC TECHNIQUES
FOR THE SOLUTION OF ELLIPTIC AND PARABOLIC
PARTIAL DIFFERENTIAL EQUATIONS
IN TWO AND THREE SPACE DIMENSIONS

by

Elias Anastasios Lipitakis, B.Sc.

A Doctoral Thesis
Submitted in partial fulfilment of the requirements
for the award of Doctor of Philosophy
of the Loughborough University of Technology

Supervisor:  Professor D.J. Evans, Ph.D., D.Sc.
Department of Computer Studies.

DECLARATION

I declare that the following thesis is a record of research work carried out by me, and that the thesis is of my own composition. I also certify that neither this thesis nor the original work contained therein has been submitted to this or any other institution for a degree.

E.A. LIPITAKIS.
DEDICATED TO

Christina and Alexandra
ACKNOWLEDGEMENTS

The author wishes to express his deep gratitude and appreciation to Professor D.J. Evans, Head of the Department of Computer Studies for introducing him to the subject as well as for his continuous advice, guidance and encouragement at every stage of this research work.

He also expresses his thanks to Dr. A. Benson, for valuable comments; to the staff of the Computing laboratory of Loughborough University of Technology, where the bulk of the computing was carried out, for their assistance and co-operation; and to Miss J.M. Briers for her expert typing of this thesis.

Finally, grateful acknowledgements are due to my Parents, Anastasios and Alexandra Lipitakis, and Father-in-Law, D.S. Georgacopoulos, for their affection, encouragement and help; and to my Wife, Christina, and Daughter, Alexandra, for more than can be expressed.
CONTENTS

CHAPTER 1: BASIC MATHEMATICAL CONCEPTS

1.1 Partial Differential Equations - Basic Concepts .......................... 1
1.2 Finite Difference Approximations to Derivatives ......................... 5
1.3 Basic Matrix Properties and Concepts ...................................... 10
1.4 Vector and Matrix Norms ..................................................... 13
1.5 Eigenvalues of a Matrix ..................................................... 15
1.6 Basic Definitions ............................................................. 16

CHAPTER 2: ELLIPTIC AND PARABOLIC EQUATIONS IN TWO AND THREE
SPACE DIMENSIONS - STANDARD METHODS OF SOLUTION

Section A: The Elliptic Problem
2.1 Introduction ........................................................................... 18
2.2 The Model Problems ............................................................ 23
2.3 Some Point Iterative Methods ................................................. 26
2.4 The Conjugate Gradient Method ............................................. 40

Section B: The Parabolic Problem
2.5 A Parabolic 2D-Example ....................................................... 45
2.6 Derivation of Finite Difference Formulae ................................. 46
2.7 Explicit and Implicit Methods ............................................... 47
2.8 A Parabolic 3D-Example ....................................................... 54

CHAPTER 3: BASIC ALGORITHMS FOR TWO AND THREE DIMENSIONAL P.D.E.'S

3.1 Introduction ............................................................................. 58
3.2 The Algorithm for the solution of large, unsymmetric, quindia
gonal, sparse linear systems (The LUBOT Algorithm) ...................... 58
3.3 The Normalized Algorithm for the solution of large, symmetric, quindia
gonal, sparse linear systems (The NORMBAND Algorithm) .............. 65
3.4 The Algorithm for the solution of large, unsymmetric, seven diagonal, sparse linear systems (The LUBOT-3D Algorithm) ............... 67
3.5 The Normalized Algorithm for the solution of large, symmetric, seven diagonal, sparse linear systems (The NB3D Algorithm) ............... 75
CHAPTER 4: APPROXIMATE ALGORITHMS FOR TWO AND THREE DIMENSIONAL P.D.E.'S

4.1 Introduction ............................................................. 82
4.2 The Approximate Algorithm for the solution of large, unsymmetric, quindagonal, sparse linear systems (The ALUBOT Algorithm) .................. 82
4.3 The Approximate Normalized Algorithm for the solution of large, symmetric, quindagonal, sparse linear systems (The NOBAR Algorithm) .... 85
4.4 The Approximate Algorithm for the solution of large, unsymmetric seven diagonal, sparse linear systems (The ALUBOT-3D Algorithm) ........ 89
4.5 The Approximate Normalized Algorithm for the solution of large, symmetric, seven diagonal, sparse linear systems (The NOBAR-3D Algorithm) .. 94

CHAPTER 5: NORMALIZED IMPLICIT CONJUGATE GRADIENT METHODS

5.1 Introduction ............................................................. 100
5.2 The Optimum Value of the "Fill-in" Parameter r for the 2D-Model Problem .................................................. 100
5.3 The Conjugate Gradient Method as an Iterative Method ................................................................. 108
5.4 The Normalized Implicit Conjugate Gradient Method ................................................................. 110
5.5 Computational Work and Experimental Results in Two and Three Space Dimensions ................................................................. 114

CHAPTER 6: NORMALIZED SECOND ORDER METHODS

6.1 Introduction ............................................................. 125
6.2 The Conjugate Gradient Method as a Second Degree Method ................................................................. 125
6.3 The Normalized Implicit Conjugate Gradient Method as a Second Degree Method ................................................................. 127
6.4 A Semi-Empirical Procedure for the Normalized Conjugate Gradient Second Degree Methods ................................................................. 138
6.5 Derivation of Standard Normalized Implicit Methods and Standard Implicit Methods ................................................................. 144
CHAPTER 1

BASIC MATHEMATICAL CONCEPTS
1.1 PARTIAL DIFFERENTIAL EQUATIONS - BASIC CONCEPTS

The numerical treatment of many problems in Mathematical Physics or Engineering, involving the rates of change of unknown quantities (derivatives) with respect to two, three or more independent variables, leads to the solution of a Partial Differential Equation (P.D.E.) or a set of such equations.

Let \( R \) denote a bounded, connected, plane region with a boundary \( C \) consisting of one or more differential curves and \( \gamma(x, y) \) a function defined on \( C \). Let \( U(x, y) \) be a function continuous in \( R+C \), twice differentiable in \( R \) and satisfying in \( R \) the general second order P.D.E.

\[
a \frac{\partial^2 U}{\partial x^2} + b \frac{\partial^2 U}{\partial x \partial y} + c \frac{\partial^2 U}{\partial y^2} + d \frac{\partial U}{\partial x} + e \frac{\partial U}{\partial y} + fU = h \quad (1.1.1)
\]

and on \( C \) the Dirichlet condition

\[
U(x, y) = \gamma(x, y) \quad (1.1.2)
\]

(Alternatively on \( C \), the inward normal derivative \( \frac{\partial U}{\partial z} \) (Neumann condition) or a linear combination of \( U \) and \( \frac{\partial U}{\partial z} \) (Robbins condition) may be specified).

The exact solution \( U(x, y) \) to the above P.D.E. in region \( R \) satisfies the equation (1.1.1) at every point in \( R \) and matches the 'boundary conditions' on \( C \). The coefficients of (1.1.1) may be constants (including the value zero) or functions of independent variables \( x \) and \( y \) (linear case) or of the dependent variable \( U \) and its first derivatives (non-linear case).

Equations of form (1.1.1) are conventionally classified with respect to the sign of the quantity of discriminant \( \Delta = b^2 - 4ac \).

Specifically, it is defined to be

- elliptic when \( b^2 - 4ac < 0 \) (1.1.3a)
- parabolic when \( b^2 - 4ac = 0 \) (1.1.3b)
- hyperbolic when \( b^2 - 4ac > 0 \) (1.1.3c)

for all \( x, y, U \) in the region \( R \) under consideration.
Typical and respective examples are:

\[
\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = \begin{cases} 
 g(x,y), & \text{Poisson's equation} \\
 0, & \text{Laplace's equation}
\end{cases}
\quad (1.1.4a)
\]

\[
\frac{\partial U}{\partial t} - c^2 \frac{\partial^2 U}{\partial x^2} = 0, \quad \text{Diffusion or Heat conduction equation}
\quad (1.1.4c)
\]

\[
\frac{\partial^2 U}{\partial t^2} - c^2 \frac{\partial^2 U}{\partial x^2} = 0, \quad \text{Wave equation}
\quad (1.1.4d)
\]

If the quantity \( b^2 - 4ac \) depends only upon \((x,y)\), the type of equation at a point is completely determined by the co-ordinates of the point. When the coefficients of \((1.1.1)\) depend upon the dependent variable \( U \) and its first derivatives, the nature of the equations at a point \((x,y)\) depends not only upon the location of that point, but also upon the behaviour of the solution itself at that point.

For example,

(a) The equation

\[
(1-y) \frac{\partial^2 U}{\partial x^2} + 2x \frac{\partial^2 U}{\partial x \partial y} + (1+y) \frac{\partial^2 U}{\partial y^2} = h
\quad (1.1.5)
\]

is elliptic inside the unit circle \( x^2 + y^2 = 1 \), parabolic on the boundary and hyperbolic outside the circle.

(b) The nature of the equation

\[
\frac{\partial^2 U}{\partial x^2} + U \frac{\partial^2 U}{\partial y^2} = h
\quad (1.1.6)
\]

at a point depends upon the sign of \( U \) at that point and, in any specific problem, depends also upon the side conditions which serve to complete the formulation of that problem.

The two elliptic P.D.E's \((1.1.4a)-(1.1.4b)\) are generally associated with steady-state or equilibrium problems. For example,

(i) the electric potential, associated with a two dimensional electron distribution of charge density, satisfies Poisson's
equation and is the mathematical expression of the total electric flux through any closed surface is equal to the total charge enclosed, whereas,

(ii) the velocity potential for the steady flow of incompressible non-viscous fluid satisfies Laplace's equation, stating that the rate at which such fluid enters any given region is equal to the rate at which it leaves it. Such problems are referred to as 'boundary value problems', since the dependent variable is usually specified on the boundary of the region under consideration, e.g. see system (1.1.1) (1.1.2). Boundary value problems frequently occur in applications such as reservoir problems, reactor studies, numerical weather forecasting, etc.

Parabolic and hyperbolic P.D.E's, in general, result from diffusion, equalization or oscillatory processes and the usual independent variables are time and space.

The simplest parabolic P.D.E. given by (1.1.4c) governs the flow of heat in a thin homogeneous bar or rod, assuming that the radiation and convection are neglected. The temperature distribution along the bar or rod is usually known at some instant in time. This is termed the 'initial condition'. The boundary conditions consist of appropriate end conditions, which are either the temperature given at two ends of the bar or rod or some measure of the diffusion from the ends. Such problems are called 'initial-boundary value problems' (sometimes only the terminology 'initial value problem' is used).

A major class of hyperbolic equations arise from vibration problems or those in which discontinuities persist in time. In the case of the simplest hyperbolic equation (1.1.4d), giving the transverse displacement at a given distance from one end of a
vibrating string of given length after a certain time, the initial conditions, usually \( U \) and \( \frac{\partial U}{\partial t} \), are given at some instant in time and the boundary conditions are given on two lines \((x=a,b)\).

For the solution of problems with arbitrarily shaped regions and general boundary conditions, an exact solution to a given P.D.E. is not usually possible to determine. Only in the simplest cases can a solution be analytical either in implicit form or that involving a finite formula.

The 'approximate methods', which have been developed to tackle this problem, can be divided into two groups:

(i) **Analytical approximate methods**, in which the approximate solution is in an analytical form, e.g. the truncating of an infinite series. They consist mainly of Fourier's method of solving boundary value problems in P.D.E's, where the exact solution is in the form of a certain infinite series and the approximate solution is the sum of the first few terms.

(ii) **Numerical approximate methods**, in which approximate values of the required solution can be found at various points of a region under consideration in a tabular form. The most widely used numerical method for solving P.D.E's, applicable to both linear and non-linear problems, is the 'finite difference' method, which will be exclusively used throughout this thesis. The method of 'characteristics' for solving hyperbolic equations and sets of equations is also essentially a finite difference method, only in this method the P.D.E. or set of P.D.E's is first reduced to an equivalent set of Ordinary Differential Equations (O.D.E's) or linear algebraic equations, which is then solved by numerical direct or iterative methods.
1.2 FINITE DIFFERENCE APPROXIMATIONS TO DERIVATIVES

We consider, without any loss of generality, that the problem is to solve the elliptic equation (1.14a) with independent variables \( x, y \) in a connected region \( R \), in the \( X-Y \) plane. Let \( \overline{R}=R+C \) be the closure of the considered region \( R \) with boundary \( C \). We overlay \( \overline{R} \) with a system of rectangular mesh lines formed by two families of equally spaced straight lines, which are parallel to coordinate axes viz.,

\[
\begin{align*}
x &= x_0 + ih, \quad i=0, \pm 1, \pm 2, \ldots, \\
y &= y_0 + jh, \quad j=0, \pm 1, \pm 2, \ldots,
\end{align*}
\]

where \((x_0, y_0)\) is any conveniently chosen origin for the "mesh coordinates" \( i,j \).

The intersection points are called mesh points (grid or lattice points, nodes, pivots) and the distances between the parallel lines are called mesh sizes (lengths).

A mesh point \( P_{ij} \) is called 'regular' if the four adjacent points \( P_{i+1,j}, P_{i-1,j}, P_{i,j+1}, P_{i,j-1} \) (see Fig. 1.1) are also mesh points contained in \( \overline{R} \); otherwise \( P_{ij} \) is called 'irregular'.

![Figure 1.1](image-url)
An approximate solution to the differential equation is then found at the \( n \) mesh points \( P_{1,1}, P_{1,2}, \ldots, P_{1,j}, \ldots \) and the problem is now reduced to the solution of \( n \)-algebraic equations (linear, if the differential equations are linear) involving approximate values of \( U \) at the \( n \)-mesh points internal to \( C \).

**Taylor Series Expansion**

This method is probably the best known of all for deriving finite difference approximations.

We assume that the solution of \((1.11)\) has the required continuous derivatives of higher order in a sufficiently large neighbourhood about a point \((x,y)\) and a constant mesh size \( h \), then by Taylor's theorem the values at the surrounding mesh points can be determined, viz.,

\[
U(x \pm h, y) = U(x, y) \pm h \frac{\partial U}{\partial x} + \frac{h^2}{2} \frac{\partial^2 U}{\partial x^2} \pm \cdots \quad (1.2.1)
\]

\[
U(x, y \pm h) = U(x, y) \pm h \frac{\partial U}{\partial y} + \frac{h^2}{2} \frac{\partial^2 U}{\partial y^2} \pm \cdots \quad (1.2.2)
\]

Combining these formulae we obtain

\[
\frac{\partial U}{\partial x} = \frac{U(x+h, y) - U(x-h, y)}{2h} + O(h^2) \quad (1.2.3)
\]

\[
\frac{\partial^2 U}{\partial x^2} = \frac{U(x+h, y) - 2U(x, y) + U(x-h, y)}{h^2} + O(h^2) \quad (1.2.4)
\]

and

\[
\frac{\partial U}{\partial y} = \frac{U(x, y+h) - U(x, y-h)}{2h} + O(h^2) \quad (1.2.5)
\]

\[
\frac{\partial^2 U}{\partial y^2} = \frac{U(x, y+h) - 2U(x, y) + U(x, y-h)}{h^2} + O(h^2) \quad (1.2.6)
\]

where the \( O \)-notation denotes that if \( S \) is any set and \( f, g \) be real or complex functions defined on \( S \) then,

\[
f(t) = O(\phi(t)) \quad \text{as} \quad t \to a \quad \text{with} \quad t, a \in S \quad (1.2.7a)
\]

if there exists a positive number \( P \), such that

\[
|f(t)| \leq P|\phi(t)| \quad (1.2.7b)
\]
for all \( t \) sufficiently close to \( a \).

Let us denote \( U(x,y) \) by \( U_{i,j} \), for a general mesh point \((x,y) = (ih, jh)\). Then, Poisson's equation

\[
\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = g(x,y),
\]

using the formulae (1.2.4), (1.2.6), can be replaced at the point \((x_i, y_j)\) by

\[
\frac{1}{h^2} (U_{i+1,j} + U_{i-1,j} + U_{i,j+1} + U_{i,j-1} - 4U_{i,j}) = g_{i,j} + \frac{h^2}{12} \left( \frac{\partial^4 U}{\partial x^4} + \frac{\partial^4 U}{\partial y^4} \right)_{i,j} + \ldots \tag{1.2.8}
\]

or equivalently,

\[
4U_{i,j} - U_{i+1,j} - U_{i-1,j} - U_{i,j+1} - U_{i,j-1} = -h^2 g_{i,j} - \frac{h^4}{12} \left( \frac{\partial^4 U}{\partial x^4} + \frac{\partial^4 U}{\partial y^4} \right)_{i,j} - \ldots \tag{1.2.9}
\]

As "local truncation error" of the above formula are defined the terms on the right hand side of (1.2.9), excluding \(-h^2 g_{i,j}\), while the term \( O(h^4) \) is defined as the 'principal part' of this error. The local truncation error in (1.2.9) is usually neglected and then scanning over the mesh points with such a formula, a set of simultaneous equations can be obtained, with unknown functions \( u_{i,j} \), the finite difference approximation of the exact solution \( U_{i,j} \) at the point \((ih, jh)\).

The above set of equations can be written in matrix form as

\[
A \cdot u = s \tag{1.2.10}
\]

where \( s \) is a vector composed of the known values \(-h^2 g_{i,j}\) plus values of the finite difference approximate solution \( u_{i,j} \) given on the boundary \( C \). Note that if \((x_1, y_1) \in C\) then \( u_{1,j} = U_{1,j} \).

Assuming that the region under consideration is the unit
square and there are \( n^2 \) internal mesh points, then the coefficient matrix \( A \) of (1.2.10) is of order \( n^2 \), while \( u \) and \( s \) are \((n^2 \times 1)\) column vectors. From (1.2.9) it can be easily seen that the better accuracy is obtained as the mesh size \( h \) tends to zero.

We consider now the following initial-boundary value problem defined by

\[
\frac{\partial U}{\partial t} = \frac{\partial^2 U}{\partial x^2} \tag{1.2.11}
\]

in the region

\[
0 \leq x \leq a ; \quad t \geq 0
\]

with initial conditions

\[
U(x,0) = f(x) ; \quad 0 \leq x \leq a \tag{1.2.11a}
\]

and boundary conditions consisting of

\[
U(0,t) = g_0(t), \quad U(a,t) = g_a(t) ; \quad t \geq 0. \tag{1.2.11b}
\]

We cover the region under consideration by a rectangular grid with spacings \( h_X \) and \( \Delta t \) in the \( X \) and \( t \) (time)-directions respectively (see Figure 1.2).

Let \( U_{i,j}^{(k)} \) and \( u_{i,j}^{(k)} \) denote the exact and finite difference solution respectively of equation (1.2.11) at the point \((ih_X, k\Delta t)\).

Then, using Taylor series expansions, a simple replacement of (1.2.11)
leads to the following difference scheme:

\[
\frac{U_{i}^{(k+1)} - U_{i}^{(k)}}{\Delta t} = \frac{U_{i+1}^{(k)} - 2U_{i}^{(k)} + U_{i-1}^{(k)}}{h^{2}} + O(\Delta t + \frac{h^{2}}{\Delta t})
\]  

(1.2.12)

which can be written as

\[
U_{i}^{(k+1)} = (1 - 2r)U_{i}^{(k)} + r(U_{i+1}^{(k)} + U_{i-1}^{(k)}) + O(\Delta t^{2} + \Delta t \cdot h^{2})
\]  

(1.2.13)

where \( r = \frac{\Delta t}{h^{2}} \) is called the 'mesh ratio'. Note that \( U_{i}^{(k+1)} \) is expressed in (1.2.13) solely in terms of the values of \( U \) at the \( k^{th} \)-time level. Such schemes are called explicit (open) whereas schemes involving more than one point at the \((k+1)^{th}\) time level are called implicit (closed). Assuming that the local truncation error in (1.2.13) is neglected, then scanning over each mesh point in turn, in the interval \( 0 < x < a \), we obtain a set of simultaneous equations, which can be expressed in matrix notation as

\[
\mathbf{u}^{(k+1)} = A \mathbf{u}^{(k)} , \quad k \geq 0 .
\]  

(1.2.14)

An analogous implicit scheme could be written as

\[
A_{1} \mathbf{u}^{(k+1)} = A_{2} \mathbf{u}^{(k)} , \quad k \geq 0 .
\]  

(1.2.15)

The numerical integration of the differential equation (1.2.11) is obtained by a step by step procedure represented by both equations (1.2.14), (1.2.15). This process is continued until the final solution \( \mathbf{u}^{T} \) is obtained at some time \( T = rt \Delta t \).

It can be seen that the application of finite difference method to the numerical solution of P.D.E.'s leads to the solution of a system of simultaneous equations, such as (1.2.10), (1.2.15). The coefficient matrix of these systems is usually a large "sparse" (with many zero elements) matrix, with a certain number of non-zero elements in each row, and with characteristic properties such as irreducibility, diagonal dominance, positive definiteness etc., which will be defined in the next section.
Note that the structure of the coefficient matrix is of major importance for the method of solution of the resulting system of simultaneous equations and several techniques for this purpose will be presented in Chapters 3 and 4.

In the following section a basic knowledge of matrix and linear algebra theory has been presupposed, with the subject being found in [3], [26], [29], [57], [61].

1.3 BASIC MATRIX PROPERTIES AND CONCEPTS

Definition 1.3.1

The matrix \( A = (a_{i,j}) \) of order \( n \) is said to be 'irreducible' if \( n=1 \) or if \( n>1 \) and given any two non-empty disjoint subsets \( S \) and \( T \) of \( W \), the set of the first \( n \)-positive integers, such that \( S+T=W \), there exists some \( a_{i,j} \neq 0 \) such that \( i \in S \) and \( j \in T \).

The following alternative definition is given in [67, (p.37)]:

Definition 1.3.2

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

\[
P^{-1}AP = \begin{bmatrix} A_1 & 0 \\ A_2 & A_3 \end{bmatrix}
\]

where \( A_1 \) and \( A_3 \) are square matrices and where all elements of \( 0 \) vanish.

Note that a permutation matrix is a square matrix which in each row and each column has some one entry unity and all others zero.

Definition 1.3.3

An \((n \times n)\) matrix \( A = (a_{i,j}) \) is said to be "diagonally dominant"
if
\[ |a_{i,i}| \geq \sum_{j=1 \atop j \neq i}^{n} |a_{i,j}|, \text{ for all } i \in [1,n]. \] (1.3.1)

The matrix \( A \) is 'strictly diagonally dominant' if strict inequality is valid for all \( i \in [1,n] \) in relationship (1.3.1).

If \( A \) is irreducible and diagonally dominant, with strict inequality in (1.3.1) for at least one value of \( i \), then \( A \) is said to be "irreducibly diagonally dominant".

**Definition 1.3.4**

If \( A \) is real matrix and \( x \) is complex, then \( A \) is said to be 'positive definite' matrix if
\[ (x, Ax) > 0, \text{ for all } x \neq 0. \]

(Note that if \( x, y \) are complex, then
\[ (x, y) = \sum_{i=1}^{n} x_i \overline{y}_i, \text{ where } \overline{y}_i \text{ is the complex conjugate of } y_i. \] The quantity \((x, y)\) is called the inner-product of \( x, y \).)

**Definition 1.3.5**

If \( A \) is real matrix and \( x \) is complex, then \( A \) is said to be 'non-negative (or positive semi)-definite' matrix if
\[ (x, Ax) \geq 0 \text{ for all } x \neq 0, \]
with equality for at least one \( x \neq 0 \).

The following theorem, given without proof, can be used as an alternative definition of positive/non-negative definite matrix.

**Theorem 1.3.1**

A real matrix is positive/non-negative definite if and only if
all its eigenvalues are positive/(non-negative, with at least one eigenvalue equal to zero).

Let \( A \) be a positive definite matrix. The matrix \( A \) can be
written as $A = GJG^{-1}$, where $J$ is a positive diagonal matrix and $G$ can be taken to be an orthogonal matrix i.e., $G^T = G^{-1}$ (see [61](p.16)).

Then, by Theorem 1.3.1 we obtain that

$$A^{\frac{1}{2}} = GJ^{\frac{1}{2}}G^{-1}$$

(1.3.2)

is a positive definite, provided that $J^{\frac{1}{2}}$ is the diagonal matrix whose elements are the positive square roots of the corresponding elements of $J$. It can be easily seen that $(A^{\frac{1}{2}})^2 = (GJ^{\frac{1}{2}}G^{-1})^2 = A$.

**Theorem 1.3.2**

A real symmetric matrix $A$ is positive/non-negative definite if and only if it can be written in the form $A = G^T G$, where $G$ is some non-singular/singular matrix.

**Proof**

(I) Let us assume that $A = G^T G$ (det $G \neq 0$).

Then, for all $x \neq 0$, we have

$$x^TAx = x^T G^T Gx = (Gx)^T Gx > 0.$$  

Hence, by Definition 1.3.4, the matrix $A$ is positive definite.

(II) Let $A$ be real and positive definite matrix.

Since $A = A^{\frac{1}{2}} A^{\frac{1}{2}}$ and $A^{\frac{1}{2}}$ is symmetric, we have that $A = (A^{\frac{1}{2}})^T A^{\frac{1}{2}}$,

where $A^{\frac{1}{2}}$ is defined in (1.3.2) and is also positive definite.

Hence $\det A^{\frac{1}{2}} \neq 0$ and then it can be easily seen that the choice $G = A^{\frac{1}{2}}$ leads to the required conclusion.

If $A$ is non-negative definite matrix, the proof can be similarly obtained.

**Definition 1.3.6**

A real $(n \times n)$ matrix $A = (a_{i,j})$, with all its off-diagonal elements non-positive ($a_{i,j} \leq 0$, for all $i \neq j$) is said to be:

(i) a "Stieltjes matrix" if $A$ is symmetric and positive definite,

(ii) a "M-matrix" if $A$ is non-singular and $A^{-1} \succeq 0$. 
1.4 VECTOR AND MATRIX NORMS

Definition 1.4.1

If \(x = (x_1, x_2, \ldots, x_n)\) is an \(n\)-component vector, then the following quantities are defined as the \(L_1\), \(L_2\) and \(L_\infty\) norms of \(x\) respectively:

\[
||x||_1 = \sum_{i=1}^{n} |x_i| \tag{1.4.1}
\]

\[
||x||_2 = \left( \sum_{i=1}^{n} |x_i|^2 \right)^{1/2} \text{ (Euclidean or } L_2 \text{ Norm or length of } x) \tag{1.4.2}
\]

\[
||x||_\infty = \max_{i \in [1,n]} |x_i| \text{ (Maximum or Uniform Norm)} \tag{1.4.3}
\]

Definition 1.4.2

A matrix norm is said to be 'compatible' with a vector norm \(||x||\) if

\[
||Ax|| \leq ||A|| \cdot ||x||, \text{ for all } x \neq 0.
\]

Definition 1.4.3

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

\[
||A|| = \sup_{x \neq 0} \frac{||Ax||}{||x||} \tag{1.4.4}
\]

or equivalently:

\[
||A|| = \sup_{||y||=1} ||Ay|| \tag{1.4.5}
\]

with the matrix norm satisfying the compatibility relation

\[
||Ay|| \leq ||A|| \cdot ||y||.
\]

Definition 1.4.4

Let \(A\) be an \(n \times n\) matrix with eigenvalues \(\lambda_i, i \in [1,n]\) then

the "spectral radius" of \(A\) is defined as

\[
\rho(A) = \max_{i \in [1,n]} |\lambda_i|.
\]

It can be easily shown that for any matrix and any norm, we have

\[
||A|| \geq \rho(A). \tag{1.4.6}
\]
Proof
Let us assume that $Ax = \lambda x_i, i \in [1, n]$ is the equation associating the non-zero eigenvector $x$ with the eigenvalues $\lambda_i$ of $A$. Then, from Definition 1.4.2 and for any eigenvalue $\lambda_i$ of $A$, we have

$$|\lambda_i| |x_i| = |\lambda x_i| = |Ax_i| |x_i|,$$

which leads to (1.4.6).

Definition 1.4.5

Let $A$ be a square matrix of order $n$. Then, the subordinate norms associated with $L_1$, $L_2$ and $L_\infty$ vector Norms are defined as:

$$||A||_1 = \max_{j \in [1, n]} \sum_{i=1}^{n} |a_{i,j}| \quad \text{(Maximum absolute column sum)} \quad (1.4.7)$$

$$||A||_2 = \rho(A^T A)^{\frac{1}{2}} \quad \text{(Spectral Norm)} \quad (1.4.8)$$

$$||A||_\infty = \max_{i \in [1, n]} \sum_{j=1}^{n} |a_{i,j}| \quad \text{(Maximum absolute row sum)} \quad (1.4.9)$$

It can be easily shown from (1.4.8) that if $A$ is a symmetric $n \times n$ matrix, then we get

$$||A||_2 = \rho(A) \quad (1.4.10)$$

Proof

Given that $A$ is symmetric we have,

$$||A||_2^2 = \rho(A^T A) = \rho(A^2) = \rho^2(A), \quad (1.4.11)$$

hence (1.4.10) follows.

Definition 1.4.6

Let $A=(a_{i,j})$ be a positive definite $n \times n$ matrix and $x=(x_1, x_2, \ldots, x_n)$ be a $n$-component vector.

Then, the $A$-Norm of $x$ is defined as

$$||x||_A = (Ax, x)^{\frac{1}{2}}.$$

1.5 EIGENVALUES OF A MATRIX

Theorem 1.5.1

Let $A$ be an arbitrary $n \times n$ complex matrix, then all the eigenvalues of $A$ lie in the union of the disks

$$|z-a_{i,i}| \leq \sum_{\substack{j=1 \atop j \neq i}}^{n} |a_{i,j}|, \quad i \in [1,n].$$

(1.5.1)

The above theorem is due to Gerschgorin [30] and its proof may be found in [57],[35],[54].

Corollary 1

If $A=(a_{i,j})$ is an arbitrary $n \times n$ complex matrix and we have

$$v_1 = \max_{i \in [1,n]} \sum_{j=1}^{n} |a_{i,j}| \quad (1.5.2)$$

and

$$v_2 = \max_{j \in [1,n]} \sum_{i=1}^{n} |a_{i,j}| \quad (1.5.3)$$

then

$$\rho(A) \leq \min(v_1,v_2). \quad (1.5.4)$$

The condition (1.5.4) is a direct consequence of the fact that $A$ and $A^T$ have the same eigenvalues.

Theorem 1.5.2

Let $A=(a_{i,j})$ be an $n \times n$ strictly or irreducibly diagonally dominant complex matrix. Then the matrix $A$ is non-singular and if $a_{i,i}, i \in [1,n]$ are positive real numbers the eigenvalues $\lambda_i$ of $A$ satisfy

$$\Re\{\lambda_i\} > 0, \quad i \in [1,n]. \quad (1.5.5)$$

Proof

The proof can be easily obtained from Theorem 1.5.1 and can be found in [57](p.23).

Corollary 2

If $A=(a_{i,j})$ is a symmetric $n \times n$ strictly or irreducibly
diagonally dominant matrix with \( a_{i,i} > 0 \), i.e. [1, n], then \( A \) is positive definite.

**Proof**

Since a symmetric matrix has real eigenvalues the result follows from Theorems 1.3.1 and 1.5.2.

A useful lower bound for the smallest eigenvalue \( \lambda_1 \) of the matrix \( A = (a_{i,j}) \) can be obtained from the following result due to Collatz [11],

\[
\lambda_1 \geq \min_{i \in [1, n]} \left( |a_{i,i}| - \sum_{j=1 \atop j \neq i}^{n} |a_{i,j}| \right).
\]  

**1.6 BASIC DEFINITIONS**

**Definition 1.6.1**

Let \( x_i, i=1,2,\ldots,n \) be a set of vectors in a unitary space \( \mathcal{T} \). If \( A \) is a positive definite operator on \( \mathcal{T} \), then the set of vectors \( x_i \) is said to be "\( A \)-conjugate" or "\( A \)-orthogonal" if

\[
(Ax_i, x_j) = 0, \quad \text{where } i \neq j.
\]  

**Definition 1.6.2**

Consider the equation \( F(x) = C \), where \( C \) is a constant, which represents an ellipsoid in the \( n \)-dimensional space. Let \( x = (x_1, x_2, x_3, \ldots, x_n) \) be a point on the surface of the above ellipsoid. Then, assuming that the function \( F \) is continuous and differentiable, the vector with components \( \frac{\partial F}{\partial x_i}, i=1,2,\ldots,n \) is said to be the "Gradient" of \( F(x) \) (represented by \( \text{Grad } F \) or \( \nabla F \)).

At any point \( P \) the Gradient vector of \( F \) is normal to the surface \( F(x) = C \), which passes through \( P \).

For a scalar point function \( \phi(x,y,z) \) in rectangular coordinates it is
\[ \nabla \Phi = i \frac{\partial \Phi}{\partial x} + j \frac{\partial \Phi}{\partial y} + k \frac{\partial \Phi}{\partial z} \]  \hspace{1cm} (1.6.2)

where \( i, j, k \) are unit vectors along the \( X, Y, Z \) axes.

**Definition 1.6.3**

Let \( f_r(u_1, u_2, \ldots, u_n), r = 1, 2, \ldots, n \) be \( n \)-functions of the variables \( u_1, u_2, \ldots, u_n \) and let each of the partial derivatives \( \frac{\partial f_r}{\partial u_s}, r, s = 1, 2, \ldots, n \) be a continuous function of \( u_1, u_2, \ldots, u_n \).

Then, the Jacobian matrix \( f' \) of the functions \( f_1, f_2, \ldots, f_n \) with respect to \( u_1, u_2, \ldots, u_n \) is defined by:

\[
\begin{bmatrix}
\frac{\partial f_1}{\partial u_1} & \frac{\partial f_1}{\partial u_2} & \cdots & \frac{\partial f_1}{\partial u_n} \\
\frac{\partial f_2}{\partial u_1} & \frac{\partial f_2}{\partial u_2} & \cdots & \frac{\partial f_2}{\partial u_n} \\
\cdots & \cdots & \cdots & \cdots \\
\frac{\partial f_n}{\partial u_1} & \frac{\partial f_n}{\partial u_2} & \cdots & \frac{\partial f_n}{\partial u_n}
\end{bmatrix}
\]

\hspace{1cm} (1.6.3)

**Definition 1.6.4**

Let \( R \) denote an open region of \( n \)-dimensional space (\( n = 2 \) or 3) with a boundary \( C \) made up of one or more differentiable curves or surfaces.

Then, an operator \( L \) is said to be "self adjoint" if for any two sufficiently smooth functions \( \phi_1, \phi_2 \) the integral

\[
\iint_R [\phi_2 L(\phi_1) - \phi_1 L(\phi_2)] \, dx dy
\]

\hspace{1cm} (1.6.4)

is a function of the values of \( \phi_1, \phi_2 \) and their derivatives on \( C \) alone.
CHAPTER 2

ELLIPTIC AND PARABOLIC EQUATIONS IN TWO AND THREE SPACE DIMENSIONS - STANDARD METHODS OF SOLUTION
SECTION A: THE ELLIPTIC PROBLEM

2.1 INTRODUCTION

We will be concerned first with the application of the method of finite differences to obtain an approximate solution for the 'self-adjoint' two-dimensional problem defined by

\[
\frac{\partial}{\partial x}(C_1(x,y)\frac{\partial U}{\partial x}) + \frac{\partial}{\partial y}(C_2(x,y)\frac{\partial U}{\partial y}) + P(x,y)U(x,y) + Q(x,y) = 0, \quad (2.1.1)
\]

subject to the Dirichlet boundary conditions

\[
U(x,y) = \gamma(x,y), \quad (x,y) \in \mathcal{C}, \quad (2.1.1a)
\]

where the region \( R \) and its boundary \( \mathcal{C} \) are defined in section 1.1. The functions \( C_1, C_2, P, Q \) are assumed to be 'sufficiently smooth' functions (continuous and differentiable) and satisfy in \( R = R + \mathcal{C} \) the conditions

\[
C_1(x,y) > 0, \quad C_2(x,y) > 0, \quad P(x,y) < 0. \quad (2.1.1b)
\]

Consequently, from (1.1.3a) it follows that equation (2.1.1) is of elliptic type.

In a similar way as in section 1.2, a network of straight lines of spacings \( h_x \) and \( h_y \), parallel to each of the co-ordinate axes, is superimposed over the region \( R \). Then, using Taylor series, a discrete averaged centred finite difference approximation for the x-derivative of the form

\[
\frac{\partial}{\partial x}(C_1(x,y)\frac{\partial U}{\partial x}) \approx C_1(\frac{x}{2},y) \left[ \frac{u(x+h_x,y)-u(x,y)}{h_x^2} \right] - C_1(\frac{x-h_x}{2},y) \left[ \frac{u(x,y)-u(x-h_x,y)}{h_x^2} \right] \quad (2.1.2)
\]

with analogous expression for \( \frac{\partial}{\partial y}(C_2(x,y)\frac{\partial U}{\partial y}) \) can be obtained,
provided that the \((x,y)\) is a regular mesh point. In the case of an irregular mesh point, the Taylor series expansion of \(U(x,y)\) about such a point may still be used, giving approximations similar to (1.2.3)-(1.2.6) and (2.1.2). Substitution of the above finite difference approximations for the derivatives in (2.1.1), assuming that \(h_x = h_y = h\), and \(u_{i,j}\) denotes \(u(ih,jh)\), leads to the following linear five-point finite difference equation at the point \((ih,jh)\)

\[
d_{i,j}u_{i,j} - t_{i,j}u_{i,j} - b_{i,j}u_{i,j+1} - r_{i,j}u_{i,j+1} - \zeta_{i,j}u_{i-1,j} = s_{i,j} \tag{2.1.2a}
\]

where \(d_{i,j}, t_{i,j}, b_{i,j}, r_{i,j}, \zeta_{i,j} > 0\) and \(d_{i,j} \geq t_{i,j} + b_{i,j} + r_{i,j} + \zeta_{i,j}\) for all \(i,j\) with equality at interior point if \(P=0\).

Equation (2.1.2a) is generally represented by the 'molecule' (stencil or star) in Figure 2.1.

\[
\begin{align*}
t_{ij} & \quad ij+1 \\
Q_{ij} & \quad i-j \\
d_{ij} & \quad ij \\
r_{ij} & \quad i+1j \\
b_{ij} & \quad ij-1
\end{align*}
\]

FIGURE 2.1

which when applied at each regular mesh point, assuming that a fixed labelling is considered, yields a set of inhomogeneous, linear, simultaneous, symmetric difference equations, which can be expressed in matrix notation as

\[
Au = s \tag{2.1.3}
\]
where the vectors $u_i$ and $s$ consist of the unknown approximate solution $u_{i,j}$ and the known boundary value plus the quantity $h^2 Q(x,y)$ respectively.

The coefficient matrix $A$ is a sparse, real, quindiagonal, $(n \times n)$ matrix ($n$ is the number of unknown mesh points) with the following properties:

(i) Symmetric ,

(ii) Positive definite ,

(iii) Diagonally dominant ,

(iv) Irreducible ,

and

(v) It has positive diagonal and non-positive off-diagonal elements.

If all the interior mesh points in the region $R$ are regular, then $A$ will be symmetric and by Corollary 2 will be positive definite. Property (v) results from the fact that the diagonal entries of $A$ are given by:

$$[C_1(x+\frac{h}{2},y) + C_1(x-\frac{h}{2},y) + C_2(x,y+\frac{h}{2}) + C_2(x,y-\frac{h}{2}) - h^2 P(x,y)]$$

and the off-diagonal elements consist of the quantities:

$$-C_1(x+\frac{h}{2},y), -C_1(x-\frac{h}{2},y), -C_2(x,y+\frac{h}{2}), -C_2(x,y-\frac{h}{2}),$$

while properties (iii) and (iv) follow from Definition 1.3.3 and Theorem (p. 20) in [57] respectively.

Three Dimensional Case

We consider now a class of problems defined by the self-adjoint P.D.E. in three space dimensions, namely:

$$\frac{\partial}{\partial x}(C_1(x,y,z)\frac{\partial U}{\partial x}) + \frac{\partial}{\partial y}(C_2(x,y,z)\frac{\partial U}{\partial y}) + \frac{\partial}{\partial z}(C_3(x,y,z)\frac{\partial U}{\partial z}) +$$

$$+ P(x,y,z)U(x,y,z) + Q(x,y,z) = 0,$$

$(x,y,z) \in \mathbb{R}$ \hspace{1cm} (2.1.5)
where $C_1, C_2, C_3$ are strictly positive functions, $P$ non-positive and $U(x,y,z) \in \mathbb{R}$, where $R$ is the interior of a compact region subject to the general boundary conditions

$$\alpha U + \beta \frac{\partial U}{\partial \gamma} = \gamma \quad (2.1.6)$$

for $U \in \mathcal{C}$, the exterior boundary of $R$.

On $\mathcal{C}$, $\alpha$ and $\beta$ are positive, piecewise continuous, and $\gamma$ denotes the direction of the outward drawn normal. The coefficients of (2.1.5) are assumed to be 'sufficiently smooth' functions.

The region under consideration $R$ is covered by a volumetric grid system $R_h$, with spacings $h_x, h_y, h_z$ defined by:

$$R_h = \{(ih_x, jh_y, kh_z): 0 \leq i, j, k \leq M\}. \quad (2.1.7)$$

Assuming that $(i,j,k)$ denotes the grid point $(ih_x, jh_y, kh_z)$ and $U_{i,j,k}$ denotes $U(ih_x, jh_y, kh_z)$, a discrete approximation for $x$-derivative of the form:

$$\frac{\partial}{\partial x}(C_1(x,y,z)\frac{\partial U}{\partial x}) = C_1(x+\frac{x}{2},y,z) \left[ \frac{u(x+h_x,y,z)-u(x,y,z)}{h_x^2} \right] - C_1(x-\frac{x}{2},y,z) \left[ \frac{u(x,y,z)-u(x-h_x,y,z)}{h_x^2} \right] \quad (2.1.8)$$

with an analogous expression for $\frac{\partial}{\partial y}(C_2(x,y,z)\frac{\partial U}{\partial y})$ and $\frac{\partial}{\partial z}(C_3(x,y,z)\frac{\partial U}{\partial z})$ are used to derive a linear finite difference equation at point $(ih_x, jh_y, kh_z)$.

When the seven-point, three-dimensional molecule (Figure 2.2) is used...
the following finite difference equation is obtained

\[
D_{i,j,k} u_{i,j,k} - I_{i,j,k} u_{i,j,k} - L_{i,j,k} u_{i,j,k} + O_{i,j,k} u_{i,j,k} + T_{i,j,k} u_{i,j,k} + B_{i,j,k} u_{i,j,k} - R_{i,j,k} u_{i,j,k} = S_{i,j,k} \quad (2.1.9)
\]

with \( D_{i,j,k} > 0 \), \( I_{i,j,k} \), \( O_{i,j,k} \), \( T_{i,j,k} \), \( B_{i,j,k} \), \( R_{i,j,k} \), \( L_{i,j,k} \) (representing mnemonic abbreviations for point Inplane, Outplane, Top, Bottom, Right and Left of the point \((i h_x, j h_y, k h_z)\)) and

\[
D_{i,j,k} u_{i,j,k} + I_{i,j,k} u_{i,j,k} + L_{i,j,k} u_{i,j,k} + O_{i,j,k} u_{i,j,k} + T_{i,j,k} u_{i,j,k} + B_{i,j,k} u_{i,j,k} + R_{i,j,k} u_{i,j,k} + L_{i,j,k} u_{i,j,k} = S_{i,j,k} \quad (2.1.10)
\]

for all \( i,j,k \), with equality at interior point if \( P=0 \).

Under the same considerations as in the 2D-case, the obtained set of symmetric difference equations may be expressed as the system (2.1.3), the coefficient matrix \( A \) being a sparse, real, seven-diagonal, square matrix with the properties (2.1.4).

Henceforth, throughout the thesis, the solution of the system \( Au = S \) will be restricted to matrices \( A \) with the properties (2.1.4b)-(2.1.4e) unless otherwise specified.
2.2 THE MODEL PROBLEMS

Let us consider the following 2D-model problem:

We seek to determine a continuous, twice differentiable function $u(x,y)$ in the region,

$$R = (0,1) \times (0,1)$$

with boundary $C$, satisfying Laplace's equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0, \quad (x,y) \in R,$$

subject to the Dirichlet boundary conditions

$$U(x,y) = 0, \quad (x,y) \in C.$$  \hspace{1cm} (2.2.1)

The region $R$ under consideration is covered by a rectilinear net with mesh spacing $h$ in the $X,Y$ directions and mesh points $(x_i, y_j)$, where $x_i = ih$, $i = 0, 1, \ldots, M$; $y_j = jh$, $j = 0, 1, \ldots, M$.

This problem is a special case $(C_1(x,y) = C_2(x,y) - 1, P(x,y) = Q(x,y) = 0)$ of the general 2D-problem given in section 2.1.

Substituting the finite difference approximations for the derivatives in (2.2.1) the following five-point formula is obtained

$$4u_{i,j} - u_{i+1,j} - u_{i-1,j} - u_{i,j+1} - u_{i,j-1} = 0.$$  \hspace{1cm} (2.2.2)

If we order the $(M-1)^2$ internal mesh points column-wise (Figure 2.3)
the coefficient matrix $A$ of the obtained system (2.1.3) is a real, square, quindiagonal, sparse matrix of order $m^2 = (M-1)^2$ and of the general form,

$$A = \begin{bmatrix}
A_1 & -I & & \\
-I & A_2 & -I & 0 \\
& -I & \ddots & -I \\
& & \ddots & \ddots & -I \\
& & & -I & A_m
\end{bmatrix} \quad (2.2.4a)$$

where $I_m$ is the unit matrix of order $m$ and $A_i$, $i \in [1,m]$ matrices of order $m$, given by,

$$A_i = \begin{bmatrix}
4 & -1 \\
-1 & 4 & -1 & 0 \\
& -1 & 4 & -1 \\
& & \ddots & \ddots & -1 \\
& & & 0 & -1 \\
& & & & 4
\end{bmatrix}, \quad i \in [1,m]. \quad (2.2.4b)$$

The solution vector $\mathbf{u}$ and the right hand side vector $\mathbf{s}$ of (2.1.3) are $(m^2 \times 1)$ column vectors defined as before in section 2.1.

3D-model Problem

We consider now the Laplace's equation in three space dimensions:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} = 0, \quad (x,y,z) \in \Omega, \quad (2.2.5)$$

subject to the Dirichlet boundary conditions

$$U(x,y,z) = 0, \quad (x,y,z) \in \Gamma \quad (2.2.5a)$$

where $\Omega = (0,1) \times (0,1) \times (0,1)$ and $\Gamma$ its boundary.

The region under consideration $\Omega$ (unit cube) is covered by an equally spaced three-dimensional grid, defined by,
where \( M.h=1 \) and \( h \) the grid spacing.

Discrete approximations to the derivatives in (2.2.5), in an analogous way to (2.1.8), leads to the following seven-point finite difference equation

\[
6u_{i,j,k} - u_{i+1,j,k} - u_{i-1,j,k} - u_{i,j+1,k} - u_{i,j-1,k} - u_{i,j,k+1} - u_{i,j,k-1} = 0.
\]

(2.2.6)

Ordering the \((M-1)^3\) internal points of \( R_h \) with increasing values of \( j \), then \( i \), then \( k \) (Figure 2.4).

**FIGURE 2.4**

the coefficient matrix \( A \) of the obtained system (2.1.3) is a real, square, seven-diagonal, sparse matrix of order \( m^3 = (M-1)^3 \) and of the general form:
where I is the unit matrix of order $m^2$ and $A_i$, $i \in [1, m]$ are matrices of order $m^2$ given by:

$$A_i = \begin{bmatrix}
-1 & -I \\
-1 & A_{i-1} & -I \\
& -1 & A_{i-2} & -I & 0 \\
& & -1 & A_{i-3} & -I & 0 \\
& & & -1 & A_{i-4} & -I & 0 \\
& & & & -1 & A_{i-5} & -I & 0 \\
& & & & & & & \vdots
\end{bmatrix}, j=(i-1)m, \ i \in [1, m]$$  \hspace{1cm} (2.2.7a)

where $J$ is now the unit matrix of order $m$ and $B_i$, $i \in [1, m]$ are matrices of order $m$ given by

$$B_i = \begin{bmatrix}
0 & -J \\
-1 & B_{i-1} & -J \\
& -1 & B_{i-2} & -J & 0 \\
& & -1 & B_{i-3} & -J & 0 \\
& & & -1 & B_{i-4} & -J & 0 \\
& & & & -1 & B_{i-5} & -J & 0 \\
& & & & & & & \vdots
\end{bmatrix}, \ i \in [1, m].$$  \hspace{1cm} (2.2.7b)

The solution vector $u$ and the right hand side vector $s$ of (2.1.3), in this case, are $(m^3 \times 1)$ column vectors.

2.3 SOME POINT ITERATIVE METHODS

For the solution of the linear system (2.1.3), where the coefficient matrix $A$ is a large, sparse matrix with no more than five or seven non-zero elements in any row, e.g. formulae (2.2.3) or (2.2.6) respectively, iterative methods of successive approximations
are usually preferable to direct methods such as Gaussian elimination and Cramer's rule.

The iterative process, starts with an arbitrary initial approximation (can be chosen arbitrarily close to the exact solution of the difference equations) to the solution and then it is successively modified according to some predetermined rule.

Consider the system (2.1.3) and let \( u^{(n)} \) be a sequence, such that,

\[
\frac{u^{(n)}}{u^{(n)}} \rightarrow A^{-1}s \quad \text{as } n \to \infty .
\]

Generally an iterative scheme is said to be of degree \( v \) if \( u^{(n)} \)

is a function of \( A, s, u^{(n-1)}, u^{(n-2)}, \ldots, u^{(n-v)} \).

Since computer storage requirements must be minimised the value of \( v \) must be kept reasonably small. Consequently, for an iteration of first degree (\( v=1 \)), we obtain

\[
u^{(n)} = f_n(A, s, u^{(n-1)}) .
\]

The iterative process (2.3.1) is said to be

(i) 'linear' if \( f_n \) is a linear function of \( u^{(n-1)} \) and

(ii) 'stationary' if \( f_n \) is independent of \( n \).

A general iterative scheme of the form

\[
u^{(n+1)} = \theta^{(n)} + f_n(u^{(n)})
\]

is said to be 'non-stationary' if either \( \theta \) or \( f_n \) (or both) is (are) dependent on \( n \).

Let \( A=E-F \), so the system (2.1.3) can be written as

\[
Eu = Fu + s .
\]

Assuming that \( u^{(n)} \) denotes the \( n \text{th} \) approximation to the solution \( u \), then the following iterative scheme can be obtained

\[
Eu^{(n+1)} = Fu^{(n)} + s
\]

where the initial approximation (starting vector) \( u^{(0)} \) is an
arbitrary column vector.

The matrix $E$ (assuming that $\det(E) \neq 0$ and $E^{-1}$ exists) is chosen so that $u^{(n+1)}$ can be easily obtained from (2.3.3), provided that $u^{(n)}$ is known, and is the structure of $E$, which determines two important classes of iterative methods, specifically:

(i) if $E$ is a diagonal matrix, then the methods of 'Simultaneous displacements' [e.g. Jacobi, Richardson, etc.] are developed.

(ii) if $E$ is a lower triangular matrix the methods of 'Successive displacements' [e.g. Gauss-Seidel, Successive Over-relaxation (S.O.R.), etc.] arise.

Note that although in the 'Successive displacement' method the ordering of the mesh points is such that the latest estimate $u_i^{(n+1)}$ of the components of $u$ is used in order to determine a new estimate of a component of $u^{(n+1)}$, in the 'Simultaneous displacement' method the order in which the components $u_i^{(n+1)}$ are obtained is of no consequence.

Since in this thesis we are solely interested in 'Simultaneous displacement' methods, in the following we define the basic methods of this important class of Iterative Procedures.

Class of 'Simultaneous Displacement' Methods

Let us consider without loss of generality that the coefficient matrix $A$ of the system (2.1.3) can be written as

$$A = D - L - U$$

where $D$ is a positive diagonal matrix whose elements are the diagonal elements of $A$ and $L, U$ are respectively lower and upper triangular matrices with zero diagonal elements.
Since $D$ is positive diagonal matrix and so $D^{-1}$ exists, let

$$ B = D^{-1}(L+U) \quad \text{and} \quad c = D^{-1}s. \quad (2.3.5) $$

Combining (2.1.3), (2.3.4) and (2.3.5) we obtain

$$ u = B.u + c. \quad (2.3.6) $$

Then, the iterative scheme defined by

$$ u^{(n+1)} = B.u^{(n)} + c, \quad n \geq 0, \quad (2.3.7) $$

is known as 'Jacobi' iteration and can be equivalently written as

$$ u^{(n+1)} = u^{(n)} + (c - D^{-1}Au^{(n)}). \quad (2.3.8) $$

The method of 'Simultaneous displacement' is defined by

$$ u^{(n+1)} = u^{(n)} + a(c - D^{-1}Au^{(n)}) \quad (2.3.9) $$

where $a$ is an acceleration parameter (positive constant) chosen to speed up convergence.

Assuming, without loss of generality, that $D$ may be chosen as the identity matrix $I$, the Jacobi (2.3.7) and Simultaneous Displacement (2.3.9) iterations can be re-written respectively as,

$$ u^{(n+1)} = (L+U)u^{(n)} + s, \quad (2.3.10) $$

and

$$ u^{(n+1)} = u^{(n)} + a(s - Au^{(n)}), \quad (2.3.11) $$

or equivalently

$$ u^{(n+1)} = u^{(n)} + r^{(n)}, \quad (2.3.12) $$

and

$$ u^{(n+1)} = u^{(n)} + a.r^{(n)} = (I-aA)u^{(n)} + a.s \quad (2.3.13) $$

where $r$ is the residual vector defined by

$$ r^{(n)} = s - Au^{(n)}. \quad (2.3.14) $$

Convergence of Simultaneous Displacement Methods

The iterative methods described previously, can be written in the general form,

$$ u^{(n+1)} = \tilde{M}u^{(n)} + d, \quad n \geq 0 \quad (2.3.15) $$

where $\tilde{M}$ is known as the 'iteration matrix' of the considered method and $d$ is a column vector of constants.
We consider now the system (2.1.3) and the iterative scheme (2.3.15).

**Definition 2.3.1**

The 'consistency condition' for an iterative process is said to be satisfied if the solution $u$ of system (2.1.3) is substituted for $u^{(n)}$ in (2.3.15) then $u^{(n+1)}$ is also the solution $u$ of the system, i.e. the iterative procedure makes no further modification of successive iterates once the solution has been obtained.

**Definition 2.3.2**

An iterative method is said to be 'convergent', if for all initial vectors $u^{(0)}$, each component of the successive iterates $u^{(n)}$ tends to the corresponding component of the solution $u$ of $Au = s$, for any given $s$.

**Theorem 2.3.1**

An iterative method which can be expressed in the form of equation (2.3.15) converges if and only if (iff) $\rho(\tilde{M}) < 1$.

**Proof**

Let $e^{(n)} = u^{(n)} - u$ be the error vector after $n$-iterative steps. We assume that the iterative method is consistent, i.e.

$$u = \tilde{M} u + d$$  \hspace{1cm} (2.3.16)

then from (2.3.15), (2.3.16) we obtain

$$e^{(n+1)} = \tilde{M} e^{(n)}$$  \hspace{1cm} (2.3.17)

and furthermore,

$$e^{(n)} = \tilde{M} e^{(0)}$$  \hspace{1cm} (2.3.18)

where $e^{(0)}$ is the error vector associated with the initial vector $u^{(0)}$. Assuming that $e^{(0)}$ is bounded, i.e. $|e^{(0)}_i| \leq \varepsilon$, $i \in [1, N]$ for some constant $\varepsilon$, then from (2.3.18) it follows that $e^{(n)} \to 0$, as $n \to \infty$ if and only if $\tilde{M} \to \Box$ (where $\Box$ denotes the null matrix) as $n \to \infty$. 
By a theorem in [57] (p.82) (stating: if \( A \) is an arbitrary \( n \times n \) matrix the \( A \) is convergent iff \( \rho(A)<1 \) this will be true if and only if \( \rho(M)<1 \).

Note that if the coefficient matrix \( A \) has the properties (2.1.4) it can be shown [63] (pp.13-14) that the Jacobi method converges.

**Rate of Convergence of Simultaneous Displacement Methods**

The effectiveness of an iterative method is generally evaluated both from the computational work required per iteration and from the number of iterations required for convergence.

We usually say for practical purposes that an iterative method has converged when

\[
||e^{(n)}|| = \varepsilon ||e^{(0)}||,
\]

where \( e \) is the error vector, \( \varepsilon \) is a predetermined positive factor and \( || \ || \) denotes the \( L_2 \) or spectral norm.

From (2.3.18) we obtain

\[
||e^{(n)}|| = ||\hat{M}^n e^{(0)}|| \cdot ||e^{(0)}||.
\]

(Note that in the case of a non-stationary iteration the relationships (2.3.18), (2.3.20) become

\[
||e^{(n)}|| = \hat{M}_n \hat{M}_{n-1} \ldots \hat{M}_1 e^{(0)},
\]

and

\[
||e^{(n)}|| = ||\sum_{i=0}^{n} \hat{M}_i e^{(0)}|| \cdot ||e^{(0)}||,
\]

respectively).

It is known that \( ||\hat{M}^n|| \to 0 \) as \( n \to \infty \) iff \( \rho(M)<1 \), so (2.3.19) can be satisfied choosing \( n \) large enough such that

\[
||\hat{M}^n|| \leq \varepsilon.
\]

Then, for all \( n \) sufficiently large that \( ||\hat{M}^n||<1 \), (2.3.21) is equivalent to

\[
n \geq \frac{k \eta \varepsilon}{\rho(M)^n}.
\]
**Definition 2.3.3**

Consider the iterative method (2.3.15). Then the 'average rate of convergence' is defined by

\[ R_n(M) = \frac{1}{n} \log |M^n| \]  

(2.3.23)

**Definition 2.3.4**

The 'asymptotic rate of convergence' is defined by

\[ R(M) = \lim_{n \to \infty} R_n(M) = -\log [\rho(M)] \]  

(2.3.24)

Henceforth, we shall refer to the \( R(M) \) simple as the 'rate of convergence'.

It can be shown [61] (p.87), that \( \rho(M) = \lim_{n \to \infty} (|M^n|)^{1/n} \), so equality (2.3.24) is a direct result of (2.3.23).

Note that from Definitions 2.3.2-2.3.4 we obtain that

(i) the number of iterations needed to reduce an initial error by a predetermined factor is approximately inversely proportional to the 'average rate of convergence'.

(ii) the rapidity of convergence depends on the rapidity with which the norm \( |M^n| \) tends to zero, as \( n \to \infty \).

(iii) the smaller the value of \( \rho(M) \), the greater the rate of convergence \( R(M) \). Consequently, the iterative processes achieve their fastest rate of convergence when \( \rho(M) \) is reduced to a minimum.

Consider now the simultaneous iteration defined by (2.3.15).

It can be shown that the error vector satisfies:

\[ \underline{e}_{(n+1)} = (I-aA)\underline{e}_{(n)} \]  

(2.3.25)

or

\[ \underline{e}_{(n+1)} = (I-aA)^{(n+1)}\underline{e}_{(0)} \]  

(2.3.26)
Note that the error operator is constant throughout the iteration (stationary, linear iterative procedure).

Let us assume that the matrix \((I-aA)\) has \(N\)-linearly independent eigenvectors \(v_i\), associated with \(N\)-distinct eigenvalues \(\lambda_i\) and let \(\mu_i, i \in [1,N]\) be the eigenvalues of \(A\). Given that an arbitrary vector \(e^{(0)}\) can be expressed as
\[
e^{(0)} = \sum_{i=1}^{N} c_i v_i,  \tag{2.3.27}
\]
where \(c_i, i \in [1,N]\) constants, we have that
\[
e^{(n+1)} = \sum_{i=1}^{N} c_i \lambda_i^{(n+1)} v_i = \lambda_1^{(n+1)} v_1 + \lambda_2^{(n+1)} v_2 + \ldots.  \tag{2.3.28}
\]
Hence, the iterative scheme converges if \(p(I-aA)<1\) i.e., the modulus of the largest eigenvalue of \((I-aA)\) must be less than unity.

Let \(m, M\) be the extreme eigenvalues of \(A\) e.g.,
\[
0 \leq m \leq \mu_i \leq M < +\infty, i \in [1,N]. \tag{2.3.29}
\]
Since
\[\lambda_i = 1-a\mu_i, i \in [1,N]\]
the necessary and sufficient condition for convergence gives
\[|1-a\mu_i|<1, i \in [1,N], \tag{2.3.30}\]
hence,
\[0<a<\frac{2}{M}. \tag{2.3.31}\]

The choice of the parameter \(a\) in such a way that \(p(I-aA)\) is minimised, leads to the fastest rate of convergence. The optimal \(a\) is obtained for the smallest of \(\max\{|1-am|, |1-aM|\}\) and this value of \(a\) is given by:
\[|1-am| = |1-aM|\]
or equivalently
\[a = \frac{2}{m+M}. \tag{2.3.32}\]
With this choice of a, for all i, we obtain

$$|1-a_{M-i}| \leq \frac{M-m}{M+m} < 1.$$  \hspace{1cm} (2.3.33)

From (2.3.24) the (asymptotic) rate of convergence $R_s$ is given by

$$R_s = -\ln(\rho(I-aA))$$  \hspace{1cm} (2.3.34)

so that from (2.3.33) we have

$$R_s \geq \frac{2}{Mm}, \text{ for } \frac{M}{m} \gg 1$$  \hspace{1cm} (2.3.35)

where the ratio $\frac{M}{m}$ is known as the P-condition number of A.

Note that the rate of convergence for the Simultaneous displacement method is independent of the ordering of the points.

Consider now the iterative scheme (2.3.15).

If a sequence of different factors $a_n$ (instead of the constant factors a) for each iteration is multiplied by each component of the residual vector $r^{(n)}$ and then added to each component of the $u^{(n)}$ to give the next value of $u^{(n+1)}$, the following non-stationary iterative scheme, known as 'Richardson's Method' [51], is obtained:

$$u^{(n+1)} = u^{(n)} + a_n r^{(n)} = (I-a_nA)u^{(n)} + a_n s.$$  \hspace{1cm} (2.3.36)

The constants $a_n$ are either given in terms of the extreme eigenvalues of the matrix A or calculated during the iteration from formulae involving $u^{(n)}$. The error vector for the iterative scheme (2.3.36) is given by:

$$e^{(n+1)} = (I-a_n A)e^{(n)}$$  \hspace{1cm} (2.3.37)

or

$$e^{(n+1)} = \prod_{i=0}^{n} (I-a_i A)e^{(0)}.$$  \hspace{1cm} (2.3.38)

Note that the error operator changes for each iteration (Non-stationary iterative method).

Let

$$F_{n+1}(A) = \prod_{i=0}^{n} (1-a_i A).$$  \hspace{1cm} (2.3.39)
If \( u_i, \ i \in [1,N] \) are the \( N \)-eigenvalues of the positive definite matrix \( A \) and \( v_i \) are the corresponding eigenvectors, then the eigenvalues and eigenvectors of \( F_{n+1}(A) \) are \( F_{n+1}(u_i) \) and \( v_i \) respectively.

A combination of (2.3.38), (2.3.27) yields

\[
e^{(n+1)} = F_{n+1}(A)e^{(0)} = \sum_{i=1}^{N} F_{n+1}(A)c_i v_i
\]

or

\[
e^{(n+1)} = \sum_{i=1}^{N} c_i v_i F_{n+1}(u_i) . \tag{2.3.40}
\]

The minimization of the error vector is now equivalent to the following problem:

Find a polynomial \( F_{n+1}(x) \) of degree \( (n+1) \) such that

\[
\max_{x \in [m,M]} |F_{n+1}(x)| , \tag{2.3.41}
\]

is minimized, under the constraint

\[
F_{n+1}(0) = 1 . \tag{2.3.42}
\]

The polynomial satisfying (2.3.41), (2.3.42) is given in [41] to be:

\[
F_{n+1}(x) = \frac{T_{n+1}^{(M+m-2x)}}{T_{n+1}^{(M-m)}} , \tag{2.3.43}
\]

where \( T_n \) is the Chebyshev polynomial of degree \( n \) given by:

\[
T_n(x) = \cos(n \cos^{-1}x) = \frac{1}{2}[[x+\sqrt{x^2-1}]^n+[x-\sqrt{x^2-1}]^{-n}] \tag{2.3.43a}
\]

adjusted in the interval \([-1,1]\). The optimal choice of parameters \( a_i \) is such that the zeros of \( F_{n+1}(x) \) coincide with the set \( \{a_i\} \), provided that the extreme eigenvalues \( m,M \) and the value of the ratio \( \frac{||r^{(0)}||}{||r^{(n)}||} \), at which the iteration is terminated, are given.

From (2.3.39) the zeros of \( F_{n+1}(x) \) are:

\[
x_i = [a_i^{(n+1)}]^{-1} , \ i \in [1,n+1] , \tag{2.3.43b}
\]
while, from (2.3.43), the zeros of $T_{n+1}(\frac{M+m-2x}{M-m})$ and consequently of $F_{n+1}(x)$ are:

$$\frac{M+m-2x}{M-m} = \cos \left\{ \frac{(2i-1)\pi}{2(n+1)} \right\}, \quad i \in \{1,n+1\}. \quad (2.3.43c)$$

By equating the zeros of polynomials we obtain

$$a_{i}^{(n+1)} = \frac{2}{(M+m)-(M-m)\cos[\frac{(2i-1)\pi}{2(n+1)}]}, \quad (2.3.44)$$

and assuming that only $v$-iterations are to be performed then,

$$a_{i}^{(v)} = \frac{2}{(M+m)-(M-m)\cos[\frac{(2i-1)\pi}{2v}]}, \quad i \in \{1,v\}. \quad (2.3.45)$$

Since the maximum absolute value of the numerator of (2.3.43) is unity, then the maximum value of $F_{n+1}(x)$, as $n \to \infty$, is given by:

$$\max_{x \in [m,M]} |F_{n+1}(x)| = \left[ T_{n+1}(\frac{M+m}{M-m}) \right]^{-1}. \quad (2.3.46)$$

Consider now the iterative scheme (2.3.36).

**Definition 2.3.5**

The 'average convergence factor' for the first $v$-iterative steps is defined by the quantity

$$\left\| \frac{e^{(v)}}{e^{(0)}} \right\|^{1/v}.$$  

**Definition 2.3.6**

The 'average rate of convergence' for the first $v$-iterative steps is defined by the quantity

$$-\frac{1}{v} \ln \left[ \frac{\left\| e^{(v)} \right\|}{\left\| e^{(0)} \right\|} \right]^{1/v}.$$  

It can be shown [27] that the average rate of convergence is bounded by:

$$-\frac{1}{v} \ln 2 + \ln (z_{0} + \sqrt{2z_{0}^{2} - 1}), \quad (2.3.47)$$

where $z_{0} = \frac{M+m}{M-m} > 1$. 
The rate of convergence, is the asymptotic value of (2.3.47) as \( \nu \to \infty \), and is given by
\[
R_R = \frac{\omega_1}{z_0 + \sqrt{z_0^2 - 1}}.
\] (2.3.48)

Given that for most problems \( z_0 = 1 + \frac{2}{m} \), then we obtain
\[
R_R = \frac{2}{\sqrt{m}} \quad \text{for } m > 1.
\] (2.3.49)

Assuming that the extreme eigenvalues \( m, M \), the value of \( \nu \) and the value of the ratio \( \frac{||r^{(0)}||}{||r^{(\nu)}||} \) (at which the iteration is to be terminated) are given, the procedure is then simplified, to calculate the terms of the sequence \( \{a_i\} \), i.e. \( \{1, \nu\} \), which is given by (2.3.44) and is the best possible choice if only \( \nu \)-iterations are to be performed.

Consider now the following linear stationary iteration of second degree, known as 'second order Richardson's iterative scheme'
\[
\mathbf{u}^{(n+1)} = \mathbf{u}^{(n)} + a(s - A\mathbf{u}^{(n)}) + \beta(\mathbf{u}^{(n)} - \mathbf{u}^{(n-1)})
\] (2.3.50)

where parameters \( a, \beta \) remain constant throughout the iteration and are chosen to provide maximum convergence to the solution. The error vector for (2.3.50) satisfies
\[
\mathbf{e}^{(n+1)} = [(1 + \beta)I - aA]\mathbf{e}^{(n)} - \beta\mathbf{e}^{(n-1)}
\] (2.3.51)

thus for each error mode we have
\[
\mathbf{e}_i^{(n+1)} = \left[(1 + \beta - a\lambda_i)\lambda_i - \beta\lambda_i\right] \mathbf{e}_i^{(n)} - \beta\mathbf{e}_i^{(n-1)}.
\] (2.3.52)

Let \( \lambda_i \) be the eigenvalues of the matrix associated with the iterative process then
\[
\mathbf{e}_i^{(n+1)} = \lambda_i\mathbf{e}_i^{(n)} = \lambda_i^2\mathbf{e}_i^{(n-1)}.
\] (2.3.53)

From (2.3.52), (2.3.53) we obtain
\[
\mathbf{e}_i^{(n+1)} = \left[(1 + \beta - a\lambda_i)\lambda_i - \beta\lambda_i\right] \mathbf{e}_i^{(n)}
\] (2.3.54)

and combination of (2.3.53), (2.3.54) leads to
\[
\gamma_i^2 - (1 + \beta - a\lambda_i)\gamma_i + \beta = 0,
\] (2.3.55)
which yields

\[ Y_i = \frac{\delta_1 \pm \sqrt{\delta_1^2 - 4\beta}}{2} \]  

(2.3.56)

where \( \delta_1 = 1 + \beta - \alpha u_1 \).

Choosing the parameters \( \alpha, \beta \) so that \( (\delta_1^2 - 4\beta) < 0 \) for all \( u_1 \) (which means that \( Y_i \) will be complex) all \( |Y_i| \) will be identical. Then, the choices

\[ 1 + \beta - \alpha m = 2\sqrt{\beta} \]  

(2.3.57a)

\[ 1 + \beta - \alpha M = -2\sqrt{\beta} \]  

(2.3.57b)

lead to

\[ a = \left( \frac{2}{\sqrt{m} + \sqrt{M}} \right)^2, \quad \beta = \frac{\sqrt{M} - \sqrt{m}}{\sqrt{M} + \sqrt{m}} \]  

(2.3.58)

The above choice of parameters, which makes the square root of (2.3.56) zero or negative, yields,

\[ |Y_i| = \sqrt{\beta} \]  

(2.3.59)

The eigenvalues \( Y_i \) are complex but all have the same absolute value and thus all the error modes are decreased at the same rate. Note that \( |Y_i| \) is always smaller than the maximum of \( |\lambda_d|, |\lambda_d| \) and consequently the procedure is always more rapidly convergent than the Simultaneous Iteration. In the case that the estimates of the extreme eigenvalues \( m, M \) are not exact, then it is advisable to underestimate \( m \) and overestimate \( M \) assuring the \( Y_i \) being complex and therefore all the modes decaying at the same rate. From (2.3.50) it can be seen that to commence the iterative process two iterates are required and although one initial iterate is available, only \( |Y_i| \) is known and the second initial iterate is unobtainable. Thus the convergence rate indicated by (2.3.59) is asymptotically only approached as \( i \to +\infty \). Given that the spectral radius of the iteration is \( \sqrt{\beta} \), then the rate of convergence is \( \frac{2}{\sqrt{m}} \).
Note that for each iteration of the method at least two vectors need to be stored.

Another non-stationary second degree iteration, less sensitive to round-off errors, is the 'Chebychev second order method' defined by

\[ u(n+1) = u(n) + a_n (s - Au(n)) + \beta_n (u(n) - u(n-1)) \]  (2.3.60)

where \( a_n, \beta_n \) are varied with each iterative step.

If the parameters \( a_n, \beta_n \) have to be computed during the iteration, the technique proves to be inefficient especially for large values of \( N \). For pre-determined values of \( a_n, \beta_n \) the computational work per iteration is not much more than that of the iteration for the simultaneous displacement method and the storage requirements are no greater. The values of parameters \( a_n, \beta_n \) are given in [55] as follows:

\[
a_n = \frac{4T_n(M+m)}{(M-m)T_{n+1}(\frac{M+m}{M-m})}, \quad \beta_n = \frac{T_{n+1}(M+m)}{T_n(\frac{M+m}{M-m})}. \]  (2.3.61)

The error vector is given by

\[
e(n+1) = \frac{T_{n+1}(\frac{M+m-2x}{M-m})}{T_{n+1}(\frac{M+m}{M-m})} e(0), \quad (2.3.62)
\]

which leads to identical results to those obtained using Richardson's method for rate of convergence.

Since \( a_n, \beta_n \) are less than unity round off-errors do not appear and the method is preferred to Richardson's method with the only inconvenient matter raised being the estimation of the extreme eigenvalues \( m, M \).

Finally, for the 2D-model problem it can easily be shown [27] (p.226) that the simultaneous displacement method of (2.3.13) is equivalent to Jacobi method of (2.3.12).
The processes defined so far, belong to the class of point iterative methods, in which each component of \( u^{(n)} \) is expressed 'explicitly' i.e., can be expressed by itself using already computed approximate values of the other unknowns. Grouping the equations of the original system, according to a predetermined rule\(^{+}\), subsets of the components can be solved at once, giving rise to the so-called 'Block iterative' methods.

With proper formulation, all the discussed point iterative methods give their counterpart block iterative methods. Since, in the following chapters, we are interested in Point iterative methods, the discussion for the Block methods is terminated here.

2.4 THE CONJUGATE GRADIENT METHOD

Consider the system,

\[
A \cdot x = s
\]  

(2.4.1)

assuming that the coefficient matrix \( A \) is real, symmetric and positive definite. Let \( x_1 \) be an arbitrary vector and \( r_1 \) the corresponding residual, as defined in section 2.3.

The system (2.4.1) can be written as the gradient of a quadratic function

\[
F[x] = \frac{1}{2} x \cdot Ax - s \cdot x
\]  

(2.4.2)

The solution of system (2.4.1) is equivalent to the minimization of function \( F[x] \), which defines an ellipsoid in the \( N \)-dimensional space of the elements of \( x \). For an arbitrary trial vector \( x_1 \), the residual \( r_1 \) is given by:

\[
r_1 = s - Ax_1 = -\text{Grad } F[x_1]
\]  

(2.4.3)

\(^{+}\)For an elliptic P.D.E. a logical block to be advanced simultaneously can consist of all points on a row or column of a rectangular mesh.
and it is normal to the surface of the ellipsoid defined by (2.4.2) (see Definition 1.6.2).

Consider the iterative scheme:

$$x_{i+1} = x_i + a_i m_i$$  \hspace{1cm} (2.4.4)

where $a_i$ is an arbitrary constant dependent upon $i$ and $m_i$ is an arbitrary direction.

The problem now is to choose an $a_i$ such that the quadratic function $F[x_{i+1}]$ will be a minimum for a given direction $m_i$.

Consider now a plane defined by $m_i$ and $r_i$. Then, the intersection of this plane and the surface $F[x]=\text{constant}$ is represented by Figure 2.5.

Note that the tip of the vector $x_i$ appears as a point in this plane. The minimum $F[x_{i+1}]$ is tangent to the direction $m_i$ and so the residual $r_{i+1}$ is also normal to $m_i$. 

\hspace{1cm} FIGURE 2.5
From (2.4.3) we can easily obtain
\[ r_{i+1} - r_i = s - Ax_{i+1} - [s - Ax_i] = A[x_{i+1} - x_i] \]  
(2.4.5)
and combining (2.4.4), (2.4.5) we have
\[ r_{i+1} = r_i - a_i A m_i \]  
(2.4.6)
Then, using the orthogonality relation \( r_{i+1} \cdot m_i = 0 \), equation (2.4.6) becomes:
\[ r_{i+1} \cdot m_i = r_i \cdot m_i - a_i A m_i \cdot m_i = 0 \]  
(2.4.7)
which leads to
\[ a_i = \frac{r_i \cdot m_i}{m_i \cdot A m_i} \]  
(2.4.8)
The above choice of \( a_i \) systematically reduces \( F[x_{i+1}] \) and the method converges for any given \( m_i \).

If the direction \( m_i \) is chosen as vector \( a_i \), directed from a point on the ellipse towards the center of the ellipses, then the 'conjugate directions' are defined by \( \sigma_i \) and a vector \( t_i \), tangent to the ellipse at \( x_i \). The vectors \( \sigma_i \) and \( t_i \) are orthogonal with respect to the matrix \( A \), that is
\[ \sigma_i \cdot A t_i = 0 \]  
(2.4.9)
The minimum \( F[x_i] \) occurs at the centre of a manifold of ellipses in the plane passing through \( r_i \) and \( t_i \) (Figure 2.6).
for an $a_i$ given from (2.4.8) by

$$a_i = \frac{r_i \cdot \sigma_i}{\sigma_i \cdot A \sigma_i} \quad . \quad (2.4.10)$$

The iterative scheme (2.4.4) and the residual (2.4.6) become respectively:

$$x_{i+1} = x_i + a_i \sigma_i \quad . \quad (2.4.11)$$

and

$$r_{i+1} = r_i - a_i A \sigma_i \quad . \quad (2.4.12)$$

The residual $r_{i+1}$ is normal to $F[x_{i+1}]=c$ at the point $x_{i+1}$, the centre of the ellipses in the plane at which the function $F[x]$ is minimal and is also orthogonal to the plane, which is tangent to the surface at point $x_{i+1}$, and to the vectors $\sigma_i, r_i, t_i$, which lie in this plane.

Choosing $F[x_{i+1}]=c$ at the point $x_{i+1}$, the plane at which the function $F[x]$ is to the plane, which is tangent to the vectors $\sigma_i, r_i, t_i$, which lie

from (2.4.9) we obtain

$$\sigma_i \cdot A \sigma_i = r_i \cdot A r_i + \beta_i \cdot t_i \cdot A t_i = 0 \quad , \quad (2.4.14)$$

or

$$\beta_i = -\frac{r_i \cdot A t_i}{t_i \cdot A t_i} \quad . \quad (2.4.15)$$

Considering that the tangent vector $t_{i+1}$ lies in the plane of $r_i, t_i$ and with the choice

$$t_{i+1} = \sigma_i \quad , \quad (2.4.16)$$

from (2.4.13), (2.4.15) we have respectively,

$$\sigma_i = r_i + \beta_i \sigma_{i-1} \quad , \quad (2.4.17)$$

and

$$\beta_i = -\frac{r_i \cdot A \sigma_{i-1}}{\sigma_{i-1} \cdot A \sigma_{i-1}} \quad . \quad (2.4.18)$$

The equations (2.4.10)-(2.4.12) and (2.4.17),(2.4.18) imply the following orthogonality relations:

$$r_{i+1} \cdot r_i = 0 \quad ; \quad r_{i+1} \cdot \sigma_i = 0 \quad , \quad (2.4.19)$$
and
\[ r_{i+1} \cdot r_{i-1} = 0; \quad \sigma_{i} \cdot A \sigma_{i-1} = 0. \] (2.4.19a)

Moreover, it can be shown [39] that all \( r_i \) are mutually orthogonal and all \( \sigma_i \) are mutually conjugate, i.e.,
\[ r_i \cdot r_j = 0; \quad \sigma_i \cdot A \sigma_j = 0, \text{ for } i \neq j. \] (2.4.20)

The iterative scheme of Conjugate Gradient method begins with an arbitrary \( x_0 \) as initial vector and computation of \( r_0 \).

Since equations (2.4.19) hold for all \( r_i \), then \( \sigma_0 = r_0 \) i.e., \( \beta_0 = 0 \).
The method is exact in \( N \)-iterative steps with \( |r_{i+1}| < |r_i| \) at each step.

The procedure amounts to passing \( N \)-mutually orthogonal planes through the ellipsoid \( F[x] \) and finding the center of the resultant manifold of ellipses in each plane.

From equations (2.4.10)-(2.4.12) and (2.4.17), (2.4.18), a second order iterative scheme can be derived as follows:
\[ r_{i+1} = (1 + \frac{a_i \beta_{i+1}}{a_{i-1}}) r_i - \frac{a_i \beta_i}{a_{i-1}} r_{i-1} - a_i A r_i, \] (2.4.21)
\[ x_{i+1} = (1 + \frac{a_i \beta_{i+1}}{a_{i-1}}) x_i - \frac{a_i \beta_i}{a_{i-1}} x_{i-1} + a_i (Ax_i - s) \] (2.4.22)
and
\[ \sigma_{i+1} = (1 + \beta_{i+1}) \sigma_i - \beta_{i} \sigma_{i-1} - a_i A \sigma_i \] (2.4.23)
with \( \beta_0 = 0 \).

Note that equation (2.4.22) is of the same form as the second order Richardson and Chebyshev procedures given by (2.3.50) and (2.3.60) respectively, except that here the coefficients are variables (instead of constants in Richardson case) chosen to minimize the quadratic function \( F[x] \) instead to generate the polynomial (Chebyshev case).

Although the Conjugate Gradient method theoretically gives an exact answer in \( N \)-iterative steps, the residuals are not truly
orthogonal in the actual practice because of the round-off errors. So, generally we have $r_{i-1} \neq 0$ but it might be expected to be very close to zero. If the obtained solution $x_{i+1}$ becomes too inaccurate because of round-off errors then the iteration either terminated if $r_i$ is sufficiently small or could be restarted with $x_{i+1}$ as initial guess.

Note that the inner products needed for the calculation of parameters $a_i, \beta_i$ are considerable time-consuming operations and the method depends upon orthogonality (the error reduction may become very slow if orthogonality breaks down [32]).

**SECTION B: THE PARABOLIC PROBLEM**

### 2.5 A PARABOLIC 2D-EXAMPLE

We consider the quasi-linear (see [12]) diffusion equation of the following form:

$$\frac{\partial U}{\partial t} = \frac{\partial}{\partial x}(\Lambda \frac{\partial U}{\partial x}) + \frac{\partial}{\partial y}(\Lambda \frac{\partial U}{\partial y}), \quad \Lambda = \Lambda(x,y),$$

(2.5.1)

in the region $R = \{x, y; 0 < x, y < 1\}$

with initial condition

$$U(x, y, 0) = f(x, y), \quad (x, y) \in R$$

(2.5.2)

and subject to the boundary condition

$$U(x, y, t) = \gamma(x, y, t), \quad (x, y, t) \in C \times [0 < t < T]$$

(2.5.3)

where $C$ is the boundary of $R$.

The region $R$ is covered by a rectilinear net with mesh spacings $h_x, h_y, \Delta t$ in the $x, y, t$ directions respectively and mesh points $(x_i, y_j, t_k)$ where:
46

\[ x_i = ih_x, \quad i \in [0, p] \]
\[ y_j = jh_y, \quad j \in [0, m] \]
\[ t_k = k.\Delta t, \quad k \in [0, T/\Delta t] \].

(2.5.4)

Let \( u_{i,j}^{(k)} \) and \( U_{i,j}^{(k)} \) denote the approximate (finite difference) and exact solution respectively of equation (2.5.1) at the point \((ih_x, jh_y, k\Delta t)\).

2.6 DERIVATION OF FINITE DIFFERENCE FORMULAE

Definition 2.6.1

Consider a function \( y = f(x) \) and a constant interval size \( h = x_{n+1} - x_n \). Let \( y_n = f(x_n) \), then if \( \Delta, \nabla, \delta, \mu \) be the forward, backward, central and average difference operators respectively, we have:

\[ \Delta y_n = y_{n+1} - y_n, \quad \nabla y_n = y_n - y_{n-1} \]

and

\[ \delta y_n = y_{n+1} - y_{n-1}, \quad \mu y_n = \frac{1}{2} [y_{n+1} + y_{n-1}] \].

(2.6.0)

We assume that any given partial derivatives of \( U \) are continuous and uniformly bounded for all \( x, y, t \in \mathbb{R} \).

The exact difference replacement of equation (2.5.1) can be obtained from the following Taylor expansion

\[ U(x, y, t + \Delta t) = (1 + \Delta t \frac{\partial}{\partial t} + \frac{(\Delta t)^2}{2!} \frac{\partial^2}{\partial t^2} + \frac{(\Delta t)^3}{3!} \frac{\partial^3}{\partial t^3} \ldots)U(x, y, t) \]

\[ = \exp(\Delta t \frac{\partial}{\partial t}) U(x, y, t) \].

(2.6.1)

It is well known [43], [33] that

\[ \frac{\partial}{\partial x} = \frac{2}{h_x} \sinh^{-1}(\delta x/2) = \frac{1}{h_x}(\delta x - \frac{\delta^3 x}{2.3!} + \frac{\delta^5 x}{2^4.5!} - \ldots) \]

(2.6.2)

\[ \frac{\partial}{\partial y} = \frac{2}{h_y} \sinh^{-1}(\delta y/2) = \frac{1}{h_y}(\delta y - \frac{\delta^3 y}{2.3!} + \frac{\delta^5 y}{2^4.5!} - \ldots) \]

(2.6.2a)

and

\[ \frac{\partial^2}{\partial x^2} = \frac{1}{2} \left( \frac{\delta^2 x}{12} + \frac{1}{90} \delta^6 x \ldots \right) \]

(2.6.3)

\[ \frac{\partial^2}{\partial y^2} = \frac{1}{2} \left( \frac{\delta^2 y}{12} + \frac{1}{90} \delta^6 y \ldots \right) \]

(2.6.3a)
where

$$\delta u_{i,j}^{(k)} = (u_{i+1,j}^{(k)} - u_{i,j}^{(k)}) - (u_{i,j}^{(k)} - u_{i-1,j}^{(k)}),$$  \(2.6.4\)

$$\delta^2 u_{i,j}^{(k)} = (u_{i+1,j}^{(k)} - 2u_{i,j}^{(k)} + u_{i-1,j}^{(k)}),$$  \(2.6.5\)

and so on.

We consider now the following approximations

$$\frac{\partial}{\partial x} (\Lambda \delta_x) u_{i,j}^{(k)} = \Lambda_{i+1,j}^{(k)} - \Lambda_{i,j}^{(k)} (u_{i,j}^{(k)} - u_{i-1,j}^{(k)}),$$  \(2.6.6\)

$$\frac{\partial}{\partial y} (\Lambda \delta_y) u_{i,j}^{(k)} = \Lambda_{i,j+1}^{(k)} - \Lambda_{i,j}^{(k)} (u_{i,j}^{(k)} - u_{i,j-1}^{(k)}).$$  \(2.6.7\)

$$\frac{\partial}{\partial x} (\Lambda \delta_x) u_{i,j}^{(k)} = \Lambda_{i+1,j}^{(k)} - \Lambda_{i,j}^{(k)} (u_{i,j}^{(k)} - u_{i-1,j}^{(k)}),$$  \(2.6.8\)

### 2.7 Explicit and Implicit Methods

It can be easily seen that equation (2.5.1) can be written as

$$u_{i,j}^{(k+1)} = \exp(\Delta t [\frac{\partial}{\partial x} (\Lambda \delta_x) + \frac{\partial}{\partial y} (\Lambda \delta_y)]) u_{i,j}^{(k)}.$$  \(2.7.1\)

Then, the substitution of relationships (2.6.6) into equation (2.7.1) and expansion of the right hand side of (2.7.1), neglecting terms of order \(\Delta t^2\) and above, leads to the following standard explicit formula, involving five grid points at \(t=k \Delta t\),

$$u_{i,j}^{(k+1)} = \{1 + r_x^2 (\Lambda_{i+1,j}^{(k)} + \Lambda_{i-1,j}^{(k)}) + r_y^2 (\Lambda_{i,j+1}^{(k)} + \Lambda_{i,j-1}^{(k)})\} u_{i,j}^{(k)}.$$  \(2.7.2\)

where

$$r_x = \Delta t / h_x^2, \quad r_y = \Delta t / h_y^2,$$  \(2.7.3\)

are the mesh ratios in the \(x, y\) directions and \(u_{i,j}^{(k)}\) denotes \(u_{i,j}^{(k)}\).

Then, combination of (2.6.7), (2.6.8), (2.7.2) gives

$$u_{i,j}^{(k+1)} = [1 - r_x (\Lambda_{i+1,j}^{(k)} + \Lambda_{i-1,j}^{(k)}) - r_y (\Lambda_{i,j+1}^{(k)} + \Lambda_{i,j-1}^{(k)})] u_{i,j}^{(k)} +$$

$$+ r_x (\Lambda_{i+1,j}^{(k)} u_{i+1,j}^{(k)} + \Lambda_{i-1,j}^{(k)} u_{i-1,j}^{(k)}) + r_y (\Lambda_{i,j+1}^{(k)} u_{i,j+1}^{(k)} + \Lambda_{i,j-1}^{(k)} u_{i,j-1}^{(k)}).$$  \(2.7.4\)
In order to find the local truncation error of (2.7.4), we consider the error vector at the mesh point \((i_{hx}, j_{hy}, k_{At})\) defined by:

\[ e_{i,j}^{(k)} = u_{i,j}^{(k)} - u_{i,j} \]

(2.7.5)

Using Taylor series expansion (see section 1.2, equation (2.6.1)) and from (2.7.4), (2.7.5) we have

\[
e_{i,j}^{(k+1)} = \left[ 1 - r_x (\Lambda_{i+\frac{1}{2},j}^{(k)} + \Lambda_{i-\frac{1}{2},j}^{(k)}) - r_y (\Lambda_{i,j+\frac{1}{2}}^{(k)} + \Lambda_{i,j-\frac{1}{2}}^{(k)}) \right] e_{i,j}^{(k)} + \\
\left[ r_x (\Lambda_{i+\frac{1}{2},j}^{(k)} e_{i+\frac{1}{2},j}^{(k)} + \Lambda_{i-\frac{1}{2},j}^{(k)} e_{i-\frac{1}{2},j}^{(k)}) + r_y (\Lambda_{i,j+\frac{1}{2}}^{(k)} e_{i,j+\frac{1}{2}}^{(k)} + \Lambda_{i,j-\frac{1}{2}}^{(k)} e_{i,j-\frac{1}{2}}^{(k)}) \right] e_{i,j}^{(k)} + \\
\frac{(\Delta t)^2}{2} \left[ \frac{\partial^2 u}{\partial x^2} \right]_{i,j}^{(k)} - (\Delta t) \left[ \frac{\partial^4 u}{\partial x^4} \right]_{i,j}^{(k)} + \frac{(\Delta t)^2}{2} \left[ \frac{\partial^2 u}{\partial y^2} \right]_{i,j}^{(k)} - (\Delta t) \left[ \frac{\partial^4 u}{\partial y^4} \right]_{i,j}^{(k)} + \ldots
\]

(2.7.6)

Assuming now that the adjacent points to an internal point \((i,j)\) belong to the region \(R\) under consideration then from (2.7.6) we get that the local truncation error of (2.7.4) is

\[ O(\Delta t^2 + \Delta t(h_{x}^2 + h_{y}^2)) \]

(2.7.7)

Note that if the region \(R\) under consideration is non-rectangular then a special treatment for the internal grid points adjacent to the boundary is required.

![Figure 2.7](image-url)
In such a case, let

\[(i+1,j), (i-\varepsilon_1,j), (i,j+\varepsilon_2), (i,j-1)\]

be the adjacent points to the internal point \((i,j)\). The mesh points
\[(i-\varepsilon_1,j), (i,j+\varepsilon_2)\] are not exactly distance \(h_x, h_y\) away but \(\varepsilon_1 h_x, \varepsilon_2 h_y\) respectively with \(\varepsilon_1, \varepsilon_2 \in (0,1]\), (Figure 2.7).

Then the following finite difference approximations are obtained

\[
\frac{\partial}{\partial x} \left( \Lambda \frac{\partial U}{\partial x} \right) = \frac{1}{1+\varepsilon_1} \frac{2}{h_x} \left[ \frac{u_{i+1,j} - u_{i,j}}{h_x} - \Delta_i - \frac{u_{i,j} - u_{i-1,j}}{\varepsilon_1 h_x} \right]
\]

\[
\frac{\partial}{\partial y} \left( \Lambda \frac{\partial U}{\partial y} \right) = \frac{1}{1+\varepsilon_2} \frac{2}{h_y} \left[ \frac{u_{i,j+\varepsilon_2} - u_{i,j}}{\varepsilon_2 h_y} - \Delta_j - \frac{u_{i,j} - u_{i,j-1}}{h_y} \right]
\]

(2.7.8)

and the difference equation valid at an internal point such as
\((i,j)\) in Figure 2.7, is given by an analogous equation to (2.7.4) as follows:

\[
u_{i,j}^{(k+1)} = \left[ 1 - \frac{2r_x}{1+\varepsilon_1} \left( \Lambda \right)_{i+\frac{1}{2},j} + \frac{1}{\varepsilon_1} \left( \Lambda \right)_{i-\frac{1}{2},j/2,j} - \frac{2r_y}{1+\varepsilon_2} \left( \Lambda \right)_{i,j+\frac{1}{2},j+\frac{1}{2}} + \frac{2r_y}{1+\varepsilon_2} \left( \Lambda \right)_{i,j} \right] \nu_{i,j+1}^{(k)}
\]

\[
+ \frac{2r_x}{1+\varepsilon_1} \left( \Lambda \right)_{i,j+\frac{1}{2}} \nu_{i+1,j}^{(k)} + \frac{1}{\varepsilon_1} \left( \Lambda \right)_{i-\frac{1}{2},j} \nu_{i,j}^{(k)}
\]

\[
+ \frac{2r_y}{1+\varepsilon_2} \left( \Lambda \right)_{i+\frac{1}{2},j} \nu_{i,j+1}^{(k)} + \frac{1}{\varepsilon_2} \left( \Lambda \right)_{i,j} \nu_{i,j}^{(k)}
\]

(2.7.8a)

Since this thesis is concerned only withImplicit methods, the Explicit method has been included only for completeness. It is well known [52],[43] that the latter method, in the case of constant coefficients \(\Lambda(x,y)\equiv1\) in equation (2.5.1), is conditionally stable against the growth of rounding errors only when

\[
\frac{\Delta t}{h_x^2} \text{ and } \frac{\Delta t}{h_y^2}.
\]

Given that the application of Explicit
difference methods for the solution of initial boundary value problems, in two or more space dimensions, is limited because of poor stability properties, the Implicit methods with superior stability properties are always used.

For example, formula (2.7.1) can be written as

$$\exp\left(-At \left[ \frac{\partial}{\partial x} (\Lambda \frac{\partial}{\partial x}) + \frac{\partial}{\partial y} (\Lambda \frac{\partial}{\partial y}) \right] \right) u^{(k+1)} = u^{(k)} \quad (2.7.9)$$

where $U^{(k)}$ denotes $U_{i,j}^{(k)}$. Then, combination of (2.6.6), (2.7.9) leads to

$$\exp(-r_x \delta_x (\Lambda \delta_x)) \exp(-r_y \delta_y (\Lambda \delta_y)) u^{(k+1)} \approx u^{(k)} \quad (2.7.9a)$$

where the mesh ratios $r_x, r_y$ and $\delta_x (\Lambda \delta_x), \delta_y (\Lambda \delta_y)$ are given by (2.7.3) and (2.6.7), (2.6.8) respectively. The expansion of the left-hand side of equation (2.7.9a), neglecting terms of order $\Delta t^2$ and above, gives the following two dimensional 'fully implicit' scheme:

$$[1-r_x \delta_x (\Lambda \delta_x) - r_y \delta_y (\Lambda \delta_y)] u^{(k+1)} = u^{(k)} + O(\Delta t^2 + \Delta t(h_x^2 + h_y^2)) \quad (2.7.9b)$$

Advantages of the above formula is stability but we go on to discuss the 'Crank-Nicolson' formula in detail, which is not only stable but also of higher order accuracy.

The exact difference replacement formula (2.7.1) can be re-written as

$$\exp\left( -\frac{1}{2} \Delta t \left[ \frac{\partial}{\partial x} (\lambda \frac{\partial}{\partial x}) + \frac{\partial}{\partial y} (\lambda \frac{\partial}{\partial y}) \right] \right) u^{(k+1)} =$$

$$= \exp\left( -\frac{1}{2} \Delta t \left[ \frac{\partial}{\partial x} (\lambda \frac{\partial}{\partial x}) + \frac{\partial}{\partial y} (\lambda \frac{\partial}{\partial y}) \right] \right) u^{(k)} \quad (2.7.10)$$

and assuming that the approximations (2.6.6) are substituted in (2.7.10) we get

$$\exp\left( -\frac{1}{2} r_x \delta_x (\Lambda \delta_x) \right) \exp\left( -\frac{1}{2} r_y \delta_y (\Lambda \delta_y) \right) u^{(k+1)} =$$

$$= \exp\left( \frac{1}{2} r_x \delta_x (\Lambda \delta_x) \right) \exp\left( \frac{1}{2} r_y \delta_y (\Lambda \delta_y) \right) u^{(k)} \quad (2.7.10a)$$
If we expand both sides of equation (2.7.10a), neglecting terms of order $\Delta t^2$ and above, and average the operation over two time intervals, we obtain the following 'Crank-Nicolson' formulae in two space dimensions [13]:

\[
[1-\frac{r}{2} \delta_x^x (\Delta \delta^x_x) - \frac{r}{2} \delta_y^y (\Delta \delta^y_y)] u^{(k+1)} = [1+\frac{r}{2} \delta_x^x (\Delta \delta^x_x) + \frac{r}{2} \delta_y^y (\Delta \delta^y_y)] u^{(k)} + O(\Delta t^3 + \Delta t h_x^2 + h_y^2).
\]  

(2.7.10b)

It is well known [52] that the Crank-Nicolson implicit formula is unconditionally stable against the growth of the rounding errors.

A great number of implicit finite difference methods have been proposed for the solution of initial boundary value problems involving linear parabolic equations in the region $R$, [52], [27], [28], [21]. Some governing criteria for the choice of the best implicit finite difference method in a particular problem are:

(i) the nature of the coefficients in the parabolic equation,

(ii) the shape of the region $R$ under consideration,

(iii) the type of boundary condition on $C$ for $t>0$.

From relationships (2.6.6)-(2.6.8) and Crank-Nicolson finite difference formula (2.7.10b) the finite difference discretization of equation (2.5.1) on the chosen grid (see Figure 2.8), can be written as a series of five-point linear finite difference equations of the form:

\[
\begin{align*}
d_{i,j} u_{i,j-1,k+1} + a_{i,j} u_{i-1,j,k+1} + b_{i,j} u_{i,j,k+1} + c_{i,j} u_{i+1,j,k+1} + e_{i,j} u_{i,j+1,k+1} &= s_{i,j,k} \\
& \quad \text{for } i \in [1,p-1], j \in [1,m-1],
\end{align*}
\]  

(2.7.11)
where,
\[
d_{i,j} = -\frac{r_y A_{i,j}}{2}; \quad e_{i,j} = -\frac{r_y A_{i,j+\frac{1}{2}}}{2}; \\
a_{i,j} = -\frac{r_x A_{i-\frac{1}{2},j}}{2}; \quad c_{i,j} = -\frac{r_x A_{i+\frac{1}{2},j}}{2}; \\
b_{i,j} = 1 + \frac{r_y}{2}[A_{i,j-\frac{1}{2}} + A_{i,j+\frac{1}{2}}] + \frac{r_x}{2}[A_{i-\frac{1}{2},j} + A_{i+\frac{1}{2},j}];
\]

and
\[
s_{i,j,k} = u_{i,j,k} + \frac{r_x}{2}[A_{i+\frac{1}{2},j} (u_{i+1,j,k} - u_{i,j,k}) - A_{i-\frac{1}{2},j} (u_{i-1,j,k} - u_{i,j,k})] + \frac{r_y}{2}[A_{i,j+\frac{1}{2}} (u_{i,j+1,k} - u_{i,j,k}) - A_{i,j-\frac{1}{2}} (u_{i,j-1,k} - u_{i,j,k})] + \frac{r_y}{2}[A_{i,j,k+\frac{1}{2}} (u_{i,j,k+1} - u_{i,j,k}) - A_{i,j,k-\frac{1}{2}} (u_{i,j,k-1} - u_{i,j,k})];
\]

with \( r_x = \Delta t/h_x^2 \), \( r_y = \Delta t/h_y^2 \).

FIGURE 2.8

Grouping the above system of finite difference equations into matrix form, we obtain a sparse, quindiaogonal matrix of order \( n=(m-1)(p-1) \) of the following form:
where \( c_q = 0 \) if \( q = t(m-1), 1 \leq t \leq p-2 \)
\[ a_q = 0 \] if \( q = t(m-1)+1, 1 \leq t \leq p-2 \)

and the obtained system

\[
Au = s \tag{2.7.17}
\]

has to be solved at each time step, to give the required solution to the problem.

**Remark**

A fully implicit finite difference formula in two space dimensions is given in (2.7.9b). In such a case, the coefficients of the five-point linear finite difference equation (2.7.11) are given as follows:

\[
d_{i,j} = -r_i A_{i,j-\frac{1}{2}}; \quad e_{i,j} = -r_i A_{i,j+\frac{1}{2}} \tag{2.7.18}
\]
\[
a_{i,j} = -r_i A_{i-\frac{1}{2},j}; \quad c_{i,j} = -r_i A_{i+\frac{1}{2},j} \tag{2.7.19}
\]
\[
b_{i,j} = 1+r_x(A_{i+\frac{1}{2},j} A_{i-\frac{1}{2},j})+r_y(A_{i,j+\frac{1}{2}} A_{i,j-\frac{1}{2}}) \tag{2.7.20}
\]

and

\[
s_{i,j,k} = u_{i,j,k} \tag{2.7.21}
\]
2.8 A PARABOLIC 3D-EXAMPLE

We consider now the application of the finite difference method to obtain an approximate solution for the parabolic three dimensional P.D.E. defined by:

\[
\frac{\partial}{\partial x} (A(x,y,z) \frac{\partial u}{\partial x}) + \frac{\partial}{\partial y} (C(x,y,z) \frac{\partial u}{\partial y}) + \frac{\partial}{\partial z} (E(x,y,z) \frac{\partial u}{\partial z}) = F(x,y,z) \frac{\partial u}{\partial t} + Q(x,y,z),
\]

(2.8.1)
in the region \( \bar{R} = R \times [0 < t < T] \),

(2.8.2)
where

\[ R = \{ (x, y, z); 0 < x, y, z < 1 \}, \]

(2.8.3)
with the initial condition,

\[ u(x,y,z,0) = f(x,y,z), \quad (x,y,z) \in R \]

(2.8.4)
subject to the boundary conditions

\[ u(x,y,z,t) = g(x,y,z,t), \quad (x,y,z,t) \in \bar{C} \times [0 < t < T] \]

(2.8.5)
where \( \bar{C} \) is the boundary of \( R \), and \( A, C, E, F \) are known coefficients (in general they are functions of position). \( Q \) is a source term, \( x, y, z \) are the position co-ordinates and \( u \) is the dependent variable.

We cover the region \( R \) with a volumetric grid system \( R_h \), with spacings \( h_x, h_y, h_z \) and \( \ell \) in \( X, Y, Z \) and \( t \) directions respectively and mesh points \( (x_i, y_j, z_k, t_r) \), defined by

\[
R_h = \{ (ih_x, jh_y, kh_z, r\ell): \begin{cases} 
0 \leq i \leq \xi + 1 \\
0 \leq j \leq \theta + 1 \\
0 \leq k \leq \xi + 1 \\
0 \leq r \leq T/\ell 
\end{cases} \}.
\]

(2.8.6)
The three dimensional molecule is shown in Figure 2.9.
Assuming that \( P(i, j, k) \) and \( u_{i,j,k} \) denote the grid point \( P(\text{ih}_x, \text{jh}_y, \text{kh}_z) \) and \( u(\text{ih}_x, \text{jh}_y, \text{kh}_z) \) respectively, discrete approximations to the partial derivatives in (2.8.1) are used to derive the following linear finite difference equations at the point \( P(i, j, k) \)

\[
A(x+i\frac{h_x}{2}, y, z) \left[ \frac{u(x+h_x, y, z) - u(x, y, z)}{h_x^2} \right] - A(x-i\frac{h_x}{2}, y, z) \left[ \frac{u(x-h_x, y, z) - u(x, y, z)}{h_x^2} \right] + \\
+ C(x, y+j\frac{h_y}{2}, z) \left[ \frac{u(x, y+h_y, z) - u(x, y, z)}{h_y^2} \right] - C(x, y-j\frac{h_y}{2}, z) \left[ \frac{u(x, y-h_y, z) - u(x, y, z)}{h_y^2} \right] + \\
+ E(x, y, z+k\frac{h_z}{2}) \left[ \frac{u(x, y, z+h_z) - u(x, y, z)}{h_z^2} \right] - E(x, y, z-k\frac{h_z}{2}) \left[ \frac{u(x, y, z-h_z) - u(x, y, z)}{h_z^2} \right] - \\
- \frac{1}{\Delta t} F(x, y, z) [u(x, y, z) - \bar{u}(x, y, z)] = Q(x, y, z) \quad (2.8.7)
\]
where \( u(x,y,z) \) and \( \bar{u}(x,y,z) \) are the values of the dependent variable at the \((k+1)\) and \(k\) time levels respectively and \( \Delta t \) is the time step.

When the seven-point, three dimensional molecule (Figure 2.9) is used, we obtain the following seven-point finite difference equation:

\[
\begin{align*}
D_{i,j,k}u_{i,j,k} &+ R_{i,j,k}u_{i+1,j,k} + L_{i,j,k}u_{i-1,j,k} + T_{i,j,k}u_{i,j+1,k} + F_{i,j,k}u_{i,j,k+1} + \sum_{l=1}^{Q} I_{i,j,k}u_{i,j,k} = S_{i,j,k},
\end{align*}
\]

for \( i \in [1,\xi], \ j \in [1,\eta], \ k \in [1,\zeta], \)

where

\[
D_{i,j,k} = \frac{h_x}{h_z} (A_{i+\frac{1}{2},j,k} + B_{i,j,k} + C_{i,j+\frac{1}{2},k}) - (C_{i,j+\frac{1}{2},k} + C_{i,j-\frac{1}{2},k} + D_{i,j,k}),
\]

\[
I_{i,j,k} = E_{i,j,k+\frac{1}{2}} - E_{i,j,k-\frac{1}{2}},
\]

\[
T_{i,j,k} = C_{i,j+\frac{1}{2},k} - C_{i,j-\frac{1}{2},k},
\]

\[
R_{i,j,k} = A_{i+\frac{1}{2},j,k} - A_{i-\frac{1}{2},j,k},
\]

and

\[
S_{i,j,k} = (O_{i,j,k} - \frac{1}{\Delta t} F_{i,j,k} \bar{u}_{i,j,k})h_x h_y h_z.
\]

We order the points of \( R_h \) with increasing values of \( j \), then \( i \) and then \( k \). If we order equation (2.8.8) in the same sequence then a set of difference equations is obtained which may be expressed in matrix notation as:

\[
\Omega \tilde{u} = \tilde{s}
\]

where \( \tilde{u} \) and \( \tilde{s} \) are \([\xi \eta \zeta \times 1]\) column vectors, which consist of the unknown approximate solutions \( u_{i,j,k} \) and the right hand side.
quantity (2.8.13) plus the known boundary values respectively, and $\Omega$ is a seven diagonal, sparse matrix of order $n=\xi\theta\xi$ and bandwidths $m=\theta+1$, $p=\theta\xi+1$.

\[
\begin{bmatrix}
    b_1 & c_1 & \tau_1 & r_1 \\
    a_2 & b_2 & c_2 & \tau_2 & r_2 & 0 \\
    & a_3 & b_3 & c_3 & \tau_3 & r_3 & 0 \\
    & & & & & & \vdots \\
    & & & & & & \vdots \\
    & & & & & & \vdots \\
    v_{m-1} & v_m & & & & & r_{n-p+1} \\
    v_{m+1} & & & & & & \tau_{n-m+1} \\
    & & v_{m+2} & & & & \vdots \\
    & s_{p+1} & & v_{m+2} & & & \vdots \\
    & & & s_{p+2} & & & \vdots \\
    & & & & s_{n-1} & v_{n-1} & a_{n-1} b_{n-1} c_{n-1} \\
    0 & s_n & v_n & a_n b_n c_n & & & 0 \\
    & & & & & & 0 \\
\end{bmatrix}
\begin{bmatrix}
u_{1,1,1} \\
u_{1,2,1} \\
\vdots \\
u_{1,\theta,1} \\
u_{2,1,1} \\
\vdots \\
u_{2,\theta,1} \\
u_{2,\zeta,\theta,\xi} \\
\end{bmatrix}
\begin{bmatrix}s_{1,1,1} \\
s_{1,2,1} \\
\vdots \\
s_{1,\theta,1} \\
s_{2,1,1} \\
\vdots \\
s_{2,\theta,1} \\
s_{\zeta,\theta,\xi} \\
\end{bmatrix}
\]

(2.8.15)

where: $c_q = 0$, if $q=\theta$, $1 \leq \xi \leq \xi - 1$

$a_q = 0$, if $q=\theta+1$, $1 \leq \xi \leq \xi - 1$

and which has to be solved to give the required solution to the problem at each time step.
Chapter 3

Basic Algorithms for Two and Three Dimensional P.D.E.'s
3.1 INTRODUCTION

In this Chapter we introduce algorithmic solution methods for the large sparse linear systems derived from finite-difference discretization of parabolic and elliptic p.d.e's in both two and three space dimensions. The coefficient matrix is shown to be factorized exactly to yield direct algorithmic procedures for the finite difference solution.

3.2 THE ALGORITHM FOR THE SOLUTION OF LARGE, UNSYMMETRIC, QUINDIAGONAL SPARSE LINEAR SYSTEMS (THE LUBOT ALGORITHM)

An investigation into the current implicit solution processes involving the diffusion equation with one space dimension reveals the fact that extensive use is made of the tridiagonal matrix algorithm [57] for solving the three term finite difference equations.

For the two dimensional case, the implicit methods lead to the requirement that large sparse matrices have to be solved for each time interval, a compact Gaussian elimination process without interchanges and an approximate elimination scheme have been derived [24] with the use of an echelon "moving frame" process to devise an efficient algorithmic process by which the Gaussian factorization procedure is carried out.

In this section we introduce a sparse triangular factorization LU method for the solution of similar problems by use of a matrix "bordering" technique [34], [26].

Let $A$ (a large order, diagonally dominant, unsymmetric, quindiagonal, sparse matrix) be factorized into the product of two matrices $L, U$ i.e.,

$$ A = L \cdot U \quad (3.2.1) $$

where $A$ is defined in (2.7.16)
L is a strictly lower triangular matrix and
U is a strictly upper triangular matrix with unit diagonal elements given below.

The coefficient matrix $A$ can be written in the following partitioned form:

$$
A = \begin{pmatrix}
0 & \tau_1 & \tau_2 & \cdots & \tau_{n-m+1} \\
0 & 0 & \ddots & \ddots & \vdots \\
0 & 0 & \ddots & \ddots & \vdots \\
0 & 0 & \ddots & \ddots & \vdots \\
0 & 0 & \cdots & 0 & 0
\end{pmatrix}
$$

$$
\begin{pmatrix}
A_11 & A_{12} \\
A_{21} & A_{22}
\end{pmatrix}
$$

(3.2.2)
The LU triangular matrices in (3.2.1) are of the following form:

\[
L = \begin{bmatrix}
\omega_1 \\
\beta_1 \omega_2 \\
\beta_2 \omega_3 \\
\gamma_{1,1} & \gamma_{2,1} & \gamma_{3,1} & \cdots & \gamma_{m-2,1} (\gamma_{m-1,1} + \beta_{m-1}) & \omega_m \\
\gamma_{1,2} & \gamma_{2,2} & \gamma_{3,2} & \cdots & \gamma_{m-2,2} (\gamma_{m-1,2} + \beta_{m-1}) & \omega_{m+1} \\
\gamma_{1,n-m+1} & \gamma_{2,n-m+1} & \gamma_{3,n-m+1} & \cdots & \gamma_{m-1,n-m+1} + \beta_{n-1} & \omega_n
\end{bmatrix}
\]

\[
= \begin{bmatrix}
L_{11} & 0 \\
L_{21} & L_{22}
\end{bmatrix}
\]

(3.2.3)
Replacing the right hand side of (3.2.1) by its partitioned forms leads to

\[
LU = \begin{bmatrix}
    L_{11} & 0 \\
    L_{21} & L_{22}
\end{bmatrix}
\begin{bmatrix}
    U_{11} & U_{12} \\
    0 & U_{22}
\end{bmatrix}
= \begin{bmatrix}
    L_{11}U_{11} & L_{11}U_{12} \\
    L_{21}U_{11} & L_{21}U_{12} + L_{22}U_{22}
\end{bmatrix}
\]

(3.2.5)

and on equating with the partitioned form of A in (3.2.2) we obtain the identities:

\[
A_{11} = L_{11}U_{11} ; \quad A_{12} = L_{11}U_{12} ;
\]

(3.2.6a)

\[
A_{21} = L_{21}U_{11} ; \quad A_{22} = L_{21}U_{12} + L_{22}U_{22}.
\]

(3.2.6b)

The elements of \( L_{11} \) and \( U_{11} \) are well known and may be easily obtained by applying the tridiagonal algorithm [57] as follows:

\[
\omega_1 = b_1 ; \quad \beta_1 = a_2 ; \quad g_1 = c_1/\omega_1 ,
\]

for \( i=2,3,\ldots,m-2 \)

\[
\omega_i = b_i - \beta_i-1g_i-1 ; \quad \beta_i = a_i+1 ; \quad g_i = c_i/\omega_i
\]

(3.2.7a)

and

\[
\omega_{m-1} = b_{m-1} - \beta_{m-2}g_{m-2}.
\]

(3.2.7c)

The elements of \( L_{21}, U_{21}, L_{22} \) and \( U_{22} \) can be obtained from the following relations:

for \( j=1,2,\ldots,n-m+1 \)

\[
h_{1,j} = \tau_j/\omega_j ; \quad \gamma_{1,j} = \nu_{j+m-1}
\]

and

\[
g_{m+j-2} = c_{m+j-2}/\omega_{m+j-2} ; \quad \beta_{m+j-2} = a_{m+j-1},
\]

(3.2.8)

(3.2.9)

whilst for \( j<m-2 \)

\[
h_{i,j} = -\beta_{i+j-2}h_{i,j}/\omega_{i+j-1}, \text{ for } i=2,3,\ldots,m-j
\]

(3.2.10)

and

\[
\gamma_{i,j} = -g_{i+j-2}\gamma_{i-1,j}, \text{ for } i=2,3,\ldots,m-j.
\]

(3.2.11)

Then, for all \( i>1 \)

\[
\gamma_{i,j} = -g_{i+j-2}\gamma_{i-1,j} - \sum_{k=1}^{i-1} \gamma_{k,j}h_{k-i+m,i+j-m}
\]

(3.2.12)

\[
h_{i,j} = (-\beta_{i+j-2}h_{i-1,j} - \sum_{k=1}^{i-1} \gamma_{k-i+m,i+j-m}h_{k,j})/\omega_{i+j-1},
\]

(3.2.13)
for either \( i=(m-j+1), (m-j+2), \ldots, (m-1) \) and all \( j \leq m-1 \)

or \( i=2, 3, \ldots, m-1 \) for \( j>m-1 \).

Then, for \( i=m-1 \)

\[
\omega_{m+j-1} = b_{m+j-1} - \beta_{i+j-1} h_{i,j} - g_{i+j-1} \gamma_{i,j} - h_{i-j} g_{i+j-1} - \sum_{k=1}^{i} \gamma_{k,j} h_{k,j},
\]

for \( m+j-1 = b_{m+j-1} - \beta_{i+j-1} h_{i,j} - g_{i+j-1} \gamma_{i,j} - h_{i-j} g_{i+j-1} - \sum_{k=1}^{i} \gamma_{k,j} h_{k,j} \).

The original linear system

\[
Auu = s
\]

can then be solved by rewriting (3.2.15) as

\[
Luu = s.
\]

This can be solved directly for \( u \) in terms of an auxiliary vector \( \chi \) where

\[
Lu = \chi \quad \text{and} \quad Ly = s.
\]

The initial processes yield the algorithm

\[
y_1 = s_1 / \omega_1 ;
\]

\[
y_i = (s_i - \beta_i Y_{i-1}) / \omega_i, \quad \text{for } i=2, 3, \ldots, m-1
\]

and

\[
y_i = (s_i - \beta_i Y_{i-1} - \sum_{k=i-m+1}^{i} \gamma_{k-i+m,i-m+1} Y_{k}) / \omega_i,
\]

for \( i=m, m+1, \ldots, n \).

Then a back substitution process yields \( u \) in terms of \( \chi \), the components being given by the equations:

\[
u_n = \gamma_n
\]

and

\[
u_i = \gamma_i - g_i u_{i+1} - \sum_{j=p}^{q} h_{i-j+m,j-m+1} Y_j,
\]

where \( p, q \) are given by:

If \( m \geq \frac{n+1}{2} \)

\[
p = i+1, \quad q = n \quad \text{for } i=n-1, n-2, \ldots, m \text{ (except if } m=n)\]

\[
p = m, \quad q = n \quad \text{for } i=m-1, m-2, \ldots, n-m+1\]

\[
p = m, \quad q = i+m-1 \quad \text{for } i=n-m, n-m-1, \ldots, 2, 1 \text{ (except if } m=n)\]

If \( m < \frac{n+1}{2} \)

\[
\text{If } n \text{ is even and } m = \frac{n}{2},
\]
then
\begin{align*}
p &= i + 1, \quad q = n \quad \text{for } i = n-1, n-2, \ldots, m+1 \\
p &= m + 1, \quad q = n-1 \quad \text{for } i = m \\
p &= m, \quad q = i + m - 1 \quad \text{for } i = m-1, m-2, \ldots, 2, 1.
\end{align*}

else
\begin{align*}
p &= i + 1, \quad q = n \quad \text{for } i = n-1, n-2, \ldots, n-m+1 \\
p &= i + 1, \quad q = i + m - 1 \quad \text{for } i = n-m, n-m-1, \ldots, m \\
p &= m, \quad q = i + m - 1 \quad \text{for } i = m-1, m-2, \ldots, 2, 1.
\end{align*}

With \([s]\) we denote the greatest integer not exceeding \(s\).

If \(r_i = 0, \quad i = 1, 2, \ldots, n-m+1\) and \(v_i = 0, \quad i = m, m+1, \ldots, n\),

the algorithm (henceforth called the LUBOT algorithm) reduces to

the form of the common tridiagonal system \([57]\).

Storage requirements and computational work

Although the \(\beta, \omega, g\) vector stores can be overwritten the \(\gamma, h\) arrays and \(s\) vector stores have to be strictly preserved. Given

that the memory space required for the \(h\) array is \((m-1)(N-m+1)\) words,

the total memory requirement for the LUBOT algorithm is \(= (2m+4)N\) words.

The amount of work involved is given by:

(i) decompose \(A_{11} = L_{11}U_{11}\) \(\Rightarrow (2m-4)\) operations

(ii) form \(U_{12} = L_{11}^{-1}A_{12}\) \(\Rightarrow (m^2-2m+1)\) operations

(iii) solve for \(L_{21}: \quad L_{21}U_{11} = A_{21}\) \(\Rightarrow (\frac{m^2-3m+2}{2})\) operations

(iv) decompose \(A_{22} = L_{21}U_{12} + L_{22}U_{22}\) \(\Rightarrow (m^2+m)N\) operations

Therefore, the total operations for the factorization stage are

\(= (m^2+m)N\) and given that the forward-backward substitution process

requires \(= (2m+1)N\) operations, the total operations for this algorithm

are \(= (m^2+3m+1)N\), for \(m<<N\).
It is well known [14] that for a symmetric and positive definite matrix $A$ there exists a unique factorization of the form:

$$A = DT'TD$$  \hspace{1cm} (3.3.1)$$

where $D$ is a unique, positive diagonal matrix,

$T$ is a unique, real, upper triangular matrix with unit diagonal elements and $T'$ denotes the transpose of $T$.

With $A$ given by:

$$A = \begin{bmatrix}
a_{11} & b_{11} & c_1 & 0 \\
b_{11} & a_{22} & b_{22} & c_2 & 0 \\
b_{22} & a_{33} & b_{33} & & \ddots \\
& b_{n-2,n-2} & a_{n-1,n-1} & b_{n-1,n-1} & \ddots \\
& & & b_{n,n} & \ddots \\
& & & & & \ddots \\
& & & & & & b_{n-m+1,n-m+1} \\
\end{bmatrix}$$  \hspace{1cm} (3.3.2)$$

$D$ and $T$ in (3.3.1) are of the form:

$$D = \text{diag} \{d_1, d_2, \ldots, d_n\}$$  \hspace{1cm} (3.3.3)$$

$$T = \begin{bmatrix}
e_{11} & n_{12} & 0 & & \cdots \\
1 & e_{22} & t_{22} & 0 & \cdots \\
1 & e_{m-2} & t_{m-2,1} & t_{m-2,2} & \cdots \\
1 & e_{m-1} & t_{m-1,1} & t_{m-1,2} & \cdots \\
& & & & \cdots \\
& & & & & t_{i,n-m+1} \\
& & & & & t_{2,n-m+1} \\
& & & & & t_{m-2,1} \\
& & & & & t_{m-2,2} \\
& & & & & 1 \\
& & & & & 1 \\
& & & & & 1 \\
\end{bmatrix}$$  \hspace{1cm} (3.3.4)$$
The following normalized algorithm [6], due to Benson and Evans, gives the elements of \( D \) and \( T \),

\[
\begin{align*}
d_i &= \sqrt{a_i}; & d_1 &= \left\{ \frac{b_i - 2}{d_{i-1}} \right\}^{1/2}; & e_{i-1} &= \frac{b_i - 1}{d_{i-1} d_i},
\end{align*}
\]

(3.3.5)

Then, for \( i = 2, 3, \ldots, m - 1 \),

\[
\begin{align*}
\text{for } j = 1, 2, 3, \ldots, n - m + 1 & \\
\times_i &= \frac{c_j}{d_j}; & v &= \frac{b_{m+j-2}}{d_{m+j-2}} \tag{3.3.6}
\end{align*}
\]

whilst for \( j \leq m - 2 \),

\[
\begin{align*}
\times_i &= -e_{i+j-2} x_{i-1} \text{, for } i = 2, 3, \ldots, m - j. \tag{3.3.7}
\end{align*}
\]

Then, for all \( j > 1 \),

\[
\begin{align*}
\times_i &= -e_{i+j-2} x_{i-1} - \sum_{k=1}^{i-1} x_{k} t_{i-k+i+m, i+j-m} \tag{3.3.8}
\end{align*}
\]

for either \( i = (m-j+1), (m-j+2), \ldots, (m-1) \) and all \( j \leq m - 1 \) or \( i = 2, 3, \ldots, (m-1) \) for \( j > m - 1 \).

Then,

\[
\begin{align*}
d_{m+j-1} &= \left\{ a_{m+j-1} - \sum_{k=1}^{m-2} x_{k}^2 (x_{m-1} + v)^2 \right\}^{1/2}; & e_{m+j-2} &= \frac{v}{d_{m+j-1}} \tag{3.3.9}
\end{align*}
\]

and

\[
\begin{align*}
t_{i,j} &= \frac{x_i}{d_{m+j-1}}, \quad i = 1, 2, \ldots, m - 1. \tag{3.3.10}
\end{align*}
\]

The linear system (3.2.15) can then be solved by rewriting it in the form

\[
D T' T D u = s \tag{3.3.11}
\]

or equivalently

\[
T' T (D u) = D^{-1} s \tag{3.3.12}
\]

and introducing the auxiliary vectors \( y \) and \( g \) where

\[
y = D u \quad ; \quad g = D^{-1} s \tag{3.3.13}
\]

the problem is reduced to solving

\[
T' T y = g
\]

then \( y \) is obtained in terms of an auxiliary vector \( h \) where

\[
T y = h \quad \text{and} \quad T' h = g \tag{3.3.14}
\]

i.e.,

\[
h_1 = g_1 \quad ; \quad h_i = g_i - e_{i-1} h_{i-1}, \quad i = 2, 3, \ldots, m - 1. \tag{3.3.15}
\]
and
\[ h_i = g_i - e_i h_{i-1} - \sum_{k=i-m+1}^{i-1} t_{k+i+m, i-m+1} h_k, \quad i=m, m+1, \ldots, n \]  

A back substitution process, in a similar manner as in section 3.2 gives the $\chi$ in terms of $h$ and finally the solution $u$ is obtained from $u = D^{-1} \chi$.

The total memory requirements for the NORMBAND algorithm is
\[ \approx (m+3)N \] words.

The amount of work involved in the factorization is
\[ \approx \frac{(m-1)(m-2) + 3m-1)}{2} N \] mults + $N$ square roots.

The normalization and forward-backward substitution processes require respectively $N$ divisions and \[ \approx 2mN \] mults + $N$ divisions.

Therefore, the total number of operations for this algorithm is
\[ \approx \frac{(m-1)(m-2)}{2} + 5m + 1)N \] mults + $N$ square roots, for $m \ll N$.

Note that although the factorization $A = DT'TD$ is preferable for theoretical purposes, it has the disadvantage of involving $N$ square roots, which is relatively costly in computer time, and in this case the factorization $A = LU$ may be computationally more desirable.

3.4 THE ALGORITHM FOR THE SOLUTION OF LARGE, UNSYMMETRIC, SEVEN DIAGONAL, SPARSE LINEAR SYSTEMS (THE LUBOT-3D ALGORITHM)

In this section, by extending the techniques of algorithmic construction developed in section 3.2, we introduce a sparse LU triangular factorization method for the solution of a seven-diagonal unsymmetric, linear system of bandwidth $m$ and $p$ (LUBOT-3D Algorithm) which can be regarded as a natural extension to the LUBOT-2D algorithm.

Such systems are derived from the application of finite difference method to the solution of parabolic p.d.e's in three space dimensions (see section 2.8).
Let $\Omega$ (a large, diagonally dominant, seven-diagonal, sparse matrix) be factorised into the product of the two matrices $\bar{L}, \bar{U}$, i.e.

$$\Omega = \bar{L}\bar{U}$$

(3.4.1)

where $\Omega$ as given in (2.8.15)

- $\bar{L}$ is a strictly lower triangular matrix and
- $\bar{U}$ is a strictly upper triangular matrix with unit diagonal elements.

The coefficient matrix $\Omega$ can be rewritten in the following partitioned form

$$\Omega = \begin{bmatrix} \Omega_{11} & \Omega_{12} \\ \Omega_{21} & \Omega_{22} \end{bmatrix}$$

(3.4.2)
while the triangular matrices $\bar{L}, \bar{U}$ in (3.4.1) are of the following form:
\[ \bar{U} = \begin{bmatrix} \bar{U}_{11} & \bar{U}_{12} \\ 0 & \bar{U}_{22} \end{bmatrix} \]

\[ (3.4.4) \]
Equating the partitioned form of \( L \) and \( U \), which can be obtained similarly as in (3.2.5), with the partitioned form of \( \Omega \) in (3.4.2) we obtain the following identities:

\[
\begin{align*}
\Omega_{11} &= \bar{L}_{11} \bar{U}_{11} ; & \Omega_{12} &= \bar{L}_{11} \bar{U}_{12} ; \\
\Omega_{21} &= \bar{L}_{21} \bar{U}_{11} ; & \Omega_{22} &= \bar{L}_{21} \bar{U}_{12} + \bar{L}_{22} \bar{U}_{22} .
\end{align*}
\]  

(3.4.5a)  

(3.4.5b)  

The elements of \( \bar{L}_{11} \), \( \bar{U}_{11} \) are known and may be obtained by applying the LUBOT-2D algorithm for the solution of general five term linear system of order \((p-1)\) and semibandwidth \(m\) [see relations (3.2.7)-(3.2.14)]. 

The elements of \( \bar{U}_{12}, \bar{L}_{21}, \bar{L}_{22}, \bar{U}_{22} \) may be obtained as follows:

for \( j=1,2, \ldots, n-p+1 \).

\[
t_{1,j} = r_j \omega_j ; \quad f_{1,j} = s_{p+j-1} \quad (3.4.6)
\]

and

\[
g_{p+j-2} = \frac{c_{p+j-2} \omega_{p+j-2}}{p+j-2} ; \quad \beta_{p+j-2} = \frac{a_{p+j-1}}{p+j-1} , \quad (3.4.7)
\]

whilst for \( j \geq m-2 \)

\[
t_{i,j} = \frac{\beta_{1+j-2} t_{1-1,i-1,j}}{\omega_i+j-1} ; \quad f_{i,j} = -g_{i+j-2} f_{1-1,i-1,j} , \quad \text{for } i=2,3,\ldots,m-j \quad (3.4.8)
\]

then, if \( j=1 \) and \( i=p-m+1 \)

\[
t_{i,j} = t_{1,j} + \gamma_{i+j-2} t_{1,j} / \omega_i+j-1 ; \quad f_{i,j} = f_{1,j} + \gamma_{p+j-1} \quad (3.4.9)
\]

If \( i \geq p+j+1 \) or \( j \geq p-2 \), then

\[
t_{i,j} = (-\beta_{1+j-2} t_{1-1,i-1,j} - \sum_{k=1}^{i-1} f_{k-1+i+p,i+j-p} t_{k,j}) / \omega_i+j-1 \quad (3.4.10)
\]

and

\[
f_{i,j} = -g_{i+j-2} f_{1-1,i-1,j} - \sum_{k=1}^{i-1} f_{k,j} t_{k-1+i+p,i+j-p} \quad (3.4.11)
\]

if \( i=p-m+1 \), then

\[
t_{i,j} = t_{i,j} + \gamma_{i+j-2} t_{i,j} / \omega_i+j-1 ; \quad f_{i,j} = f_{i,j} + \gamma_{p+j-1} \quad (3.4.12)
\]

else

if \( i \leq m \), then

\[
t_{i,j} = (-\beta_{1+j-2} t_{1-1,i-1,j} - \sum_{k=1}^{i-1} \gamma_{k-1+i+m,i+j-m} t_{k,j}) / \omega_i+j-1 \quad (3.4.13)
\]

and

\[
f_{i,j} = -g_{i+j-2} f_{1-1,i-1,j} - \sum_{k=1}^{i-1} f_{k,j} h_{k-1+i+m,i+j-m} \quad (3.4.14)
\]
\[ t_{i,j} = (-\beta_i + j - 2t_{i-1,j}) - \sum_{k=1}^{m-1} \gamma_{k,i+j-m} t_{i+k,i+m,j} / \omega_{i+j-1} \] (3.4.15)

and

\[ f_{i,j} = -g_i + j - 2f_{i-1,j} - \sum_{k=1}^{m-1} f_{k+i-m,j} h_k, i+j-m \] (3.4.16)

if \( i = p-m+1 \), then

\[ t_{i,j} + \tau_{i+j-l} / \omega_{i+j-1} \quad f_{i,j} + p+j-1 \] \( \) (3.4.17)

for either \( i = (m-j+1), (m-j+2), \ldots, p-1 \) and all \( j \leq m-1 \)

or

\[ i = 2, 3, \ldots, p-1 \] for \( j > m-1 \).

Then, for \( i = p-1 \)

\[ \omega_{p+j} = \beta_{p+j-1} - \beta_{i+j-l} - t_{i, j} / \omega_{i+j-1} \quad f_{i, j} + p+j-1 \] \( \) (3.4.18)

The standard notation "\( \theta_1 \leftarrow \theta_2 \)" see [37], where \( \theta_1 \) is variable and \( \theta_2 \) is variable or formula, means the value of \( \theta_1 \) is to be replaced by the value of \( \theta_2 \).

The system (2.8.14) can be solved by rewriting as before

\[ \bar{L} \bar{U} = \bar{s} \quad . \] (3.4.19)

Let

\[ \bar{y} = \bar{U} \bar{u} \] (3.4.20)

then the problem is to solve

\[ \bar{L} \bar{y} = \bar{s} \] (3.4.21)

i.e.,

\[ y_1 = s_1 \] (3.4.22)

\[ y_i = (s_i - \beta_i - y_{i-1}) / \omega_{i-1} \quad \text{for } i = 2, 3, \ldots, m-1 \] (3.4.23)

\[ y_i = (s_i - \beta_i - y_{i-1} - \sum_{k=i-m+1}^{i-1} \gamma_{k-i+m, i+m+1} y_k) / \omega_i \quad \text{for } i = m, m+1, \ldots, p-1 \] (3.4.24)

and

\[ y_i = (s_i - \beta_i - y_{i-1} - \sum_{k=i-m+1}^{i-1} \gamma_{k-i+m, i+m+1} y_k - \sum_{k=i-p+1}^{i-1} f_{k-i+p, i-p+1} y_k) / \omega_i \quad \text{for } i = p, p+1, \ldots, n \] (3.4.25)

followed by a back substitution process given by

\[ \bar{U} \bar{u} = \bar{y} \quad . \] (3.4.26)

i.e.,

\[ u_n = y_n \] (3.4.26)
and
\[ u_i = y_i - g_i y_{i+1} - \sum_{k=p}^{q} h_{i-k+m,k-m+1} y_k - \sum_{k=p}^{q} t_{i-k+p,k-p+1} y_k \]

where the quantities \( \tilde{p}, \tilde{q}, \tilde{p}, \tilde{q} \) are given by the following 'shorthand' representation.

If \( m > \lceil n/2 + 1 \rceil \)

if \( p > \lceil n/2 + 1 \rceil \) then

\[ \tilde{p} = i + 1, \; \tilde{q} = n \quad \text{for} \quad i = n-1, n-2, \ldots, p^* \quad \text{for} \quad \tilde{p} = i + 1, \; \tilde{q} = n \]
\[ i = p-1, p-2, \ldots, m^+ \]

\[ \tilde{p} = m, \; \tilde{q} = n \quad \text{for} \quad i = m-1, m-2, \ldots, n-m+1 \quad \text{for} \quad \tilde{p} = m, \; \tilde{q} = n \]

\[ \tilde{p} = m, \; \tilde{q} = i + m - 1 \quad \text{for} \quad i = n-m, n-m-1, \ldots, 1^+ \quad \text{for} \quad \tilde{p} = m, \; \tilde{q} = i + m - 1 \]

else

\[ \tilde{p} = i + 1, \; \tilde{q} = n \quad \text{for} \quad i = n-1, n-2, \ldots, m^+ \]

\[ \tilde{p} = m, \; \tilde{q} = n \quad \text{for} \quad i = m-1, m-2, \ldots, n-m+1 \]

\[ \tilde{p} = m, \; \tilde{q} = i + m - 1 \quad \text{for} \quad i = n-m, n-m-1, \ldots, 1^+ \]

Note:

(+) denotes: "except if \( m = n \)"

(*) denotes: "except if \( p = n \)"

If \( m < \lceil n/2 + 1 \rceil \)

if \( p > \lceil n/2 + 1 \rceil \)

if \( n \) is even and \( m = n/2 \), then

\[ \tilde{p} = i + 1, \; \tilde{q} = n \quad \text{for} \quad i = n-1, n-2, \ldots, p^* \quad \text{for} \quad \tilde{p} = i + 1, \; \tilde{q} = n \]
\[ i = p-1, p-2, \ldots, m^+ \]

\[ \tilde{p} = m+1, \; \tilde{q} = n-1 \quad \text{for} \quad i = m \quad \text{for} \quad \tilde{p} = p, \; \tilde{q} = n \]

\[ \tilde{p} = m, \; \tilde{q} = i + m - 1 \quad \text{for} \quad i = n-m, n-m-1, \ldots, n-p+1 \quad \text{for} \quad \tilde{p} = p, \; \tilde{q} = i + p - 1 \]
\begin{align*}
\text{else} \\
\tilde{p} = i+1, \quad \tilde{q} = n \quad \text{for} \begin{cases} 
\{ i=n-1, n-2, \ldots, p \} & \text{for } p + 1, \tilde{q} = n \\
\{ i=p-1, p-2, \ldots, n-m+1 \} & \text{for } p = p+1, \tilde{q} = n-1 
\end{cases} \\
\tilde{p} = i+1, \tilde{q} = i+m-1 \quad \text{for} \begin{cases} 
\{ i=n-m, n-m+1, \ldots, m \} & \text{for } p = p, \tilde{q} = n \\
\{ i=p-1, p-2, \ldots, m \} & \text{for } p = p, \tilde{q} = i+p-1 
\end{cases} \\
\tilde{p} = m, \tilde{q} = i+m-1 \quad \text{for} \begin{cases} 
\{ i=m-1, m-2, \ldots, n-p+1 \} & \text{for } p = p, \tilde{q} = i+p-1 \\
\{ i=n-p, n-p-1, \ldots, 1 \} & \text{for } p = p, \tilde{q} = i+p-1 
\end{cases}
\end{align*}

If \( p < [n/2+1] \)

\begin{align*}
\text{if } n \text{ is even and } p = n/2, \text{ then} \\
\tilde{p} = i+1, \tilde{q} = n \quad \text{for} \begin{cases} 
\{ i=n-1, n-2, \ldots, p+1 \} & \text{for } p = 1, \tilde{q} = n \\
\{ i=p, p-1, p-2, \ldots, m+1 \} & \text{for } p = p+1, \tilde{q} = n-1 
\end{cases} \\
\tilde{p} = m+1, \tilde{q} = n-1 \quad \text{for} \begin{cases} 
\{ i=m \} & \text{for } p = p, \tilde{q} = i+p-1 
\end{cases} \\
\tilde{p} = m, \tilde{q} = i+m-1 \quad \text{for} \begin{cases} 
\{ i=m-1, m-2, \ldots, 1 \} & \text{for } p = p, \tilde{q} = i+p-1 
\end{cases}
\end{align*}

\text{else} \\
\tilde{p} = i+1, \tilde{q} = n \quad \text{for} \begin{cases} 
\{ i=n-1, n-2, \ldots, n-m+1 \} & \text{for } p = p, \tilde{q} = n \\
\{ i=n-m, n-m-1, \ldots, n-p+1 \} & \text{for } p = i+1, \tilde{q} = i+p-1 
\end{cases} \\
\tilde{p} = i+1, \tilde{q} = i+m-1 \quad \text{for} \begin{cases} 
\{ i=n-p, n-p-1, \ldots, p \} & \text{for } p = p, \tilde{q} = i+p-1 \\
\{ i=p-1, p-2, \ldots, m \} & \text{for } p = p, \tilde{q} = i+p-1 
\end{cases} \\
\tilde{p} = m, \tilde{q} = i+m-1 \quad \text{for} \begin{cases} 
\{ i=m-1, m-2, \ldots, 1 \} & \text{for } p = p, \tilde{q} = i+p-1 
\end{cases}
\end{align*}

\text{if } p > n \text{ or } p < m

\begin{align*}
\text{if } n \text{ is even and } m = n/2, \text{ then} \\
\tilde{p} = i+1, \tilde{q} = n \quad \text{for } i=n-1, n-2, \ldots, m+1 \\
\tilde{p} = m+1, \tilde{q} = n-1 \quad \text{for } i=m \\
\tilde{p} = m, \tilde{q} = i+m-1 \quad \text{for } i=m-1, m-2, \ldots, 1. \\
\text{else} \\
\tilde{p} = i+1, \tilde{q} = n \quad \text{for } i=n-1, n-2, \ldots, n-m+1 \\
\tilde{p} = i+1, \tilde{q} = i+m-1 \quad \text{for } i=n-m, n-m-1, \ldots, m \\
\tilde{p} = m, \tilde{q} = i+m-1 \quad \text{for } i=m-1, m-2, \ldots, 1.
\end{align*}
The total memory requirement for the LUBOT-3D algorithm is 
\[(2m+2p+4)N\text{ words.}\]
The amount of work involved in the factorization stage is \(\approx(p^2+p+3)N\) operations and given that the forward-backward substitution process requires \(\approx(2m+2p-1)N\) operations, the total operations for this algorithm are \(\approx(p^2+3p+2m+2)N\), for \(m<p<N/2\).

3.5 THE NORMALIZED ALGORITHM FOR THE SOLUTION OF LARGE, SYMMETRIC, SEVEN DIAGONAL, SPARSE LINEAR SYSTEMS (THE NB3D ALGORITHM)

In this section we introduce a sparse normalized factorization method for the solution of the seven-diagonal, symmetric, linear systems of bandwidth \(m\) and \(p\) (NB3D Algorithm), resulting from the application of the finite difference method to the solution of self adjoint p.d.e's in three space dimensions (see section 2.1).

We consider, as in section 3.3, the unique factorization:
\[A = \tilde{D}\tilde{T}'\tilde{T} \tag{3.5.1}\]
where \(A\) is a large symmetric, positive definite, seven-diagonal sparse matrix, \(\tilde{D}\) is a unique positive diagonal matrix, \(\tilde{T}\) is a unique real upper triangular matrix with unit diagonal elements and \(\tilde{T}'\) denotes the transpose of \(\tilde{T}\). Then, the coefficient matrix \(A\) of the system (2.1.3) can be written in the following partitioned form:
while $\tilde{d}, \tilde{r}$ in (3.5.1) are of the following form:

$$
\tilde{d} = \begin{bmatrix}
  d_1 \\
  d_2 \\
  \vdots \\
  d_p \\
  0 \\
  \end{bmatrix},
$$

(3.5.3)
(3.5.4)
Replacing the right hand side of (3.5.1) by its partitioned forms, we obtain,

\[
\begin{bmatrix}
\tilde{D} & 0 \\
0 & \tilde{d}
\end{bmatrix}
\begin{bmatrix}
\tilde{T}' & 0 \\
r' & u'
\end{bmatrix}
\begin{bmatrix}
\tilde{T} & r \\
0 & u
\end{bmatrix}
\begin{bmatrix}
\tilde{D} & 0 \\
0 & \tilde{d}
\end{bmatrix} =
\begin{bmatrix}
\tilde{D}'\tilde{T}\tilde{D} & \tilde{D}'\tilde{r}\tilde{d} \\
\tilde{d}'\tilde{T}\tilde{D} & \tilde{d}'\tilde{r}\tilde{d} + \tilde{u}'\tilde{u}\tilde{d}
\end{bmatrix}
\]

(3.5.5)

where \(\tilde{T}', r', u'\) are the transpose of \(\tilde{T}, r, u\) respectively.
and on equating (3.5.5) with the partitioned form of A in (3.5.2) we have the identities:

\[
\tilde{A} = \tilde{D} \tilde{T}' \tilde{T} \tilde{D}; \quad \tilde{b} = \tilde{D} \tilde{T}' \tilde{r} \tilde{d}
\]

\[
\tilde{a} = \tilde{d} \tilde{r}' \tilde{r} + \tilde{u} \tilde{u}' \tilde{u}
\]

(3.5.6)  (3.5.7)

The elements of \( \tilde{D}, \tilde{T} \) are known and may be obtained by applying the normalized algorithm for the solution of symmetric, general five term linear systems of order \((p-1)\) and semibandwidth \(m\) (see section 3.3).

In order to define the elements of the submatrices \( r, u, d \) from the identities (3.5.6), (3.5.7), a test problem, with a simplified structure of the coefficient matrix \( A \), has been worked out in Appendix 1.

The equations to determine the elements of \( \tilde{T} \) matrix prove to be non-linear and a simple iterative Picard-type scheme was used in an inner loop to determine the values of \( d_p, d_{p+1}, \ldots, d_n \) in an easy manner as the direct solution of these equations proved to be intractable. The elements of the submatrices \( r, u, d \) may be obtained as follows:

For \( j=1,2,\ldots,n-p+1 \)

\[
d_{p+j-1} = d_{p+j-2} \quad \text{ (Initial guess of } d_{p+j-1} \text{ is taken as the adjacent value) }
\]

(3.5.8)

\[
r_{1,j} = h_j / d_{j} \quad \text{ (3.5.9)}
\]

and

\[
r_{p+j-1,1} = b_{p+j-2}/d_{p+j-2} \quad \text{ (3.5.9)}
\]

whilst for \( j=m-2 \)

\[
r_{i,j} = -e_{i+j-2}r_{i-1,j} \quad \text{ for } i=2,3,\ldots,m-j 
\]

(3.5.10)

then, if \( j=1 \) and \( i=p-m+1 \)

\[
r_{i,j} = \sum_{k=1}^{i-1} r_{k-i+p,i+j-p} r_{k,j}
\]

(3.5.11)

If \( j>p-2 \) or \( i>p-j+1 \), then

\[
r_{i,j} = -e_{i+j-2}r_{i-1,j} - \sum_{k=1}^{i-1} r_{k-i+p,i+j-p} r_{k,j}
\]

(3.5.12)
then, if \( i = p - m + 1 \)

\[
{r}_{i,j} = {r}_{i,j} + \frac{c_{i+j-1}}{d_{i+j-1}} + \frac{d_{p+j-1}}{d_{i+j-1}} \tag{3.5.13}
\]

else

if \( i < m \), then

\[
{r}_{i,j} = -e_{i+j-2}{r}_{i-1,j} - \sum_{k=1}^{i-1} r_{k,j} t_{k-i+m,i+j-m} \tag{3.5.14}
\]

else

\[
{r}_{i,j} = -e_{i+j-2}{r}_{i-1,j} - \sum_{k=1}^{m-1} r_{k+i-m,j} t_{k,i+j-m} \tag{3.5.15}
\]

if \( i = p - m + 1 \), then

\[
{r}_{i,j} = -e_{i+j-2}{r}_{i-1,j} + \frac{c_{i+j-1}}{d_{i+j-1}} \tag{3.5.16}
\]

for either \( i = 2, 3, \ldots, p - 1 \) and all \( j > m - 1 \) or

\[
i = \left( m - j + 1 \right), \left( m - j + 2 \right), \ldots, p - 1\text{ for } j \leq m - 1.
\]

Then, for \( i = p - 1 \)

\[
d_{p+j-1} = a_{p+j-1} / \left[ 1 + \sum_{k=1}^{p+j-2} r_{k,j} + \left( e_{p+j-2} + r_{i,j} \right)^{2} \right]^{\frac{1}{2}} \tag{3.5.17}
\]

The linear system (2.1.3) can be written as:

\[
(\bar{D}\bar{T}')\bar{T}u = \bar{s} \tag{3.5.18}
\]

from which we obtain

\[
(T'T)Du = D^{-1}s \tag{3.5.19}
\]

Let

\[
y = Du \quad \text{and} \quad g = D^{-1}s \tag{3.5.20}
\]

then the problem is reduced to solving the system

\[
(\bar{T}'\bar{T})y = g \tag{3.5.21}
\]

This can be solved directly for \( y \) in terms of the auxiliary vector \( h^* \), where

\[
\bar{\bar{T}}y = h^* \quad \text{and} \quad \bar{T}'h^* = g \tag{3.5.22}
\]

i.e.

\[
h^*_1 = g_1 \tag{3.5.23}
\]

\[
h^*_i = g_i - e_{i-1}h^*_{i-1}, \quad i = 2, 3, \ldots, m - 1 \tag{3.5.24}
\]

\[
h^*_i = g_i - e_{i-1}h^*_{i-1} - \sum_{k=i-m+1}^{i-1} t_{k-i+m,i+m+1}h^*_k, \quad i = m, m+1, \ldots, p-1 \tag{3.5.25}
\]

and

\[
h^*_i = g_i - e_{i-1}h^*_{i-1} - \sum_{k=i-m+1}^{i-1} t_{k-i+m,i+m+1}h^*_k - \sum_{k=i-p+1}^{i-1} r_{k-i+p,i+p}h^*_k, \quad i = p, p+1, \ldots, n \tag{3.5.26}
\]
A back substitution process yields $y$ in terms of $h^*$, the components being given by the equations

$$y_n = h^*_n \quad (3.5.27)$$

and

$$y_i = h^*_i - e y_{i+1} - \sum_{j=p}^{i-j+m, j-m+1} r_{i-j+p} y_j \quad (3.5.28)$$

where the values of $\tilde{p}, \tilde{q}, \tilde{p}, \tilde{q}$ are given in section 3.4.

The final solution $u$ is obtained from $u = b^{-1}y$ an operation involving only one division per component.

The total memory requirement of the NB3D algorithm is $\approx (m+p+3)N$ words. The amount of work involved for the factorization is

$$\frac{p(p-1)}{2} + 2p + 4)N \text{ mults} + N \text{ square roots.}$$

Given that the normalization and the forward-backward substitution processes require $2N$ divisions and $(2m+2p-2)N$ mults respectively, the total number of operations for this algorithm is:

$$\approx (\frac{p(p-1)(p-2)}{2} + 4p + 2m+4)N \text{ mults} + N \text{ square roots.}$$

Note that if $h_i=0, i \in [1, n-p+1]$ or $c_i=0, i \in [1, n-m+1]$ the algorithm reduces to the normalized form of the quindiaogonal system of bandwidth $m$ or $p$ respectively, [6], which is encountered usually in solving five point boundary value problems. Furthermore, if $h_i=0, i \in [1, n-p+1]$ and $c_i=0, i \in [1, n-m+1]$ the algorithm reduces to the normalized form of the common tridiagonal system [14], which is encountered in solving two point boundary value problems.
Chapter 4

Approximate Algorithms for Two and Three Dimensional P.D.E's
4.1 INTRODUCTION

For the solution of the class of problems discussed in Chapter 2, it can be easily seen that the storage requirements and computational work for the basic algorithms, as introduced in Chapter 3, is prohibitively high for computers with relatively limited core memory. In this Chapter we introduce approximate algorithmic solution methods in which the large, sparse matrix derived from the finite difference discretizations of parabolic and elliptic p.d.e's, in both two and three space dimensions, is approximately factorized to yield algorithmic procedures for use in iterative schemes for finite difference methods.

These procedures can be considered to be approximate counterparts of the algorithmic procedures given in Chapter 3.

The idea of the approximate factorization method was first proposed by Buleev [9] and Oliphant [47] and is based on the simple replacement of the coefficient matrix \( A \) by a matrix \( (A+B) \) such that

\[
A + B = L_s U_s
\]

where \( L_s \) and \( U_s \) are sparse strictly lower triangular and upper triangular matrices. Obviously there is a large number of such matrices \( B \), where the matrix \( (A+B) \) can be factored in sparse triangular matrices.

In the following, we shall attempt to outline a strategy whereby \( L_s \) and \( U_s \) are easily determined.

4.2 THE APPROXIMATE ALGORITHM FOR THE SOLUTION OF LARGE, UNSYMMETRIC, QUINDIAGONAL, SPARSE LINEAR SYSTEMS (THE ALUBOT-2D ALGORITHM)

In this section, we present an approximate triangular factorization of the coefficient matrix of system (3.2.15), resulting from the application of finite difference methods to the solution of parabolic p.d.e's in two dimensions (see section 2.5).
Let $A$ be defined as in section 3.2, then we consider the approximate factorization

$$A \approx L_s U_s$$

where the triangular matrices $L_s, U_s$ (sparse forms of $L, U$ given by (3.2.3), (3.2.4) respectively) are of the following form:

\[ L_s \equiv \begin{bmatrix} \tilde{\gamma}_1 & \tilde{\gamma}_2 & \cdots & \tilde{\gamma}_{n-m+1} & 0 \\ \tilde{\gamma}_2 & \tilde{\gamma}_3 & \cdots & \tilde{\gamma}_{n-m+1} & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \tilde{\gamma}_{n-m+1} & \tilde{\gamma}_{n-m+2} & \cdots & \tilde{\gamma}_{n-1} & 0 \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix}, \quad (4.2.1a)

\[ U_s \equiv \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ \tilde{g}_1 & 1 & \cdots & 0 & 0 \\ \tilde{g}_2 & \tilde{g}_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \tilde{g}_{m-2} & \tilde{g}_{m-1} & \cdots & 1 & 0 \\ \tilde{g}_{m-1} & \tilde{g}_m & \cdots & \tilde{g}_{m+1} & 1 \end{bmatrix}, \quad (4.2.2a)

\[ \begin{bmatrix} \tilde{h}_{1,1} & \tilde{h}_{1,2} & \cdots & \tilde{h}_{1,n-m+1} & 0 \\ \tilde{h}_{2,1} & \tilde{h}_{2,2} & \cdots & \tilde{h}_{2,n-m+1} & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \tilde{h}_{r,1} & \tilde{h}_{r,2} & \cdots & \tilde{h}_{r,n-m+1} & 0 \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix}, \quad (4.2.2b)\]
The approximate algorithm (henceforth called the ALUBOT Algorithm) which retains the \( r \) outermost off-diagonal entries, can be given in a similar compact form as the LUBOT algorithm and is expressed as follows:

\[
\tilde{\omega}_1 = b_1 ; \quad \tilde{\beta}_1 = a_2 ; \quad \tilde{\gamma}_1 = c_1/\tilde{\omega}_1 , \tag{4.2.3}
\]

for \( i=2,3,\ldots,m-2 \)

\[
\tilde{\omega}_i = b_i - \tilde{\beta}_{i-1} \tilde{\gamma}_{i-1} ; \quad \tilde{\beta}_i = a_{i+1} / \tilde{\omega}_i ; \quad \tilde{\gamma}_i = c_i / \tilde{\omega}_i , \tag{4.2.4}
\]

and

\[
\tilde{\omega}_{m-1} = b_{m-1} - \tilde{\beta}_{m-2} \tilde{\gamma}_{m-2} . \tag{4.2.5}
\]

For \( j=1,2,\ldots,n-m+1 \), we have,

\[
\tilde{h}_{1,j} = \tau_{j}/\tilde{\omega}_j ; \quad \tilde{v}_{1,j} = \nu_{j+m-1} / \tilde{\omega}_j , \tag{4.2.6}
\]

and

\[
\tilde{g}_{m+j-2} = c_{m+j-2} / \tilde{\omega}_{m+j-2} ; \quad \tilde{\beta}_{m+j-2} = a_{m+j-1} , \tag{4.2.7}
\]

whilst for \( j<r-1 \)

\[
\tilde{h}_{i,j} = -\tilde{g}_{i+j-2} \tilde{h}_{i-1,j} / \tilde{\omega}_{i+j-1} , \quad \text{for } i=2,3,\ldots,r-j+1 \tag{4.2.8}
\]

and

\[
\tilde{\gamma}_{i,j} = -\tilde{g}_{i+j-2} \tilde{\gamma}_{i-1,j} , \quad \text{for } i=2,3,\ldots,r-j+1 . \tag{4.2.9}
\]

Then, for \( j>1 \) and \( r>1 \)

\[
\tilde{v}_{i,j} = -\tilde{g}_{i+j-2} \tilde{v}_{i-1,j} - \sum_{k=1}^{i-1} \tilde{v}_{k,j} \tilde{h}_{k-i+r+1,i+j-r-1} / \tilde{\omega}_{i+j-1} , \tag{4.2.10}
\]

and

\[
\tilde{h}_{i,j} = (-\tilde{g}_{i+j-2} \tilde{h}_{i-1,j} - \sum_{k=1}^{i-1} \tilde{h}_{k-i+r+1,i+j-r-1} \tilde{\gamma}_{k,j} / \tilde{\omega}_{i+j-1} \tilde{h}_{k,j} , \tag{4.2.11}
\]

for either \( i=(r-j+2),(r-j+3),\ldots,r \) and all \( j<r \)
or \( i=2,3,\ldots,r \) for \( j>r \).

Then, for \( i=r \)

\[
\tilde{\omega}_{m+j-1} = b_{m+j-1} - \tilde{\beta}_{i+1} \tilde{\gamma}_{i-1,j} - \tilde{\gamma}_{i,j} - \tilde{\beta}_{i+j-1} \tilde{\gamma}_{i-1,j} - \tilde{\beta}_{i+j-1} \tilde{\gamma}_{i-1,j} - \sum_{k=1}^{i} \tilde{h}_{k,j} \tilde{h}_{k,j} , \tag{4.2.12}
\]

An approximate solution of linear system (3.2.15) is now given in a similar manner to (3.2.16)-(3.2.22). The forward substitution process is expressed as

\[
\tilde{\gamma}_1 = s_1 / \tilde{\omega}_1 ; \quad \tilde{\gamma}_i = (s_i - \tilde{\gamma}_{i-1} \tilde{\gamma}_{i-1} / \tilde{\omega}_i) / \tilde{\omega}_i , \quad i=2,3,\ldots,m-1 . \tag{4.2.13}
\]
and

\[ \tilde{y}_i = \left( s_i - \beta_i \tilde{y}_{i-1} \right) \tilde{y}_{i-1} - \sum_{k=i-m+1}^{i-m+r} \tilde{y}_{k-1+i+m, i-m+1} \tilde{y}_k \div \tilde{w}_i, \quad i=m, m+1, \ldots, n \]  

(4.2.14)

Whilst the backward substitution process yields the solution \( \tilde{u} \) in terms of \( \tilde{y} \) as

\[ \tilde{u}_n = \tilde{y}_n \]  

(4.2.15)

and

\[ \tilde{u}_i = \tilde{y}_i - \tilde{g}_i \tilde{u}_{i+1} - \sum_{j=p}^{q} \tilde{h}_{i-j+m, j-m+1} \tilde{y}_j, \]  

(4.2.16)

where the quantities \( p^*, q^* \) can be easily obtained from those given in section 3.2.

The total memory requirement for the ALUBOT algorithm is \((2r+6)N\) words.

The amount of work involved for the factorization is \((r^2+3r+2)N\) operations.

Given that the forward-backward substitution process requires \((2r+3)N\) operations, the total number of operations for this algorithm is \((r^2+5r+5)N\).

### 4.3 THE APPROXIMATE NORMALIZED ALGORITHM FOR THE SOLUTION OF LARGE, SYMMETRIC, QUINDIAGONAL, SPARSE LINEAR SYSTEMS (THE NOBAR ALGORITHM)

An approximate triangular factorization of the coefficient matrix of system (3.2.15), resulting from the application of the finite difference method to the solution of self adjoint p.d.e's in two space dimensions (see section 2.1) is now introduced.

Let \( A \) be defined as in section 3.3 and we consider the approximate factorization

\[ A \approx D_s T_s' T_s D_s \]  

(4.3.1)

where \( D_s, T_s \) are the sparse forms of \( D, T \) matrices in section 3.3 and \( T_s' \) denotes the transpose of \( T_s \).

With the coefficient matrix \( A \) as in (3.3.2) \( D_s, T_s \) have the
The approximate normalized algorithm (henceforth called NOBAR) obtained from retaining $r$-outermost off-diagonal entries can be expressed in the following compact form:
Then for \( j=1,2,3,\ldots,n-m+1 \), we have

\[
\tilde{x}_1 = \frac{c_j}{\tilde{d}_j} \quad ; \quad \tilde{v} = \frac{b_{m+j-2}}{\tilde{d}_{m+j-2}}
\]  
\[ (4.3.5) \]

and for \( j<r-1 \)

\[
\tilde{x}_i = \tilde{\varepsilon}_{i+j-2} \tilde{x}_{i-1}, \quad \text{for} \ i=2,3,\ldots,r+1-j.
\]  
\[ (4.3.6) \]

Then, for \( j>1 \) and \( r>1 \)

\[
\tilde{x}_i = \tilde{\varepsilon}_{i+j-2} \tilde{x}_{i-1} - \sum_{k=1}^{i-1} \tilde{x}_k \tilde{\varepsilon}_{k-i+r+1,i+j-r-1},
\]  
\[ (4.3.7) \]

and for \( i=(r-j+2),(r-j+3),\ldots,r \) and all \( j<r \) or

\[ i=2,3,\ldots,r \quad \text{for} \ j>r. \]

Then, we have

\[
\tilde{d}_{m+j-1} = \left\{ a_{m+j-1} \sum_{k=1}^{r} \tilde{x}_k \tilde{v}_2 \right\}^{\frac{1}{2}}
\]  
\[ (4.3.8) \]

\[
\tilde{c}_{m+j-2} = \tilde{v}/\tilde{d}_{m+j-1},
\]  
\[ (4.3.9) \]

and,

\[
\tilde{t}_{i,j} = \tilde{x}_i/\tilde{d}_{m+j-1}, \quad \text{for} \ i=1,2,3,\ldots,r.
\]  
\[ (4.3.10) \]

An approximate solution of the linear system (3.2.15) can then be obtained by writing

\[
D \cdot T'T \cdot D \cdot u = s,
\]  
\[ (4.3.11) \]

or equivalently

\[
(T'T \cdot D \cdot u) = D^{-1} \cdot s.
\]  
\[ (4.3.12) \]

By introducing the auxiliary vectors \( \chi \) and \( g \) where,

\[
\chi = D \cdot u; \quad g = D^{-1} \cdot s
\]  
\[ (4.3.13) \]

the problem is reduced to solving the normalized system

\[
T'_s \cdot T_s \cdot \chi = g
\]  
\[ (4.3.14) \]

and \( \chi \) is obtained in terms of an auxiliary vector \( h \), where

\[
T_s \cdot \chi = h \quad \text{and} \quad T'_s \cdot h = g
\]  
\[ (4.3.15) \]
i.e. \[ h_1 = g_1 \] ;  
\[ h_i = g_i - \sum_{j=1}^{i-1} h_{i-1}^{-1} h_{i-1} , \quad i=2,3,\ldots,m-1 ; \]  
\[ h_i = g_i - \sum_{j=1}^{i-1} h_{i-1}^{-1} h_{i-1} - \sum_{k=i-m+1}^{i-m+r} \sum_{i=m}^{n} t_{i+j} h_k , \quad i=m,m+1,\ldots,n. \]  

(4.3.16)
(4.3.17)
(4.3.18)

A back substitution process yields \( y \) in terms of \( h \), the components being given by the equations

\[ y_n = h_n \]  
\[ y_i = h_i - \sum_{j=1}^{i-1} y_{i-1} - \sum_{j=p}^{q} \sum_{i=m}^{n} t_{i-j} y_j \]  

(4.3.19)
(4.3.20)

where \( p,q \) are easily obtained from those given in section 3.2.

The final solution \( u \) is obtained from (4.3.13) as \( u = D_s^{-1} y \), an operation involving only one division per vector component.

The total memory requirements for this algorithm is \( (r+4)N \) words.

The amount of work involved for the factorization process is given by \( \frac{r(r-1)}{2} + 3r + 6 \) mults + \( N \) square roots.

The normalization and the forward-backward substitution processes require \( 2N \) divisions and \( (2r+2)N \) multiplications respectively. Therefore, the total number of operations for the NOBAR algorithm is

\[ \frac{r(r-1)}{2} + 5r + 7 \] mults + \( N \) square roots  

(4.3.21)

Remark

It can be easily seen from (4.3.3), (3.3.4) and (4.3.4), (3.3.5) that

\[ \tilde{e}_i = e_i, \quad \text{for } i=1,2,\ldots,m-2 \]  
\[ \tilde{d}_i = d_i, \quad \text{for } i=1,2,\ldots,m-1 . \]  

(4.3.22)
(4.3.23)

By calculating exactly (as in (3.3.9)) the \( \tilde{d}_i, i \in [m,n] \) and \( \tilde{e}_i, i \in [m-1,n-1] \) from relations (4.3.8) and (4.3.9) respectively i.e., \( \tilde{d}_i = d_i, i \in [m,n] \) and \( \tilde{e}_i = e_i, i \in [m-1,n-1] \), the following approximate factorization is obtained:
where \( D \) is given by (3.3.3) and (3.3.5), (3.3.9)

\[ T_s \]

is the sparse version of (3.3.4) (r-

outermost off diagonal terms are retained),

and

\[ T'_s \]

is the transpose of \( T_s \).

Although the approximate factorization (4.3.24) is preferable

for theoretical purposes and analysis, in practice it turned out to be

computationally more efficient to use the factorization of the

form (4.3.1).

4.4 THE APPROXIMATE ALGORITHM FOR THE SOLUTION OF LARGE, UNSYMMETRIC

SEVEN DIAGONAL, SPARSE LINEAR SYSTEMS (THE ALUBOT-3D ALGORITHM)

We now introduce an approximate triangular factorization of the

coefficient matrix of system (2.8.14) resulting from the application

of finite difference methods to the solution of parabolic p.d.e's

in three space dimensions (see section 2.8).

Let \( \Omega \) be defined as in section 3.4 and consider the approximate

factorization

\[ \Omega \approx \bar{L}_s \bar{U}_s \]

(4.4.1)

where the triangular matrices \( \bar{L}_s, \bar{U}_s \) (sparse forms of \( L, U \) given

by (3.4.3), (3.4.4) respectively) are of the following form:
\[
\begin{align*}
\mathbf{U}_s &= \begin{bmatrix} g_1 & h_{1,1} & h_{1,2} & \cdots & h_{1,p-m} & \cdots & h_{1,n-m+1} \\
1 & g_2 & h_{r_1,1} & h_{r_1,2} & \cdots & h_{r_1,p-m} & \cdots & h_{r_1,n-m+1} \\
1 & g_3 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
\vdots & \vdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
1 & g_{m-1} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
1 & g_m & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
1 & g_{p-2} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
1 & g_{p-1} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
1 & g_{n-1} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
\end{bmatrix} \\
&t_{1,1} & t_{1,2} & \cdots & t_{1,n-p+1} \\
0 & t_{2,1} & t_{2,2} & \cdots & t_{2,n-p+1} \\
0 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
0 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
0 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
0 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
0 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
0 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
0 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
0 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
\end{align*}
\]

(4.4.3)
It can be easily seen that the elements of $\tilde{U}_{11}, \tilde{U}_{11}$ are similar in structure to the factors in ALUBOT-2D and are known. These may be obtained by applying the ALUBOT-2D algorithm for the solution of a general five term linear system of order $(p-1)$, semibandwidth $m$, where $r_1$ outermost off-diagonal entries are retained.

The elements of the submatrices $\tilde{U}_{12}, \tilde{U}_{21}, \tilde{U}_{22}$ (cf. (4.4.2), (4.4.3)) may be obtained algorithmically as follows:

For $j=1, 2, \ldots, n-p+1$
\begin{align*}
t_{1,j} &= t_{j} / \omega_j \quad ; \quad f_{1,j} = s_{p+j-1} \\
\text{and} \\
g_{p+j-2} &= c_{p+j-2} / \omega_{p+j-2} \quad ; \quad \beta_{p+j-2} = a_{p+j-1} ,
\end{align*}

whilst if $j \neq r_2^{-1}$
\begin{align*}
t_{i,j} &= -\beta_{i+j-2} t_{i-1,j} / \omega_{i+j-1}, \text{ for } i=2, 3, \ldots, (r_2+1-j) \\
\text{and} \\
f_{i,j} &= -g_{i+j-2} t_{i-1,j} , \text{ for } i=2, 3, \ldots, (r_2+1-j)
\end{align*}

then, if $j=1$ and $i=p-m+1$
\begin{align*}
t_{1,j} &= t_{1,j} + t_{i+j-1} / \omega_{i+j-1} \quad ; \quad f_{1,j} = f_{1,j} + v_{p+j-1} .
\end{align*}

If $j > p-2$ or $i > p-j+1$, then
\begin{align*}
t_{i,j} &= (-\beta_{i+j-2} t_{i-1,j} - \sum_{k=1}^{i-1} f_{k-i+r_2+1,j} t_{k,j}) / \omega_{i+j-1} \\
\text{and} \\
f_{i,j} &= -g_{i+j-2} t_{i-1,j} - \sum_{k=1}^{i-1} f_{k,j} t_{k-i+r_2+1,j} , \text{ for } i=p-m+1 , \text{ then} \\
t_{i,j} &= t_{i,j} + t_{i+j-1} / \omega_{i+j-1} \quad ; \quad f_{i,j} = f_{i,j} + v_{p+j-1}
\end{align*}

else
\begin{align*}
t_{i,j} &= (-\beta_{i+j-2} t_{i-1,j} - \sum_{k=1}^{i-1} \gamma_{k-i+r_1+1,j} t_{k,j}) / \omega_{i+j-1} \\
\text{and} \\
f_{i,j} &= -g_{i+j-2} t_{i-1,j} - \sum_{k=1}^{i-1} f_{k,j} h_{k-i+r_1+1,j} , \text{ for } i<m , \text{ then}
\end{align*}
else
\[ t_{i,j} = \left( -\beta_{i+j-2} t_{i-1,j} - \frac{1}{2} \sum_{k=1}^{m-1} \gamma_{k,i+j-m} t_{k+i-m,j} \right) / \omega_{i+j-1}, \quad (4.4.1) \]

and
\[ f_{i,j} = -g_{i+j-2} f_{i-1,j} - \frac{1}{2} \sum_{k=1}^{m-1} f_{k+i-m,j} h_{k,i+j-m} \quad (4.4.15) \]

if \( i = p-m+1 \), then
\[ t_{i,j} = t_{i,j} / \omega_{i+j-1}; \quad f_{i,j} = f_{i,j} / \omega_{i+j-1} \quad (4.4.16) \]

for either \( i = 2, 3, \ldots, r_2 \) and all \( j > r_1 \)
or
\[ i = (r_1-j+2), (r_1-j+3), \ldots, r_2 \] for \( j \leq r_1 \).

Then, for \( i = r_2 \)
\[ \omega_{p+j-1} = b_{p+j-1} - \beta_{i+j-1} t_{i,j} - \frac{1}{2} \sum_{k=1}^{r_1} f_{k+j-1} t_{k,i} - \frac{1}{2} \sum_{k=1}^{r_1} f_{k+i-m} h_{k,i+j-m} \quad (4.4.17) \]

An approximate solution of the linear system (2.8.14) can then be obtained by writing,
\[ L_s \tilde{U} s u = s \quad (4.4.18) \]
and if we let
\[ y = \tilde{U} u \quad (4.4.19) \]

the problem is reduced to solving the triangular system
\[ L_s y = s \quad (4.4.20) \]
i.e.,
\[
\begin{align*}
y_1 &= s_1 ; \\
y_i &= \left( s_i - \beta_{i-1} y_{i-1} \right) / \omega_i, \quad \text{for } i = 2, 3, \ldots, m-1 ; \\
y_i &= \left( s_i - \beta_{i-1} y_{i-1} - \frac{1}{2} \sum_{k=1}^{i-m+1} \gamma_{k+i+m,i-m} y_k \right) / \omega_i, \\
& \quad \text{for } i = m, m+1, \ldots, p-1. \\
y_i &= \left( s_i - \beta_{i-1} y_{i-1} - \frac{1}{2} \sum_{k=1}^{i-m+1} \gamma_{k+i+m,i-m} y_k - \frac{1}{2} \sum_{k=1}^{i-p+1} f_{k-i+p,i-p} y_k \right) / \omega_i, \\
& \quad \text{for } i = p, p+1, \ldots, n. \\
\end{align*}
\]

The final solution is obtained from a back substitution process given by
\[ u_n = y_n \quad (4.4.25) \]
and

\[ u_i = y_i - q_i y_{i+1} + \sum_{k=p}^{q} h_{i-k+m} y_{i+m} + l_p y_{i+p} + l_q y_{i+q} \]  

(4.4.26)

The quantities \( p, q, q, p, q \) can be easily obtained from those given in section 3.4.

The above approximate algorithmic procedure (henceforth called the ALUBOT-3D algorithm) in which \( r_1, r_2 \) outermost off-diagonal entries are retained, has been expressed in a compact form, similar to those given in sections 3.4 and 3.5 respectively.

The total memory requirement for the ALUBOT-3D algorithm is

\[ = (2r_1 + 2r_2 + 8)N \]  

words.

The amount of work involved for the factorization process is

\[ = (r_1^2 + r_2^2 + 4r_1 + 4r_2 + 3)N \]  

mults,

and the forward-backward substitution process requires

\[ = (2r_1 + 2r_2 + 3)N \]  

mults.

Therefore, the total operations for this algorithm are

\[ = (r_1^2 + r_2^2 + 6r_1 + 6r_2 + 6)N \]  

mults.

4.5 THE APPROXIMATE NORMALIZED ALGORITHM FOR THE SOLUTION OF LARGE, SYMMETRIC, SEVEN DIAGONAL, SPARSE LINEAR SYSTEMS (THE NO BAR-3D ALGORITHM)

Finally, an approximate triangular factorization of the coefficient matrix of system (2.1.3), resulting from the application of finite difference method to the solution of self adjoint p.d.e's in three space dimensions (see section 2.1) is presented.

We consider the approximate factorization

\[ A \approx D \begin{bmatrix} T & \nabla \\ T & 0 \end{bmatrix} D \]  

(4.5.1)

where the coefficient matrix \( A \) of system (2.1.3) is given by

(3.5.2) and the matrices \( D, T \) (sparse forms of \( D, T \) given by (3.5.3), (3.5.4) respectively), are of the following form:
\( \overline{T}_s' \) denotes the transpose of \( \overline{T}_s \).
Then the approximate normalized algorithmic procedure (henceforth called the NOBAR-3D algorithm), retaining $r_1$ and $r_2$ outermost off-diagonal entries, is developed as follows:

the elements of submatrices $\delta_1$ and $\delta_2$ (cf. (4.5.2), (4.5.3)) can be obtained by applying the NOBAR-2D algorithm for the solution of a symmetric five term linear system of order $(p-1)$, semibandwidth $m$, retaining $r_1$ outermost off-diagonal entries. The elements of submatrices $\delta_1, \delta_2, \delta_3$, may be obtained algorithmically as follows:

for $j=1,2,\ldots,n-p+1$

$$
\bar{d}_{p+j-1} = \bar{d}_{p+j-2} + \bar{d}_{p+j-1}; \quad \bar{r}_{i,j} = \frac{h_i}{\bar{d}_{j}} \bar{d}_{p+j-1}; \quad (4.5.4)
$$

$$
\bar{v} = \frac{b_{p+j-2}}{\bar{d}_{p+j-1}} \bar{d}_{p+j-1}; \quad (4.5.5)
$$

whilst for $j \leq r_1 - 1$

$$
\bar{r}_{i,j} = -\bar{e}_{i+j-2} \bar{r}_{i-1,j}, \quad \text{for } i=2,3,\ldots,(r_1+1-j) \quad (4.5.6)
$$

then, if $j=1$ and $i=p-m+1$

$$
\bar{r}_{i,j} = \bar{r}_{i,j} + c_{i+j-1} \frac{\bar{d}_{i+j-1}}{\bar{d}_{p+j-1}}. \quad (4.5.7)
$$

If $i \geq p+1$ or $j > p-2$

$$
\bar{r}_{i,j} = -\bar{e}_{i+j-2} \bar{r}_{i-1,j} - \sum_{k=1}^{i-1} \bar{r}_{k-i+r_2+1,i+j-r_2-1,k,j} \quad (4.5.8)
$$

if $i=p-m+1$, then

$$
\bar{r}_{i,j} = \bar{r}_{i,j} + c_{i+j-1} \frac{\bar{d}_{i+j-1}}{\bar{d}_{p+j-1}}. \quad (4.5.9)
$$

else

if $i < m$, then

$$
\bar{r}_{i,j} = -\bar{e}_{i+j-2} \bar{r}_{i-1,j} - \sum_{k=1}^{i-1} \bar{r}_{k-i+r_1+1,i+j-r_1-1,k,j} \quad (4.5.10)
$$

else

$$
\bar{r}_{i,j} = -\bar{e}_{i+j-2} \bar{r}_{i-1,j} - \sum_{k=1}^{m-1} \bar{r}_{k,i+j-m,k+i-m,j} \quad (4.5.11)
$$

if $i=p-m+1$, then

$$
\bar{r}_{i,j} = \bar{r}_{i,j} + c_{i+j-1} \frac{\bar{d}_{i+j-1}}{\bar{d}_{p+j-1}} \bar{d}_{p+j-1}. \quad (4.5.12)
$$

for either $i=2,3,\ldots,r_2$ and all $j > r_1$ or $i=(r_1-j+2), (r_1-j+3), \ldots, r_2$ for $j \leq r_1$.
Then,

$$\vec{d}_{p+j-1} = \left\{ a_{p+j-1} / \left[ 1 + \sum_{k=1}^{r_1} \frac{r_2}{t_{k,j+p-m} + \sum_{k=1}^{r_2} \frac{r_v}{t_{k,j+v}}} \right] \right\}^{1/2}$$

(4.5.13)

and

$$\vec{s}_{p+j-2} = b_{p+j-2} \frac{\vec{d}_{p+j-2}}{p+j-2} \vec{d}_{p+j-1}$$

(4.5.14)

An approximate solution of the linear system (2.1.3) can now be obtained by writing,

$$\bar{D}_s \bar{T}' \bar{T}_s \bar{D}_u = \bar{s}$$

(4.5.15)

from which we obtain the normalized form

$$(\bar{T}'_s \bar{T}_s) \bar{D}_u = \bar{D}^{-1}_s \bar{s}$$

(4.5.16)

with \( \bar{y} = \bar{D}_u \) and \( \bar{g} = \bar{D}^{-1}_s \bar{s} \).

The problem is then to solve

$$\bar{T}'_s \bar{T}_s \bar{y} = \bar{g}$$

(4.5.18)

which can be solved directly for \( \bar{y} \) in terms of the auxiliary vector \( \bar{h}^* \), where

$$\bar{T}'_s \bar{y} = \bar{h}^* \quad \text{and} \quad \bar{T}'_s \bar{h}^* = \bar{g}$$

(4.5.19)

and are given by i.e.,

$$h_{i1}^* = g_{1} ;$$

(4.5.20)

$$h_{1}^* = g_{i-1} - \vec{e}_{i-1} h_{1}^* - \sum_{k=i-m+1}^{i-m+r_1} \vec{t}_{k-i+m,i-m+1} h^*_k, \quad \text{for } i=2,3,\ldots,m-1 ;$$

(4.5.21)

and

$$h_{i}^* = g_{i-1} - \vec{e}_{i-1} h_{1}^* - \sum_{k=i+m+1}^{i+m+1} \vec{t}_{k-i+m,i-m+1} h^*_k, \quad \text{for } i=m,m+1,\ldots,p-1. \quad \text{(4.5.22)}$$

$$h_{i}^* = g_{i-1} - \vec{e}_{i-1} h_{1}^* - \sum_{k=i+m+1}^{i+m+1} \vec{t}_{k-i+m,i-m+1} h^*_k - \sum_{k=i-p+1}^{i-p+1} \vec{t}_{k-i+p,i-p+1} h^*_k, \quad \text{for } i=p,p+1,\ldots,n. \quad \text{(4.5.23)}$$

The back substitution process yields the final solution and is expressed simply as

$$y_n = h_n^*$$

and

$$y_i = h_{i-1}^* - \vec{e}_{i-1} y_{i+1} - \sum_{j=p}^{i} \vec{q}_{i-j+m,j-m+1} y_{j-1} + \sum_{j=p}^{i} \vec{p}_{i-j+p,j-p+1} y_{j},$$

(4.5.25)

where \( \vec{p},\vec{q},\vec{p},\vec{q} \), can be easily defined from section 3.4.

The final solution \( \bar{u} \) is then obtained from

$$\bar{u} = \bar{D}^{-1}_s \bar{y}$$

The total memory requirement for the NOBAR-3D algorithm is...
\(= (r_1 + r_2 + 5)N \) words. The amount of work involved for the factorization process is given by \(= \frac{r_1(r_1-1) + r_2(r_2-1)}{2} + 4r_1 + 2r_2 + 12)N \) mults + 2N square roots. The normalization and forward-backward substitution processes require 2N divisions and \(= (2r_1 + 2r_2 + 2)N \) mults respectively. Hence, the total operations for this algorithm are

\[
= \frac{r_1(r_1-1) + r_2(r_2-1)}{2} + 6r_1 + 4r_2 + 16)N \text{ mults} + 2N \text{ square roots.}
\]

**Remark**

It should be noted at this point, that the given number of operations for the approximate algorithms i.e., ALUBOT-3D and NOBAR-3D, is not analogous to the corresponding number of operations for their exact counterparts, i.e., LUBOT3D and NB3D.

This is due to our main objective to design approximate 3D-algorithms keeping the memory requirements rather than the computational work involved to a minimum.
5.1 INTRODUCTION

Several procedures for approximate factorization of the coefficient matrix of a large, sparse, linear system have appeared in the literature and several variants of factorization methods combined with iterative methods have been developed (see: [22], [58], [56], [18], [19], [5], [16], [23], [7], [53], [2], [42], [10], [31]).

In this chapter, we introduce a normalized implicit method for the iterative solution of large, sparse systems of algebraic linear equations, which arise from the discretization on a network of grid lines of self adjoint elliptic P.D.E's. In particular we consider the Conjugate Gradient (C.G.) method and the Normalized Implicit Conjugate Gradient (N.I.C.G.) method, which is a combination of the approximate factorization technique of sections 4.3, 4.5 and the Conjugate Gradient method, for solving the model problems of section 2.2.

5.2 THE OPTIMUM VALUE OF THE FILL-IN PARAMETER r FOR THE 2D-MODEL PROBLEM

The normalized algorithm of section 3.3 was applied to many of the standard P.D.E's of Mathematical Physics, involving two space dimensions. In this algorithm the elements $t_{i,j}$ of the upper triangular matrix $T$ (see (3.3.4), satisfy the following theorem:

Theorem 5.2.1

Let $A$ be an ($n \times n$) matrix of bandwidth $m$, as is given by (3.3.2) with the properties (2.1.4) and consider the factorization $A = DT'TD$, where $D, T$ are given by (3.3.3), (3.3.4) respectively. Let $t_{i,j}$, $i \in [1, m-1]$, $j \in [1, n-m+1]$ be the elements of $T$-matrix and $r$ be the number of terms retained in bandwidth $m$. Then, the elements $t_{i,j}$
are monotonically decreased for $i \in [1, m-r]$. (i.e., the sequence $t_{1,j}, t_{2,j}, \ldots, t_{m-r,j}$, $j \in [1, n-m+1]$ decreases monotonically).

Proof

Because of the diagonally dominance of the coefficient matrix $A$ from (3.3.2) we generally obtain the inequality

$$a_i > |c_{i-m+1}| + |b_{i-1}| + |b_i| + |c_i|.$$  (5.2.1)

From the above relationship we can establish that no pivoting in the factorization is necessary for numerical stability. It can be easily seen that relations (3.3.5) and (3.3.9) give

$$|e_i| < 1, \text{ for } i=1,2,\ldots,n-1.$$  (5.2.2)

Then, from (3.3.6),(3.3.10) we have

$$|t_{1,j}| = \left| \frac{x_i}{d_{m+j-1}} \right| = \left| \frac{c_j}{d_{j}d_{m+j-1}} \right| < 1, \text{ for } j=1,2,\ldots,(n-m+1).$$  (5.2.3)

A combination of (3.3.7),(3.3.10) and (5.2.2) yields the result

$$|t_{i,j}| = \left| \frac{x_i}{d_{m+j-1}} \right| = \left| \frac{-e_i+j-2x_{i-1}}{d_{m+j-1}} \right| = \left| -e_i+j-2 \right| \cdot |t_{i-1,j}| < |t_{i-1,j}|,$$

for $i=2,3,\ldots,m-j$ and $j=1,2,\ldots,m-2$.  (5.2.4)

Similarly, from (3.3.8),(3.3.10) and (5.2.2) we obtain:

$$|t_{i,j}| = \left| \frac{x_i}{d_{m+j-1}} \right| = \left| \frac{-e_i+j-2x_{i-1}}{d_{m+j-1}} \right| = \left| \frac{-e_i+j-2x_{i-1}}{d_{m+j-1}} \right| + \frac{1}{d_{m+j-1}} \sum_{k=1}^{i-1} x_k t_{k-i+m,i+j-m}$$

$$+ M<|t_{i-1,j}| + M,$$

for either $i=(m-j+1),(m-j+2),\ldots,(m-1)$ and $j \leq m-1$

or $i=2,3,\ldots,m-1$ and $j > m-1$,

where

$$M = \frac{\left| \sum_{k=1}^{i-1} x_k t_{k-i+m,i+j-m} \right|}{d_{m+j-1}}, 0.$$  (5.2.6)
The quantity \( \sum_{k=1}^{i-1} x_k t^{k-i+m, i+j-m} \) can be modelled as a geometric series of the form \( \sum_{\lambda=1}^{n} x^\lambda t \) from which an upper bound can be easily determined.

(i.e. for the 2D model problem we can determine that

\[
\sum_{k=1}^{i-1} x_k t^{k-i+m, i+j-m} < 0.027 \quad (5.2.7)
\]

for either \( i=(m-j+1), (m-j+2), \ldots, (m-1) \) and \( j \leq m-1 \)

or \( i=2, 3, \ldots, m-1 \) for \( j > m-1 \).

For values of \( |t| \gg M \) and \( M \) positive, from (5.2.5) we immediately have the relation

\[
|t_{i,j}| < |t_{i-1,j}| \quad (5.2.8)
\]

for either \( i \in [m-j+1, m-r] \) and \( j \in [r+1, m-1] \)

or \( i \in [2, m-r] \) and \( j \in [m-1, n-m+1] \),

which guarantees the monotonicity of the terms in the \( T \) matrix.

From the relationships (5.2.3), (5.2.4) and (5.2.8) the conclusion of the theorem easily follows.

The contents of the \( T \) and \( D \) matrix arrays were scrutinised

(i) for matrices of constant order \( (N=50) \) with varying bandwidth \( m \) (\( m=10, 15, 20, 30, 40 \)) and

(ii) for matrices of varying order \( (N=50, 100, 200, 300, 400, 500, 1000) \) with constant bandwidth \( (m=20) \).

In particular, the Euclidean error norms of the approximate solutions \( u_r \) i.e., \( \left[ \sum (u-u_r)^2 \right]^{1/2} \) which were obtained by including only \( r \)-terms in the bandwidth, are given in Figures 5.1, 5.2 for the 2D-model problem.

A thorough examination of these results show that after \( r=4 \), the efficiency of the algorithm falls off rapidly and it requires a great deal of extra computational work, to achieve the required accuracy.
FIG. 5.1 : The behaviour of the NOBAR-2D Algorithm when the coefficient matrix is of order $N=50$ and the bandwidth $m$ is varied.
FIG. 5.2: The behaviour of the NOBAR-2D Algorithm when the order $N$ of the coefficient is varied with constant bandwidth ($m=20$).
Hence, for a certain class of 2D-model problems (see theorem 5.2.1) the NOBAR-2D algorithm can be efficiently applied with only the four outermost terms retained in the $T_s$ array.

Remark

From Theorem 5.2.1 it can be easily seen that for values of $|t_{i,j}|$ such that

$$\frac{|t_{i,j}|}{M} > 1$$

(5.2.9)

for either $i \in [m-j+1, m-1]$ and $j \in [2, m-1]$

or $i \in [2, m-1]$ and $j \in [m-1, n-m+1]$, the monotonicity relationship breaks down and does not apply. In particular it was noticed during the experimental investigation that in the case of narrow banded matrices the values of the elements of the $T$-matrix were not monotonic after a certain point ($i > m-r$). This is due to the values of $t_{i,j}$ becoming equal to or less than the magnitude as $M$.

Since in the examples chosen the above criteria for $M$ and $t_{i,j}$ were satisfied then we can safely adopt the procedure of neglecting the fill-in terms after $r$-terms.

Size effect of the coefficients of matrix $A$ on the value of $r$

In a further investigation we examine how the size of the coefficients of matrix $A$ of the linear system (3.2.15) effect the value of the fill-in parameter $r$.

We consider experimentally the situation where the co-diagonals $b_i$, $i \in [1, n-1]$ and $c_i$, $i \in [1, n-m+1]$ of (3.3.2) are of different size. Let the co-diagonals $b_i$ consist of larger (or strong) elements and the $m^{th}$-diagonals $c_i$ consist of small (or weak) elements. Then, we investigate the alternative case and by interchanging the above values i.e., the co-diagonals $b_i$ consist
of weak elements and the $m$th-diagonal $c_i$ consist of strong elements, we obtain the opposite situation. The error norms of the approximate solutions $u_r$, obtained by including only $r$-terms, are expressed graphically in Figure 5.3 for both cases.

From Figure 5.3 it is fairly obvious to establish that when the coefficient matrix has a strong co-diagonal the inclusion of more terms in the $T_s$ matrix produces a greater effect on the solution than if the co-diagonal is weak.

Consequently, if there are strong co-diagonal elements it is worthwhile to include as much $r$-terms as the computer storage requirements will permit, while in the case of weak co-diagonal elements the value of the fill-in parameter $r=2$ or $3$ is almost always the best and safest value to choose.

It can be observed that each of the illustrated curves of Figure 5.3 intersects the horizontal axis when $r=m-1$. Then the solution is obtained in one iteration, with the method being a direct one. The computational results demonstrating the above conclusions have been obtained for matrices of order $N=50$, bandwidth $m=10$ for the following cases:

(i) co-diagonal elements strong ($b_i=\frac{3}{8}, i\in[1,n-1]$) and $m$th-diagonal elements weak ($c_i=\frac{1}{8}, i\in[1,n-m+1]$),

(ii) standard case ($b_i=\frac{1}{4}, i\in[1,n-1]$ and $c_i=\frac{1}{4}, i\in[1,n-m+1]$) and

(iii) co-diagonal elements weak ($b_i=\frac{1}{8}, i\in[1,n-1]$) and $m$th-diagonal elements strong ($c_i=\frac{3}{8}, i\in[1,n-m+1]$).

The solution vector $u$ was chosen to be a unity vector and the error norm of the approximate solution $u_r$ has been taken to be the Euclidean error norm, i.e., $\|u-u_r\|_2$, $\|\sum(u-u_r)^2\|^\frac{1}{2}$.
r - Number of terms retained in bandwidth m

FIG. 5.3 : case i: Codiagonals strong - m\textsuperscript{th} diagonals weak,
case ii: Standard case,
case iii: Codiagonals weak - m\textsuperscript{th} diagonals strong.
5.3 THE CONJUGATE GRADIENT METHOD AS AN ITERATIVE PROCEDURE

One of the currently favourite iterative methods for the solution of large, sparse linear systems with an arbitrary symmetric, positive definite coefficient matrix is the Conjugate Gradient (C.G.) method, previously discussed in section 2.4. The reasons for its popularity are:

(i) it is easy to program [49],
(ii) it does not require any estimation of acceleration parameters,
(iii) it takes advantage of the distribution of the eigenvalues of the iteration operator,
(iv) it is relatively efficient [50].

Next, we present a revised form of the Conjugate Gradient algorithm in order to achieve a more convenient form.

Consider the linear system (2.4.1), where the coefficient matrix $A$ is symmetric, positive definite.

Given an initial approximation $x_0$ to the solution $x$ then, we form the residual $r_0$ such that

$$r_0 = b - Ax_0$$  \hspace{1cm} (5.3.1)

and let the directional vector $s_0$ be denoted as

$$s_0 = r_0$$  \hspace{1cm} (5.3.2)

Then the iterative scheme proceeds as follows. We calculate the scalar $a_i$, i.e.,

$$a_i = \frac{(r_i, r_i)}{(s_i, As_i)}$$  \hspace{1cm} (5.3.3)

or equivalently,

$$a_i = \frac{(r_i, s_i)}{(s_i, As_i)}$$  \hspace{1cm} (5.3.3a)

Then, we calculate the new iterate to minimize $F[x]$ along $s_i$ (see section 2.4),

$$x_{i+1} = x_i + a_i s_i$$  \hspace{1cm} (5.3.4)
Now the new residual \( r_{i+1} \) is formed such that
\[
r_{i+1} = r_i - a_i A a_i \tag{5.3.5}
\]
and the scalar \( \beta_{i+1} \) is calculated,
\[
\beta_{i+1} = \frac{(r_{i+1}, r_{i+1})}{(r_i, r_i)} \tag{5.3.6}
\]
or equivalently,
\[
\beta_{i+1} = \frac{-(r_{i+1}, A a_i)}{(a_i, A a_i)} \tag{5.3.6a}
\]
Then finally we calculate the new direction to be \( A \)-conjugate (see Definition 1.6.1) to the preceding direction such that,
\[
\sigma_{i+1} = r_{i+1} + \beta_{i+1} a_i \tag{5.3.7}
\]
Since the coefficient matrix \( A \) remains unmodified there is no need for it to be stored explicitly. The storage requirements and the number of multiplications per iteration required by the C.G. algorithm for the model problems are given in Tables 5.1 and 5.2.

Note that the C.G. algorithm can be efficiently applied for the solution of large, sparse systems of equations in the case where the matrix decomposition is considered to be impractical [46]. Furthermore, the algorithm can be applied when an approximate (relatively close) solution to the linear system is desirable (e.g. when the system is discrete approximation to a P.D.E.). Then, the level of correspondence of the model to the real system determines the required level of accuracy for the solution of the linear system and the algorithm could be terminated early [46].

Generally, the qualitative and quantitative behaviour of C.G. method is very well understood, [32], [38], [50], [20], [15].
5.4 THE NORMALIZED IMPLICIT CONJUGATE GRADIENT METHOD

In this section, we introduce the Normalized Implicit Conjugate Gradient (N.I.C.G.) method for solving the indicated model problems. As with the C.G. method, the N.I.C.G. method is very efficient, makes no assumptions about the structure of the coefficient matrix $A$, requires no estimation of iterative parameters and is easy to program.

Instead of considering the linear system (2.4.1) we now consider the linear system

\[(D_s T' T D_s)^{-1} A x = (D_s T' T D_s)^{-1} s\]  

(5.4.1)

where $D_s T' T D_s$ is the known approximate factorization of $A$, as given in sections 4.3, 4.5 with $D_s$, $T_s$ defined as in (4.3.2), (4.3.3) and (4.5.2), (4.5.3) respectively.

Once the factorization has been computed, the system (5.4.1) is solved by the C.G. method.

Let $x_0$ be an arbitrary initial approximation of the solution $x$, form the residual $r_0 = s - A x_0$.

We proceed to solve

\[(D_s T' T D_s)^{-1} r_0 = s\]  

(5.4.2)

and set $r_0 = s$.

Then, for $i=0,1,2,...$ calculate the vectors $x_{i+1}, r_{i+1}, \sigma_{i+1}$ and the scalar quantities $a_i, \beta_i$ as follows:

\[a_i = \frac{(r_i, r_{i+1})}{(s_i, A \sigma_i)}\]  

(5.4.3)

\[x_{i+1} = x_i + a_i \sigma_i\]  

(5.4.4)

and

\[r_{i+1} = r_i - a_i A \sigma_i\]  

(5.4.5)

Then we solve

\[(D_s T' T D_s)^{-1} r_{i+1} = s_{i+1}\]  

(5.4.6)

and evaluate

\[\beta_{i+1} = \frac{(r_{i+1}, \sigma_{i+1})}{(r_i, r_{i+1})}\]  

(5.4.7)
and 
\[ \sigma_{i+1} = \frac{r_i^+}{r_i + 1} + \beta_{i+1} \sigma_i \]  \hspace{1cm} (5.4.10) 

An equivalent, but more computational form of the iterative scheme (5.4.2)-(5.4.10) is given below:

Form 
\[ r_0 = s - Ax_0 \]  \hspace{1cm} (5.4.11) 

set 
\[ r_{-0} = D^{-1} r_0 \]  \hspace{1cm} (5.4.12) 

solve 
\[ (T'T_s) r_s^* = r_{-0} \]  \hspace{1cm} (5.4.13) 

and set 
\[ \sigma_{-0} = r_{-0} \]  \hspace{1cm} (5.4.14) 

Then for \( i=0,1,2,3,... \) calculate the vectors \( x_{i+1}, \quad \tilde{r}_{i+1}, \quad \tilde{\sigma}_{i+1} \) and the scalar quantities \( \tilde{\alpha}_{i+1}, \quad \tilde{\beta}_{i+1}, \quad \tilde{\sigma}_{i+1} \) as follows:

\[ \tilde{\alpha}_i = \frac{(\tilde{r}_i, r_i^*)}{(\tilde{\sigma}_i, r_i^*)} \]  \hspace{1cm} (5.4.15) 

\[ x_{i+1} = x_i + \tilde{\alpha}_i \tilde{r}_i \]  \hspace{1cm} (5.4.16) 

and 
\[ \tilde{r}_{i+1} = \tilde{r}_i - \tilde{\alpha}_i \tilde{\sigma}_i \]  \hspace{1cm} (5.4.17) 

Then, solve 
\[ (T'T_s) r_s^* = \tilde{r}_{i+1} \]  \hspace{1cm} (5.4.18) 

and evaluate 
\[ \tilde{\beta}_{i+1} = \frac{(\tilde{r}_{i+1}, r_{i+1})}{(\tilde{r}_i, r_i^*)} \]  \hspace{1cm} (5.4.19) 

and 
\[ \tilde{\sigma}_{i+1} = r_s^* - \tilde{\beta}_{i+1} \tilde{\sigma}_i \]  \hspace{1cm} (5.4.20) 

Geometrical representation of the N.I.C.G. method

Consider the linear system (2.4.1) and the corresponding quadratic function \( F[x] \), given by (2.4.2), which defines an ellipsoid \( E \) in the N-dimensional space of the elements of \( x \), as is shown in Figure 5.4.
FIGURE 5.4

The ellipses $c_i$, $i=0,1,2,...$ are the intersections of several planes defined by the vector $t_i$, tangent to the ellipse $c_i$ at $x_i$ and the residual $r_i$ for $i=0,1,2,...$, and the surface $F[x]=\text{constant}$. By """ we denote the projection of vectors and ellipses in the plane $P$.

Consider now the linear system (5.4.1) and the corresponding quadratic function $F_t[x]$, given by

$$F_t[x] = \frac{1}{2} x^T A^* x - s^* x \quad (5.4.21)$$

where

$$A^* = (D T^T D_s s - s_s^T D_s s_s) -1 A \quad \text{and} \quad s^* = (D T^T D_s s - s_s s_s) -1 s \quad (5.4.22)$$

[Note that, in the case of the approximate factorization $A^*DT_s T'D_s$ (see remark of section 4.3), the values of $A^*$ and $s^*$ in the quadratic function (5.4.21) are given by

$$A^* = (T^T D_s s) -1 T^T \quad \text{and} \quad s^* = (D_s T^T D_s s) -1 s \quad (5.4.23)$$

respectively].
Then, a new ellipsoid $E_{\tau}$ is defined by the transformed quadratic function $F_{\tau}[x]$, in the N-dimensional space of the elements of $x_i$ (see Figure 5.5), with the solution of the method occurring at the $x_k$.

![Diagram](image)

**FIGURE 5.5**

The ellipses $\tilde{c}_i$, are in this case, the intersections of several planes defined by the vector $\tilde{r}_i$, tangent to the ellipsoid $\tilde{c}_i$ at $x_i$, and the transformed residual $\tilde{r}_i$ for $i=0,1,2,...$, and the surface $F_{\tau}[x]=$constant.

Since the minimization of the transformed quadratic function (5.4.21) is equivalent to the solution of the system (5.4.1), the problem now reduces to the minimization of the quadratic function $F_{\tau}[x]$, instead of minimizing the quadratic function $F[x]$ as given by (2.4.2) and corresponding to the system (2.4.1).

Note that for non-positive definite matrices the existence of the minimum of the quadratic functions $F[x]$ and $F_{\tau}[x]$ respectively is not unique.
Let an arbitrary vector $x_0$ be the initial guess (a point on the surface of the ellipsoid under consideration) and consider the hyperellipsoid $F_\tau[x] = F_\tau[x_0]$ passing through this point.

The minimum of $F_\tau[x_0]$ occurs at the centre of this hyperellipsoid. Then, in each iteration we move along the chord of the hyperellipsoid, which is normal to the surface at $x_0$ (i.e., the direction of $\tilde{a}_0$) in order to arrive at $x_1$.

Note that $F_\tau[x_0] = F_\tau[x_0 + 2\tilde{a}_0 \tilde{a}_0]$ and consequently $x_1 = x_0 + \tilde{a}_0 \tilde{a}_0$ is the mid-point of this normal chord. Then, consider the projection of the hyperellipsoid $F_\tau[x] = F_\tau[x_1]$ in the (N-1) dimensional space, that is $A$-conjugate to $\tilde{a}_0$ and the above procedure is repeated. The iterative process is continued, so that at each iterative step the dimension of the space is decreased by one until finally after N-iterations all the space is evacuated or terminated when a suitable convergence criterion is satisfied.

The Three Dimensional Case

The algorithms of sections 5.3 and 5.4 can be generalized to apply to the solution of the self-adjoint, second order Elliptic P.D.E. in three space dimensions. Then, proceeding in an analogous manner to the two dimensional case, we consider the C.G.3D and the N.I.C.G.3D, which is a combination of the approximate factorization technique of section 4.5 and the C.G. algorithm, methods to solve the 3D-model problem.

5.5 COMPUTATIONAL WORK AND EXPERIMENTAL RESULTS IN TWO AND THREE DIMENSIONS

A summary of the storage and number of multiplications per iteration required for both methods applied to the model and general problems (i.e., the case of the self adjoint P.D.E's with variable coefficients in the unit square) in two and three dimensions, are given in Tables 5.1 and 5.2 respectively.
<table>
<thead>
<tr>
<th>Equation</th>
<th>Vectors in Storage</th>
<th>Storage Required</th>
<th>Number of mults per iteration</th>
<th>Number of mults for the solution of $(T_s^T r_s^* - r_s^*)$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Laplace:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C.G.</td>
<td>$r, x, \sigma, A_\sigma$</td>
<td>$4N$</td>
<td>$6N+2$</td>
<td>$0$</td>
</tr>
<tr>
<td>N.I.C.G.</td>
<td>$\vec{r}, x, \vec{\sigma}, A_\vec{\sigma}, a, b, s, u$ plus r-vectors in $T_s$ matrix</td>
<td>$(8+r)N-rn$</td>
<td>$(2r+8)N-rn-r$</td>
<td>$(2r+2)N-rn-r-2$</td>
</tr>
<tr>
<td><strong>General:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C.G.</td>
<td>$r, x, \sigma, A_\sigma, a, b, c$</td>
<td>$7N-n$</td>
<td>$10N-4n+2$</td>
<td>$0$</td>
</tr>
<tr>
<td>N.I.C.G.</td>
<td>$\vec{r}, x, \vec{\sigma}, A_\vec{\sigma}, a, b, c, s, u,$ plus r-vectors in $T_s$ matrix</td>
<td>$(9+r)N-rn$</td>
<td>$(2r+12)N-(r+4)n-r$</td>
<td>$(2r+2)N-rn-r-2$</td>
</tr>
</tbody>
</table>

**TABLE 5.1:** Two dimensional case.

Storage requirements and number of multiplications for $N \times N$ ($N = n^2$) systems, where $n = m-1$. 

115
<table>
<thead>
<tr>
<th>Equation</th>
<th>Vectors in Storage</th>
<th>Storage Required</th>
<th>Number of mults per iteration</th>
<th>Number of mults for solution of ((T_s^T T_s)^{-1})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Laplace:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C.G.3D</td>
<td>(r, x, g, Ag)</td>
<td>4N</td>
<td>6N+2</td>
<td>0</td>
</tr>
<tr>
<td>N.I.C.G.3D</td>
<td>(\tilde{r}, x, \delta, a, a, b, s, u) plus (r_1) and (r_2) vectors in (T_s) matrix</td>
<td>((8+r_1+r_2)N-r_2n^2-r_1n)</td>
<td>((2r_1+2r_2+8)N-r_2n^2-r_1n-r_1-r_2)</td>
<td>((2r_1+2r_2+2)N-r_2n^2-r_1n-r_1-r_2)</td>
</tr>
<tr>
<td>General:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C.G.3D</td>
<td>(r, x, g, A_0, a, b, c, h)</td>
<td>8N(-n^2-n)</td>
<td>12N(-6n^2+2)</td>
<td>0</td>
</tr>
<tr>
<td>N.I.C.G.3D</td>
<td>(\tilde{r}, x, \delta, A_0, a, b, c, h, s, u) plus (r_1) and (r_2) vectors in (T_s) matrix</td>
<td>((10+r_1+r_2)N-(1+r_2)n^2-(1+r_1)n)</td>
<td>((2r_1+2r_2+14)N-(r_2+6)n^2-r_1n-r_1-r_2)</td>
<td>((2r_1+2r_2+2)N-r_2n^2-r_1n-r_1-r_2)</td>
</tr>
</tbody>
</table>

**TABLE 5.2:** Three dimensional case.

Storage requirements and number of multiplications for 
\(N \times N \ (N=n^3)\) systems, where \(n^2=p-1\).
It can be easily seen from Tables 5.1, 5.2 that an upper bound on the arithmetic work per iteration for the N.I.C.G. method is $O(20N)$ multiplications when $r=4$ and $O(14N)$ multiplications when $r=1$, whilst for the N.I.C.G.3D method we obtain $O(18N)$ multiplications when $r_1=r_2=1$ and $O(30N)$ multiplications when $r_1=r_2=4$ respectively.

The numerical results for both methods on the model problems, are given in Tables 5.3 and 5.4 respectively.

The initial "guess" $x_0$ was chosen to be the zero vector.

The solution vector was chosen to be:

(i) $x_i=0$, $i \in [1,N]$ with $x_{m+1}=1$, (case II)
and

(ii) a vector of $N$-pseudo-random numbers from a uniform (rectangular) distribution on the range $(0,1)$, (case I).

The right hand side vector of equation (2.4.1) was obtained as the product of the solution vector $x$ with the coefficient matrix $A$ of (2.4.1). Following Ginsburg [20](p.68) and Reid [50](p.243) the error was taken to be the maximum norm of the recursive residual. The iterative process was terminated when the error was less than the relative precision of the arithmetic, which has been chosen to be $10^{-6}$, $10^{-8}$, $10^{-10}$ respectively. The behaviour of the following four error measures

$$||r_i||_2, (r_i(T^T T_i^{-1} r_i))^{\frac{1}{2}}, (r_i(x_k-x))^\frac{1}{2}, ||x_k-x||_2,$$

where $x_k$ means the $k$th approximant to the exact solution $x$, applied to the Laplacian matrices of order $(49 \times 49), (12 \times 12 \times 12)$ respectively, and for the model problems, is given in Figures 5.7-5.10.

The effect of number of equations on the rate of convergence for the 2D-model problem case II, is given in Figure 5.6. The computations were performed on an ICL 1904 computer. In addition, extreme cases of the problems were computed on an ICL 1906A machine.
<table>
<thead>
<tr>
<th>Mesh size h&lt;sup&gt;−1&lt;/sup&gt;</th>
<th>Case I</th>
<th>Case II</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C.G. N.I.C.G.</td>
<td>C.G. N.I.C.G.</td>
</tr>
<tr>
<td></td>
<td>r=1 2 3 4</td>
<td>r=1 2 3 4</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>10</td>
<td>25</td>
<td>12</td>
</tr>
<tr>
<td>15</td>
<td>40</td>
<td>15</td>
</tr>
<tr>
<td>20</td>
<td>53</td>
<td>19</td>
</tr>
<tr>
<td>30</td>
<td>75</td>
<td>24</td>
</tr>
<tr>
<td>40</td>
<td>101</td>
<td>31</td>
</tr>
<tr>
<td>50</td>
<td>&gt;110</td>
<td>37</td>
</tr>
<tr>
<td>60</td>
<td>&gt;110</td>
<td>43</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mesh size h&lt;sup&gt;−1&lt;/sup&gt;</th>
<th>Case I</th>
<th>Case II</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C.G. N.I.C.G.</td>
<td>C.G. N.I.C.G.</td>
</tr>
<tr>
<td></td>
<td>r=1 2 3 4</td>
<td>r=1 2 3 4</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>-10</td>
</tr>
<tr>
<td>10</td>
<td>31</td>
<td>14</td>
</tr>
<tr>
<td>15</td>
<td>47</td>
<td>20</td>
</tr>
<tr>
<td>20</td>
<td>63</td>
<td>25</td>
</tr>
<tr>
<td>30</td>
<td>91</td>
<td>35</td>
</tr>
<tr>
<td>40</td>
<td>123</td>
<td>40</td>
</tr>
<tr>
<td>50</td>
<td>&gt;130</td>
<td>49</td>
</tr>
<tr>
<td>60</td>
<td>&gt;130</td>
<td>58</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mesh size h&lt;sup&gt;−1&lt;/sup&gt;</th>
<th>Case I</th>
<th>Case II</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C.G. N.I.C.G.</td>
<td>C.G. N.I.C.G.</td>
</tr>
<tr>
<td></td>
<td>r=1 2 3 4</td>
<td>r=1 2 3 4</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>-10</td>
</tr>
<tr>
<td>10</td>
<td>35</td>
<td>18</td>
</tr>
<tr>
<td>15</td>
<td>53</td>
<td>23</td>
</tr>
<tr>
<td>20</td>
<td>71</td>
<td>28</td>
</tr>
<tr>
<td>30</td>
<td>105</td>
<td>40</td>
</tr>
<tr>
<td>40</td>
<td>140</td>
<td>52</td>
</tr>
<tr>
<td>50</td>
<td>&gt;140</td>
<td>63</td>
</tr>
<tr>
<td>60</td>
<td>&gt;140</td>
<td>73</td>
</tr>
</tbody>
</table>

**TABLE 5.3:** Number of iterations required to reduce the error to 1E-6, 1E-8, 1E-10 respectively for the 2D-model problem. The resulting sparse matrices, for the above considered mesh sizes, are of order 16,81,196, 361,841,1421,2401,3481 respectively.  
(4) For r=m-1 the method becomes a direct one.
<table>
<thead>
<tr>
<th>Mesh size ( h^{-1} )</th>
<th>Case I</th>
<th>Case II</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C.G.3D</td>
<td>N.I.C.G.3D</td>
</tr>
<tr>
<td></td>
<td>( r_1=r_2=1 )</td>
<td>( r_1=r_2=4 )</td>
</tr>
<tr>
<td>( \varepsilon=10^{-6} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>18</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>27</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>33</td>
<td>13</td>
</tr>
<tr>
<td>11</td>
<td>41</td>
<td>14</td>
</tr>
<tr>
<td>13</td>
<td>47</td>
<td>16</td>
</tr>
<tr>
<td>( \varepsilon=10^{-8} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>22</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>33</td>
<td>14</td>
</tr>
<tr>
<td>9</td>
<td>43</td>
<td>17</td>
</tr>
<tr>
<td>11</td>
<td>53</td>
<td>20</td>
</tr>
<tr>
<td>13</td>
<td>62</td>
<td>24</td>
</tr>
<tr>
<td>( \varepsilon=10^{-10} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>26</td>
<td>13</td>
</tr>
<tr>
<td>7</td>
<td>38</td>
<td>16</td>
</tr>
<tr>
<td>9</td>
<td>50</td>
<td>20</td>
</tr>
<tr>
<td>11</td>
<td>62</td>
<td>24</td>
</tr>
<tr>
<td>13</td>
<td>73</td>
<td>28</td>
</tr>
</tbody>
</table>

**TABLE 5.4:** Number of iterations required to reduce the error to 1E-6, 1E-8, 1E-10 respectively for the 3D-model problem. The resulting sparse matrices, for the above considered mesh sizes, are of order 64, 216, 512, 1000, 1728 respectively.
Fig. 5.6: Effect of number of equations on the rate of convergence (2D-model problem, case II).
FIG. 5.7 : Behaviour of the Error measures applied to the Laplacian matrix of order N=2401 (Grid 49x49).

Log of the Error Norm

Number of iterations

N.I.C.G., r=4, case II
FIG. 5.8: Behaviour of the Error measures applied to the Laplacian matrix of order $N=2401$ (Grid $49 \times 49$).
FIG. 5.9: Behaviour of the Error measures applied to the Laplacian matrix of order $N = 1728$ (Grid $12 \times 12 \times 12$).
FIG. 5.10 : Behaviour of the error measures applied to the matrix of order $N=1728$ (Grid $12 \times 12 \times 12$).
Chapter 6

Normalized Second Order Methods
6.1 INTRODUCTION

In this chapter, the representation of the Conjugate Gradient method as a non-stationary second degree iteration method is given. We, then, develop the Normalized Implicit Conjugate Gradient method as a second order method, giving a proposal for certain values of the iterative parameters \( \rho_1, \gamma_1 \) involved in both two and three space dimensions. Next, we proceed to apply the approximate two dimensional algorithms in conjunction with the standard well known stationary and non-stationary iterative methods, i.e., the Simultaneous Displacement, Second order Richardson and Chebychev methods. An experimental estimation of the optimal iteration parameters involved are obtained, resulting in a substantial saving in computational work.

6.2 THE CONJUGATE GRADIENT METHOD AS A SECOND DEGREE METHOD (C.G.S.D.)

In the following, we derive an expression for \( x_{i+1} \) in terms of \( x_i \) and \( x_{i-1} \). By (5.3.4) we have

\[
x_{i+1} = x_i + a_{i-1} \sigma_{i-1}
\]  

(6.2.1)

and

\[
x_{i+1} = x_{i-1} + a_{i-1} \sigma_{i-1}
\]

(6.2.2)

or

\[
\frac{\beta_{i}}{a_{i-1}} a_{i-1} x_{i-1} = \frac{\beta_{i}}{a_{i-1}} a_{i-1} x_{i-1} + a_{i-1} \beta_{i} \sigma_{i-1} \cdot
\]

(6.2.3)

From (6.2.1) and (5.3.7) we obtain

\[
x_{i+1} = x_i + a_i (r_i + \beta_{i} \sigma_{i-1})
\]

(6.2.4)

and a combination of (6.2.3), (6.2.4) leads to

\[
x_{i+1} = x_i + a_i r_i + \frac{\beta_{i}}{a_{i-1}} a_{i-1} x_{i-1} - \frac{\beta_{i}}{a_{i-1}} a_{i-1} x_{i-1}
\]

(6.2.5)

or equivalently,

\[
x_{i+1} = x_{i-1} + \rho_{i+1} (r_{i+1} + x_{i-1} - x_i)
\]

(6.2.6)

where

\[
\rho_{i+1} = 1 + \frac{a_i}{a_{i-1}} \beta_i
\]

(6.2.7)
$$
\gamma_{i+1} = \frac{a_i}{p_{i+1}}, \quad i=0,1,2,...,k-1. \quad (6.2.8)
$$

with \( k \) the smallest integer such that \( \frac{r_k}{r_{k-1}} = 0 \).

The derivation of formulae for the evaluation of parameters \( \rho_{i+1} \), \( \gamma_{i+1} \) using the orthogonality relations (2.4.20) follows from equation (6.2.6) and by substituting

$$
\bar{r}_i = s - A\bar{r}_i, \quad i=0,1,2,... \quad (6.2.9)
$$

we obtain the result

$$
\bar{r}_{i+1} = \bar{r}_{i-1} + \rho_{i+1} (-\gamma_{i+1} A\bar{r}_{i-1} + \bar{r}_{i-1} - \bar{r}_{i-2}) \quad (6.2.10)
$$

By forming the inner product of both sides of equation (6.2.10) with \( \bar{r}_i \) we have,

$$
0 = \rho_{i+1} (-\gamma_{i+1} (\bar{r}_i, A\bar{r}_{i-1}) + (\bar{r}_i, \bar{r}_{i-1})) \quad (6.2.11)
$$

and from (6.2.11) assuming that \( \rho_{i+1} \neq 0 \) we get,

$$
\gamma_{i+1} = \frac{(\bar{r}_i, \bar{r}_i)}{\bar{r}_i A\bar{r}_{i-1}} \quad . \quad (6.2.12)
$$

Similarly, by taking the inner product of both sides of equation (6.2.10) with \( \bar{r}_{i-1} \) yields:

$$
0 = (\bar{r}_{i-1}, \bar{r}_{i-1}) + \rho_{i+1} (-\gamma_{i+1} (\bar{r}_{i-1}, A\bar{r}_{i-1}) - (\bar{r}_{i-1}, \bar{r}_{i-1})) \quad (6.2.13)
$$

and replacing index \( i \) by \( i-1 \) in (6.2.10) we obtain,

$$
\bar{r}_i = \bar{r}_{i-2} + \rho_i (-\gamma_i A\bar{r}_{i-1} + \bar{r}_{i-1} - \bar{r}_{i-2}) \quad . \quad (6.2.14)
$$

Taking the inner product of both sides of (6.2.14) with \( \bar{r}_i \) yields:

$$
(\bar{r}_i, \bar{r}_i) = \rho_i (-\gamma_i (\bar{r}_i, A\bar{r}_{i-1})) \quad (6.2.15)
$$

or

$$
(\bar{r}_{i-1}, A\bar{r}_{i-1}) = (\bar{r}_i, A\bar{r}_{i-1}) = -\frac{(\bar{r}_i, \bar{r}_i)}{\gamma_i \rho_i} \quad (6.2.16)
$$

and then, combining (6.2.13) with (6.2.16) we obtain the result
\[ \rho_{i+1} = \left[ 1 + \frac{(r_{i-1} \cdot A r_i)}{(r_{i-1} \cdot r_i)} \gamma_{i+1} \right]^{-1} = \left[ 1 - \frac{(r_{i-1} \cdot r_i)}{(r_{i-1} \cdot r_i)} \frac{\gamma_{i+1}}{\gamma_i} \frac{1}{\rho_i} \right]^{-1} \]

for \( i = 1, 2, 3, \ldots \). (6.2.17)

(Note that \( \rho_1 = 1 \), since \( x_1 = x_0 + a_0 r_0 \)).

The formulae (6.2.12), (6.2.17) are used to generate the parameters \( \rho_i, \gamma_i \) in a recursive form in the C.G. method. An alternative derivation of these formulae can be found in [62].

6.3 THE NORMALIZED IMPLICIT CONJUGATE GRADIENT METHOD AS A SECOND DEGREE METHOD (N.I.C.G.S.D.)

A similar three-term formula analogous to (6.2.6) for \( x_{i+1} \), involving \( x_i \) and \( x_{i-1} \), can be derived as follows:

From (5.4.16) we obtain

\[ x_{i+1} = x_i + \alpha_{i} \tilde{\sigma}_i \]

and

\[ x_i = x_{i-1} + \alpha_{i-1} \tilde{\sigma}_{i-1} \]

or

\[ \tilde{\beta}_i \alpha_{i-1} \tilde{\alpha}_i \tilde{\sigma}_i = \tilde{\beta}_i \alpha_{i-1} \tilde{\alpha}_i \tilde{\sigma}_{i-1} \tilde{\sigma}_{i-1} \] (6.3.3)

A combination of (6.3.1), (6.3.3) and (5.4.20) leads to

\[ x_{i+1} = x_{i-1} + \tilde{\beta}_{i+1} \tilde{\gamma}_{i+1} r_{i+1}^* + x_{i-1} - x_{i-1} \] (6.3.4)

where the parameters \( \tilde{\beta}_i, \tilde{\gamma}_i \) are given by

\[ \tilde{\beta}_{i+1} = 1 + \frac{\tilde{\alpha}_i}{\tilde{\beta}_i} \tilde{\beta}_i \] (6.3.5)

and

\[ \tilde{\gamma}_{i+1} = \frac{\tilde{\alpha}_i}{\tilde{\beta}_i} \] (6.3.6)

for \( i = 0, 1, 2, 3, \ldots k-1 \), where \( k \) is the smallest integer, such that \( \frac{\tilde{\gamma}_k}{\tilde{\gamma}_k} = 0 \) (see equations (5.4.12), (5.4.17)).

Next, we derive the analogue formulae of (6.2.12), (6.2.17) by use of the orthogonality relations (2.4.20) and a combination of (5.4.12), (5.4.18), (6.2.9) yields
\[ r^*_i = \tilde{s} - \tilde{G} \tilde{x}_i \]  \hspace{1cm} (6.3.7)

where

\[ \tilde{s} = \tilde{G} \tilde{s} \quad \text{and} \quad \tilde{G} = (T^T T)^{-1} D^T s \quad A . \]  \hspace{1cm} (6.3.8)

From (6.3.4), (6.3.7) we obtain

\[ r^*_{i+1} = r^*_i - \tilde{G}^*_i + \tilde{r}^*_i - r^*_{i-1} \]  \hspace{1cm} (6.3.9)

and forming the inner product of both sides of (6.3.9) with \( r^*_i \)

we have

\[ 0 = \tilde{\rho}^*_{i+1} (-\tilde{\gamma}^*_{i+1} (r^*_i, r^*_{i+1}) + (r^*_i, r^*_{i+1})) \]  \hspace{1cm} (6.3.10)

Then, assuming that \( \tilde{\rho}^*_{i+1} \neq 0 \) we get

\[ \tilde{\gamma}^*_{i+1} = \frac{(r^*_i, r^*_i)}{(r^*_i, \tilde{G} r^*_{i-1})} \]  \hspace{1cm} (6.3.11)

Forming the inner product of both sides of (6.3.9) with \( r^*_{i-1} \)

we have

\[ 0 = (r^*_{i-1}, r^*_{i-1}) + \tilde{\rho}^*_{i+1} (-\tilde{\gamma}^*_{i+1} (r^*_i, \tilde{G} r^*_{i-1}) - (r^*_i, r^*_i)) \]  \hspace{1cm} (6.3.12)

or

\[ \tilde{\rho}^*_{i+1} = \left[ 1 + \frac{(r^*_i, r^*_{i-1})}{(r^*_i, \tilde{G} r^*_{i-1})} \tilde{\gamma}^*_{i+1} \right]^{-1} \]  \hspace{1cm} (6.3.13)

Also from (6.3.9) we get

\[ r^*_i = r^*_i - \tilde{\gamma}^*_{i+1} (r^*_i, \tilde{G} r^*_{i-1}) + (r^*_i, r^*_i) \]  \hspace{1cm} (6.3.14)

and taking the inner product of both sides with \( r^*_i \) we obtain

\[ (r^*_i, r^*_i) = \tilde{\rho}^*_{i} (-\tilde{\gamma}^*_{i} (r^*_i, \tilde{G} r^*_{i-1})) \]  \hspace{1cm} (6.3.15)

or

\[ (r^*_i, \tilde{G} r^*_{i-1}) = (r^*_i, \tilde{G} r^*_{i-1}) = - \frac{(r^*_i, r^*_i)}{\tilde{\rho}^*_{i} \tilde{\gamma}^*_{i}} \]  \hspace{1cm} (6.3.16)

Then, the substitution of (6.3.16) in (6.3.13) leads to the recursive formulae

\[ \tilde{\rho}^*_{i+1} = \left[ 1 - \frac{(r^*_i, r^*_i)}{(r^*_i, \tilde{G} r^*_{i-1})} \tilde{\gamma}^*_{i+1} \frac{1}{\tilde{\rho}^*_{i}} \right]^{-1} \]  \hspace{1cm} (6.3.17)

for \( i = 1, 2, 3, \ldots \)

with \( \tilde{\rho}^*_1 = 1 \), since \( x_0 = x_0^* + \tilde{x}_0 \).
The formulae (6.3.11), (6.3.17) can now be used for the generation of the parameters \( \tilde{\rho}_i, \tilde{\gamma}_i \).

Alternatively, using the relations (5.4.3), (5.4.8), (6.2.9) we obtain

\[
\bar{r}_i = s^+ - G^+ x_i, \quad (6.3.18)
\]

where \( s^+ = G^+ s \) and \( G^+ = (D \, T' \, T \, D)^{-1} A \). (6.3.19)

Proceeding in a similar manner as above, we obtain the following formulae for \( \rho^+_i, \gamma^+_i \):

\[
\gamma^+_{i+1} = \frac{(r^+_i, r^+_i)}{(r^+_i, G^+ x^+_i)}, \quad (6.3.20)
\]

and

\[
\rho^+_{i+1} = \left[ 1 - \frac{(r^+_i, x^+_i)}{(r^+_{i-1}, x^+_{i-1})} \cdot \frac{\gamma^+_{i+1}}{\gamma^+_i} \cdot \frac{1}{\rho^+_i} \right]^{-1}. \quad (6.3.21)
\]

It is easily seen that the values of parameters \( \rho^+_i, \gamma^+_i \) given by (6.3.21), (6.3.20) are easily obtained and when \( G^+ \) has the form

\[
G^+ = (T' T s)^{-1} T' T, \quad (6.3.22)
\]

which can be shown to be related to the approximate factorized form of \( A \) discussed earlier in Chapter 4.

A Factor Affecting the Stability Conditions

It can be shown, [32] (p.420), that when the ratio \( a_i / a_{i-1} \) is large (see equation (6.2.7)) the disturbance of the orthogonality relations in the C.G. method will be greatest. Consequently, the larger the ratio \( \tilde{a}_i / \tilde{a}_{i-1} \) (see equation (6.3.5)) the more rapidly rounding-off error will accumulate.

Since the scalars \( a_i, \tilde{a}_i \) lie in the range \( \frac{1}{M m} \), where \( m, M \) are the extreme eigenvalues of the coefficient matrix \( A \), it follows that \( M/m \) is an upper bound of the ratios \( a_i / a_{i-1} \) and \( \tilde{a}_i / \tilde{a}_{i-1} \) which directly affect the sensitivity to
round-off errors of the C.G. and N.I.C.G. processes respectively.

Consequently for any pre-assigned set of values of the scalars \(a_i\) and \(\tilde{a}_i\), such that \((a_i/a_{i-1})<1\) and \((\tilde{a}_i/\tilde{a}_{i-1})<1\) respectively, the stability of the method with respect to the growth of rounding-off errors can be satisfied (i.e., the algorithms are stable).

**Computational Results**

Proceeding in an analogous way to the two dimensional case, we can now extend the C.G.S.D. and N.I.C.G.S.D. methods in three space dimensions, obtaining the C.G.3D.S.D. and N.I.C.G.3D.S.D. methods respectively.

Numerical results for C.G.S.D., N.I.C.G.S.D. and C.G.3D.S.D., N.I.C.G.3D.S.D. methods for the model problems are given in Tables 6.1, 6.2, including the values of the parameters \(\rho_n, \gamma_n\) at the end of the \(n\)th iteration. For comparative purposes two cases of the N.I.C.G.S.D. \((r=1, r=4)\) and N.I.C.G.3D.S.D. \((r_1=r_2=1, r_1=r_2=4)\) methods have been considered during the experiments. The error was computed as in section 5.4 and the same termination criterion and initial guess vector used. The solution vector was chosen to be:

\[
x_i = 0, \quad i \in [1, N] \text{ with } x_{m+1} = 1, \quad i \neq m+1
\]

and the right hand side vector of equation (2.4.1) was obtained as the product of the solution vector with the coefficient matrix \(A\) of (2.4.1).

In order to observe the behaviour of the parameters \(\rho_n, \gamma_n\) of the C.G.S.D., C.G.3D.S.D. methods and \(\rho_n^+, \gamma_n^+\) of the N.I.C.G.S.D., N.I.C.G.3D.S.D. methods, as the grid size of the model problems is increased, comparative figures are given in Figures 6.2-6.4, for both two and three dimensional cases.
Finally, a comparison of the results of the N.I.C.G.S.D., N.I.C.G.3D.S.D methods, as given in Tables 6.1, 6.2, with those of N.I.C.G., N.I.C.G.3D (case II), as given in Tables 5.3, 5.4, shows that inspite of the increase in the computational work involved using second order methods, there is no substantial gain in the number of iterations of these methods against those of first order.
<table>
<thead>
<tr>
<th>Mesh Size</th>
<th>( \rho_n )</th>
<th>( \gamma_n )</th>
<th>No. of Iter. ( n )</th>
<th>( \hat{\rho}_n )</th>
<th>( \hat{\gamma}_n )</th>
<th>( n )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \varepsilon = 10^{-6} )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1.069053</td>
<td>0.298162</td>
<td>8</td>
<td>1.007466</td>
<td>0.287808</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>1.099267</td>
<td>0.223776</td>
<td>24</td>
<td>1.027916</td>
<td>0.294186</td>
<td>9</td>
</tr>
<tr>
<td>15</td>
<td>1.300436</td>
<td>0.264003</td>
<td>37</td>
<td>1.029364</td>
<td>0.369555</td>
<td>12</td>
</tr>
<tr>
<td>20</td>
<td>1.506623</td>
<td>0.249149</td>
<td>48</td>
<td>1.112857</td>
<td>0.358697</td>
<td>15</td>
</tr>
<tr>
<td>30</td>
<td>1.601615</td>
<td>0.259928</td>
<td>66</td>
<td>1.194302</td>
<td>0.422701</td>
<td>20</td>
</tr>
<tr>
<td>40</td>
<td>1.866241</td>
<td>0.243659</td>
<td>79</td>
<td>1.688252</td>
<td>0.508812</td>
<td>23</td>
</tr>
<tr>
<td>( \varepsilon = 10^{-8} )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1.069053</td>
<td>0.298162</td>
<td>~8</td>
<td>1.039101</td>
<td>0.319471</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>1.029541</td>
<td>0.249137</td>
<td>27</td>
<td>1.005562</td>
<td>0.297932</td>
<td>11</td>
</tr>
<tr>
<td>15</td>
<td>1.239204</td>
<td>0.263739</td>
<td>44</td>
<td>1.067375</td>
<td>0.351895</td>
<td>15</td>
</tr>
<tr>
<td>20</td>
<td>1.490244</td>
<td>0.267305</td>
<td>57</td>
<td>1.061609</td>
<td>0.357688</td>
<td>18</td>
</tr>
<tr>
<td>30</td>
<td>1.481521</td>
<td>0.247052</td>
<td>82</td>
<td>1.112554</td>
<td>0.398684</td>
<td>25</td>
</tr>
<tr>
<td>40</td>
<td>1.646100</td>
<td>0.253540</td>
<td>106</td>
<td>1.211116</td>
<td>0.470903</td>
<td>31</td>
</tr>
<tr>
<td>( \varepsilon = 10^{-10} )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1.069053</td>
<td>0.298162</td>
<td>~8</td>
<td>1.004009</td>
<td>0.291505</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>1.054674</td>
<td>0.275596</td>
<td>30</td>
<td>1.315973</td>
<td>0.407352</td>
<td>14</td>
</tr>
<tr>
<td>15</td>
<td>1.155045</td>
<td>0.253499</td>
<td>49</td>
<td>1.017133</td>
<td>0.329981</td>
<td>17</td>
</tr>
<tr>
<td>20</td>
<td>1.264603</td>
<td>0.249825</td>
<td>66</td>
<td>1.033903</td>
<td>0.367417</td>
<td>21</td>
</tr>
<tr>
<td>30</td>
<td>1.530361</td>
<td>0.245857</td>
<td>97</td>
<td>1.189517</td>
<td>0.460586</td>
<td>30</td>
</tr>
<tr>
<td>40</td>
<td>1.771570</td>
<td>0.246147</td>
<td>125</td>
<td>1.191915</td>
<td>0.449522</td>
<td>38</td>
</tr>
</tbody>
</table>

**Table 6.1**

Parameters \( \rho_n, \gamma_n, \hat{\rho}_n, \hat{\gamma}_n \) and work required to reduce the error to 1E-6, 1E-8, 1E-10 respectively for the 2D-model problem. The resulting sparse matrices for the above considered mesh sizes are of order 16,81,196,361,841,1421 respectively.

(*) For \( r=m-1 \) the method becomes a direct one.
<table>
<thead>
<tr>
<th>Mesh Size $k^2$</th>
<th>$\rho_0$</th>
<th>$\gamma_0$</th>
<th>No. of Iter. $n$</th>
<th>$\tilde{\rho}_n$</th>
<th>$\tilde{\gamma}_n$</th>
<th>$\tilde{n}$</th>
<th>$\tilde{\rho}_n$</th>
<th>$\tilde{\gamma}_n$</th>
<th>$\tilde{n}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e=10^{-6}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1.098210</td>
<td>0.161351</td>
<td>17</td>
<td>1.00534510</td>
<td>0.18877492</td>
<td>6</td>
<td>1.00649336</td>
<td>0.18959059</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>1.349571</td>
<td>0.175824</td>
<td>24</td>
<td>1.01636366</td>
<td>0.20085404</td>
<td>8</td>
<td>1.02070535</td>
<td>0.21138891</td>
<td>7</td>
</tr>
<tr>
<td>9</td>
<td>1.383097</td>
<td>0.164590</td>
<td>28</td>
<td>1.04634084</td>
<td>0.21988940</td>
<td>9</td>
<td>1.04019646</td>
<td>0.22447190</td>
<td>8</td>
</tr>
<tr>
<td>11</td>
<td>1.387208</td>
<td>0.164679</td>
<td>34</td>
<td>1.06898420</td>
<td>0.23945461</td>
<td>11</td>
<td>1.06532638</td>
<td>0.23534745</td>
<td>9</td>
</tr>
<tr>
<td>$e=10^{-8}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1.106920</td>
<td>0.133364</td>
<td>21</td>
<td>1.01005623</td>
<td>0.19751844</td>
<td>8</td>
<td>1.00659299</td>
<td>0.19161870</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>1.281821</td>
<td>0.164165</td>
<td>31</td>
<td>1.02800411</td>
<td>0.20962749</td>
<td>11</td>
<td>1.02042768</td>
<td>0.20104534</td>
<td>9</td>
</tr>
<tr>
<td>9</td>
<td>1.357018</td>
<td>0.156382</td>
<td>38</td>
<td>1.09045951</td>
<td>0.25259443</td>
<td>13</td>
<td>1.03682028</td>
<td>0.21623437</td>
<td>11</td>
</tr>
<tr>
<td>11</td>
<td>1.565861</td>
<td>0.184152</td>
<td>45</td>
<td>1.14026254</td>
<td>0.26781433</td>
<td>15</td>
<td>1.06795429</td>
<td>0.22639343</td>
<td>13</td>
</tr>
<tr>
<td>$e=10^{-10}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1.089927</td>
<td>0.201645</td>
<td>24</td>
<td>1.11618224</td>
<td>0.21464544</td>
<td>11</td>
<td>1.00251448</td>
<td>0.18345553</td>
<td>9</td>
</tr>
<tr>
<td>7</td>
<td>1.192356</td>
<td>0.170759</td>
<td>36</td>
<td>1.02040246</td>
<td>0.21501709</td>
<td>13</td>
<td>1.00778659</td>
<td>0.19175601</td>
<td>11</td>
</tr>
<tr>
<td>9</td>
<td>1.243097</td>
<td>0.163397</td>
<td>46</td>
<td>1.03680503</td>
<td>0.20917582</td>
<td>16</td>
<td>1.01369925</td>
<td>0.20019031</td>
<td>13</td>
</tr>
<tr>
<td>11</td>
<td>1.303604</td>
<td>0.159903</td>
<td>55</td>
<td>1.12175850</td>
<td>0.23319397</td>
<td>18</td>
<td>1.05927287</td>
<td>0.24077135</td>
<td>16</td>
</tr>
</tbody>
</table>

**Table 6.2**

Parameters $\rho_0, \gamma_0, \tilde{\rho}_n, \tilde{\gamma}_n$ and work required to reduce the error to $1E-6$, $1E-8$, $1E-10$ respectively for the 3D-model problem. The resulting sparse matrices for the above considered mesh sizes are of order 64, 216, 512, 1000 respectively.
Mesh size
tens D and 11Ynof N(CGS'D of CGS

Comparison of parameters $\eta_n$, $\gamma_n$ of CGSD and $\bar{\eta}_n$, $\bar{\gamma}_n$ of NICGSD

methods for the 2D model problem.
FIG. 6.2 : Behaviour of parameters $\phi_n$, $\delta_n$ of CG3DSD method model problem.
FIG. 6.3: Behaviour of parameters $\tilde{e}_n$, $\gamma_n$ of NICG3DSD method for the 3D model problem.
Mesh size of NICGED3D method for the 3D model problem.
6.4 A SEMI-EMPIRICAL PROCEDURE FOR THE NORMALIZED CONJUGATE GRADIENT SECOND DEGREE METHODS

A thorough examination of the values of the parameters $\tilde{\gamma}_i, \tilde{\gamma}_i$, $i \in [1, n]$ (see Figures 6.5, 6.6) reveals that after a certain number of iterations the oscillating in above parameters subsides and the values remain almost stable.

Given that the values of parameters $\tilde{\rho}_n, \tilde{\gamma}_n$ are known (see Tables 6.1, 6.2), we assume now that after a certain number $k_A$ of iterations the iterative scheme proceeds with the values of parameters $\tilde{\rho}_i, \tilde{\gamma}_i$ constant and equal to $\tilde{\rho}_n, \tilde{\gamma}_n$ respectively, i.e., $\tilde{\rho}_i = \tilde{\rho}_n, \tilde{\gamma}_i = \tilde{\gamma}_n$, $i = k, k+1, k+2, \ldots$ (Case A).

Since the previous outlined procedure pre-assumes the knowledge of parameters $\tilde{\rho}_n, \tilde{\gamma}_n$, we consider now a different approach, in which the values of $\tilde{\rho}_i, \tilde{\gamma}_i$ are retained constant after the first $k_B$-iterations, throughout the iterative process, and are equal to $\tilde{\rho}_k, \tilde{\gamma}_k$ respectively, i.e., $\tilde{\rho}_i = \tilde{\rho}_k, \tilde{\gamma}_i = \tilde{\gamma}_k$, $i = k, k+1, k+2, \ldots$ (Case B).

Since the formulation of the scalar products into the inner loop of the methods is an important time-consuming operation, substantial gains in the computational work involved can be obtained if these scalar products are not calculated at all after $(n-k_A)$ or $(n-k_B)$ iterations respectively.

With this in mind, we derive a semi-Empirical Procedure in two and three space dimensions (henceforth called N.I.C.G.S.D.-S.E.P. and N.I.C.G.3D.S.D.-S.E.P. methods respectively) giving a tentative proposal for the numbers $k_A, k_B$ of initial iterations required for the parameters $\tilde{\rho}_i, \tilde{\gamma}_i$ to settle down, provided that the number of iterations of N.I.C.G.S.D.-S.E.P., N.I.C.G.S.D. and N.I.C.G.3D.S.D.-S.E.P., N.I.C.G.3D.S.D. methods is about the same.
At this point we indicate the following case worth further investigation. The determination of $k_A$ and $k_B$ such that the obtained number of iterations of N.I.C.G.S.D.-S.E.P., N.I.C.G.3D.S.D.-S.E.P. methods will be the minimum.

Experimental results for case A and case B are given in Tables 6.3, 6.4.
Graphs illustrating the performance of NICGSD method for different values of \( r \). The plots show the convergence behavior for a specific problem with a matrix of order \( N = 1521 \) (Grid 39x39).
FIG. 6 : Behaviour of parameters $\tilde{\xi}_i, \tilde{\eta}_i$ of NICG3DSD $[A = D_3^T T_3 D_3]$ method applied to the Laplacian matrix of order $N=1000$, the 3D model problem.
<table>
<thead>
<tr>
<th>Mesh Size $\varepsilon$</th>
<th>Case A</th>
<th>Case B</th>
<th>Percentage of gains in C.W.($\ast$)</th>
<th>Percentage of gains in C.W.($\ast$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h^{-1}$</td>
<td>$k_A$</td>
<td>$k_B$</td>
<td>N.I.C.G.S.D.-S.E.P. (r=4)</td>
<td>N.I.C.G.S.D.-S.E.P. (r=4)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$10^{-6}$</td>
<td>5</td>
<td>2</td>
<td>5$^*$</td>
<td>5$^*$</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>2</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>3</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>3</td>
<td>9</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>4</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>$10^{-8}$</td>
<td>5</td>
<td>4</td>
<td>7$^*$</td>
<td>6$^*$</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>4</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>6</td>
<td>10</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>6</td>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>8</td>
<td>16</td>
<td>15</td>
</tr>
<tr>
<td>$10^{-10}$</td>
<td>5</td>
<td>6</td>
<td>8$^*$</td>
<td>8$^*$</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>6</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>9</td>
<td>12</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>9</td>
<td>16</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>12</td>
<td>18</td>
<td>18</td>
</tr>
</tbody>
</table>

TABLE 6.3

Computational work required to reduce the error to 1E-6,1E-8,1E-10 respectively for the 2D-model problem.

Case A: $\tilde{\gamma}_i = \tilde{\gamma}_n$, $i=k_A, k_A+1, k_A+2, \ldots$.
Case B: $\tilde{\gamma}_i = \tilde{\gamma}_k$, $i=k_B, k_B+1, k_B+2, \ldots$.

($\ast$) The percentage of the gain in computational work (C.W.) is given when the scalar products in the inner loop of the methods, are not calculated after the (n-$k_A$) or (n-$k_B$) iterations respectively.

The resulting sparse matrices for the above considered mesh sizes are of the order 16,81,196,361,841 respectively.

($\ast$) The fill-in parameter $r$ has been taken to be $r=3$. 

142
<table>
<thead>
<tr>
<th>Mesh Size (k_a^2)</th>
<th>(r_1=r_2=1)</th>
<th>(r_1=r_2=4)</th>
<th>Percentage of gains in CW</th>
<th>(k_b)</th>
<th>(r_1=r_2=1)</th>
<th>(r_1=r_2=4)</th>
<th>Percentage of gains in CW</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\varepsilon=10^{-6})</td>
<td>5</td>
<td>2</td>
<td>6</td>
<td>6</td>
<td>66.67</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>7</td>
<td>8</td>
<td>87.50</td>
<td>1</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>8</td>
<td>9</td>
<td>77.78</td>
<td>1</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>(\varepsilon=10^{-8})</td>
<td>5</td>
<td>3</td>
<td>8</td>
<td>8</td>
<td>62.50</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>10</td>
<td>11</td>
<td>90.91</td>
<td>1</td>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>13</td>
<td>13</td>
<td>76.92</td>
<td>2</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>(\varepsilon=10^{-10})</td>
<td>5</td>
<td>4</td>
<td>10</td>
<td>11</td>
<td>63.64</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>13</td>
<td>13</td>
<td>92.31</td>
<td>1</td>
<td>9</td>
<td>11</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>12</td>
<td>16</td>
<td>75.00</td>
<td>3</td>
<td>10</td>
<td>13</td>
</tr>
<tr>
<td>(\varepsilon=10^{-6})</td>
<td>(k_b)</td>
<td>(r_1=r_2=1)</td>
<td>(r_1=r_2=4)</td>
<td>Percentage of gains in CW</td>
<td>(k_b)</td>
<td>(r_1=r_2=1)</td>
<td>(r_1=r_2=4)</td>
</tr>
<tr>
<td>(\varepsilon=10^{-8})</td>
<td>5</td>
<td>2</td>
<td>6</td>
<td>6</td>
<td>66.67</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>7</td>
<td>8</td>
<td>75.00</td>
<td>1</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>7</td>
<td>9</td>
<td>77.78</td>
<td>1</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>(\varepsilon=10^{-10})</td>
<td>5</td>
<td>2</td>
<td>9</td>
<td>8</td>
<td>75.00</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>11</td>
<td>11</td>
<td>81.82</td>
<td>2</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>9</td>
<td>13</td>
<td>76.92</td>
<td>2</td>
<td>8</td>
<td>11</td>
</tr>
<tr>
<td>(\varepsilon=10^{-10})</td>
<td>5</td>
<td>2</td>
<td>12</td>
<td>11</td>
<td>81.82</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>14</td>
<td>13</td>
<td>84.62</td>
<td>3</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>11</td>
<td>16</td>
<td>75.00</td>
<td>3</td>
<td>11</td>
<td>13</td>
</tr>
</tbody>
</table>

**TABLE 6.4**

Computational work required to reduce the error to 1E-6, 1E-8, 1E-10 respectively for the 3D-model problem in cases A and B. The percentage of the gain in computational work (C.W.) is given as in Table 6.3. The resulting sparse matrices for the above considered mesh sizes are of the order 64, 125, 512 respectively.

---

143
Finally, we state that the determination of the optimum size of the subset of parameters is an interesting and open question and further work on this topic is planned at a future date.

6.5 DERIVATION OF STANDARD NORMALIZED IMPLICIT METHODS AND STANDARD IMPLICIT METHODS

6.5.a Derivation of standard Normalized Implicit Methods

We consider the approximate factorization (4.3.1) applied to the linear system (2.1.3) to yield the iteration

\[(DSTsTsDs)u_{i+1} = (DSTsTsDs)u_i + a(s-Au_i),\]

\[i=0,1,2,\ldots\] (6.5.1)

where \(a\) is a predetermined acceleration parameter.

It can be easily seen from (2.3.13), (2.3.25), (2.3.26) and (6.5.1) that the error vector \(e_i\) satisfies:

\[e_{i+1} = E, e_i,\] (6.5.2)

where \(E\) is the error propagation matrix, given by

\[E = I - a(DSsTsTsDs)^{-1}A.\] (6.5.3)

The iteration (6.5.1) can be alternatively written as

\[\tilde{u}_{i+1} = \tilde{u}_i + a. r^*_i\] (6.5.4)

where

\[\tilde{u}_i = Ds_s u_i\] and \[r^*_i = (Ds_sTsT)^{-1}r_i\] (6.5.5)

or equivalently as

\[T_sTs\tilde{u}_{i+1} = a. \tilde{r}_i\] (6.5.6)

where

\[\delta\tilde{u}_{i+1} = \tilde{u}_{i+1} - \tilde{u}_i\] and \[\tilde{r}_i = D_s^{-1}r_i.\] (6.5.7)

The iterative scheme (6.5.4) or (6.5.6), is an analogue of the Simultaneous Displacement Iteration given by (2.3.13) and defined as the 'Normalized Implicit Simultaneous Displacement' (N.I.S.D.) method.
We consider now the following linear stationary iteration of second degree:

\[(D_s T'_s D_s) u_{i+1} = (D_s T'_s D_s) u_i + a (s - A u_i) + \beta (u_i - u_{i-1}) \]  

(6.5.8)

which can be written as

\[ \tilde{u}_{i+1} = \tilde{u}_i + a \tilde{r}_i + (T'_s T_s)^{-1} \beta (\tilde{u}_i - \tilde{u}_{i-1}) \]  

(6.5.9)

where \( \tilde{u}_{i+1} \), \( \tilde{r}_i \) are given by (6.5.5), or equivalently,

\[ T'_s T_s \delta \tilde{u}_{i+1} = a \tilde{r}_i + \tilde{\beta} \delta \tilde{u}_i \]  

(6.5.10)

where \( \delta \tilde{u}_{i+1} \), \( \tilde{r}_i \) are given by (6.5.7) and \( a, \tilde{\beta} \) are preconditioned acceleration parameters.

The iterative scheme (6.5.9) or (6.5.10) is an analogue of the second order Richardson's method given by (2.3.50) and is defined as the 'Normalized Implicit Richardson's' (N.I.R.) method. The parameters \( a, \tilde{\beta} \) remain constant throughout the iteration and are chosen to provide maximum convergence to the solution (see relationships (2.3.58)).

Consider now the iterative scheme (6.5.8), with the parameter sequences \( a_n, \beta_n \) instead of the fixed value parameters \( a, \beta \). Then the 'Normalized Implicit Chebychev' (N.I.Ch.) method, analogue of Chebychev second order method given by (2.3.60), is a non-stationary second degree iteration defined by

\[ \tilde{u}_{i+1} = \tilde{u}_i + a_n \tilde{r}_i + (T'_s T_s)^{-1} \beta_n (\tilde{u}_i - \tilde{u}_{i-1}) \]  

(6.5.11)

where \( \tilde{u}_{i+1} \), \( \tilde{r}_i \) are given by (6.5.5), or equivalently,

\[ (T'_s T_s) \delta \tilde{u}_{i+1} = a_n \tilde{r}_i + \beta_n \delta \tilde{u}_i \]  

(6.5.12)

where \( \delta \tilde{u}_{i+1} \), \( \tilde{r}_i \) are given by (6.5.7) and \( a_n, \beta_n \) are the sequence of acceleration parameters which vary with each iterative step (see relationships (2.3.61)).
The analysis and convergence conditions of the derived N.I.S.D., N.I.R., N.I.Ch. methods can be easily obtained from section 2.3, using properly the corresponding modified iteration matrix for each of the normalized implicit methods. It can be shown that since the parameter sequences are similar to those given in section 2.3 then the N.I.Ch. method is less sensitive to round-off errors than the N.I.R. method.

6.5. Derivation of Standard Implicit Methods

Several iterative procedures for solving non-symmetric linear systems have appeared in the literature, [25], [60], [36], [16], [40].

We now introduce a class of iterative procedures for solving numerically the linear system (2.7.17) associated with the P.D.E. (2.5.1). We consider the approximate factorization (4.2.1) applied to the above linear system to yield the following iterative method (Simultaneous Displacement)

\[ \text{L}_s \text{U}_s \delta u_{i+1} = a r_i \]  \hspace{1cm} (6.5.13)

where \( L_s \) and \( U_s \) are the known triangular matrices given by (4.2.2), \( a \) is a predetermined acceleration parameter, \( r_i = s - A u_i \) and \( \delta u_{i+1} = u_{i+1} - u_i \).

For a fixed choice of parameter \( a \) and an initial guess of the solution \( u_0 \), a sequence of approximate solutions \( u_1, u_2, \ldots, u_k \) can be obtained using the ALUBOT algorithm.

Similarly the following second order iterative procedure can be formulated for the Richardson and Chebychev methods respectively,

\[ \text{L}_s \text{U}_s \delta u_{i+1} = a r_i + \beta \delta u_{i-1} \]  \hspace{1cm} (6.5.14)

and

\[ \text{L}_s \text{U}_s \delta u_{i+1} = a r_i + \beta \delta u_{n-i} \]  \hspace{1cm} (6.5.15)

Note that for the parabolic problem of section 2.5 since the solution only differs slightly from plane to plane then the obtained solution at the \( k \)th time step is a good initial approximation to commence the iterative scheme (6.5.13) at the \((k+1)\)th time step.
Remark

Since the problem under investigation has been solved in many different ways a complete knowledge of the eigenspectrum of the iteration matrix was known.

Hence, for the Chebychev method, guesses which were close to the actual values were used.

For problems without this a priori knowledge then an attempt to calculate the spectral radius of $(L_s U_s)^{-1} L U$ (or the spectral radius of $(D_s T_s' T_s D_s)^{-1} D T' T D$ for the case of section 6.5.a) must be made.

Computational Work and Numerical Experiments

Numerical results for the Normalized Implicit methods i.e., N.I.S.D., N.I.R. and N.I.Ch. methods, on the 2D-model problem and the experimental estimation of the optimal iterative parameters involved, are given in Table 6.5.

For comparative reasons the numerical results obtained applying the ALUBOT-2D algorithm in conjunction with the Simultaneous Displacement, Second Order Richardson and Chebychev methods, on the 2D-model problem, are presented and the values of the optimal iterative parameters involved are experimentally estimated.

The initial vector was chosen to be the zero vector and the solution vector was chosen: $x_i = 0$, $i[e[1,N]$ and $x_{m+1} = 1$, where $m$ is the bandwidth of the coefficient matrix of linear systems (2.4.1). The right hand side vector of (2.4.1) was obtained as the product of the solution vector $x$ with the coefficient matrix $A$ of (2.4.1). The error was computed as

$$ \text{error} = \left\| \frac{x_i^{(k+1)} - x_i}{x_{m+1}^{(k+1)}} \right\|_\infty $$

and the iteration process terminated when the error was reduced to $10^{-6}$.
The cases 1 and 2 of the Normalized Implicit methods are referred to as the approximate factorizations $A_s = D T'_s T'_s D$ and $A_s = D T'_s T'_s D$ respectively.

Note that the calculation of the "true" residual i.e., $r_i = s_i - A u_i$, into the inner loop of the Normalized Implicit (N.I.S.D., N.I.R., N.I.Ch.) methods, requires the normalization of the solution vector $u_i$ at each iterative step.

Consequently, the above methods as far as the computational work is concerned, are unfavourable when compared with the N.I.C.G. procedure, since in the latter method the residuals are computed recursively (see section 5.4).
<table>
<thead>
<tr>
<th>Method</th>
<th>Mesh Size $h^i$</th>
<th>Normalized Method (NOBAR-2D Algorithm)</th>
<th>Normalized Method (NOBAR-2D Algorithm)</th>
<th>LU-factorization (ALUBOT-2D Algorithm)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Case 1</td>
<td>Case 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Parameters: $\alpha$ $\beta$</td>
<td>Parameters: $\alpha$ $\beta$</td>
<td>Parameters: $\alpha$ $\beta$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No. of Iter.</td>
<td>No. of Iter.</td>
<td>No. of Iter.</td>
</tr>
<tr>
<td>Simultaneous Displacement</td>
<td>10</td>
<td>1.20 $\downarrow$ 0</td>
<td>11</td>
<td>1.13 $\downarrow$ 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15</td>
<td>1.32 $\downarrow$ 0</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>1.40 $\downarrow$ 0</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30</td>
<td>1.50 $\downarrow$ 0</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>2nd Order Richardson</td>
<td>10</td>
<td>1.13 $\downarrow$ 0.008</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15</td>
<td>1.24 $\downarrow$ 0.015</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>1.31 $\downarrow$ 0.022</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30</td>
<td>1.39 $\downarrow$ 0.030</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>2nd Order Chebychev</td>
<td>10</td>
<td>1.139 $\downarrow$ 0.016</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15</td>
<td>1.263 $\downarrow$ 0.030</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>1.338 $\downarrow$ 0.044</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30</td>
<td>1.405 $\downarrow$ 0.060</td>
<td>16</td>
</tr>
</tbody>
</table>

**TABLE 6.5**

The performance of Normalized Implicit (N.I.S.D., N.I.R., N.I.Ch.) methods and the application of ALUBOT-2D algorithm to Simultaneous Displacement, 2nd order Richardson and Chebychev methods for solving the 2D-model problem. The resulting sparse matrices for the above considered mesh sizes are of order 81,196,361,841, respectively.
CHAPTER 7

THREE DIMENSIONAL ELLIPTIC AND PARABOLIC PROBLEMS
7.1 INTRODUCTION

In this chapter, the application of the algorithmic procedures of Chapter 3 to three-dimensional Elliptic and Parabolic problems is presented with accompanying experimental results to verify the applicability and usefulness of the methods.

SECTION A: THE ELLIPTIC 3D-PROBLEM

7.2 STATEMENT OF THE PROBLEM AND FORMATION OF THE DIFFERENCE EQUATIONS

We consider the solution of the Laplace’s equation in three variables:

\[
\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} + \frac{\partial^2 U}{\partial z^2} = 0, \quad (x, y, z) \in \mathbb{R},
\]

where \( R \) is the unit cube as given in (2.8.3).

The following Dirichlet boundary conditions are considered in connection with the approximate solution of equation (7.2.1):

\[
U(0, y, z) = U(1, y, z) = 0, \quad 0 \leq y, z \leq 1 \quad (7.2.2)
\]

\[
U(x, 0, z) = U(x, 1, z) = \sin \pi x \sin \pi z, \quad 0 \leq x, z \leq 1 \quad (7.2.3)
\]

\[
U(x, y, 0) = U(x, y, 1) = 0, \quad 0 \leq x, y \leq 1. \quad (7.2.4)
\]

The unit cube is now covered by a three-dimensional grid, defined by:

\[
\tilde{R}_h = \{(i h_x, j h_y, k h_z): 0 \leq i, j, k \leq M\}
\]

(7.2.5)

where \( h_x, h_y, h_z \) are the lengths in the \( X, Y \) and \( Z \) directions respectively.

Let \( x = i h_x, \quad y = j h_y, \quad z = k h_z \) be the co-ordinates of \((M-1)^3\) internal grid points and assume that \( u_{i,j,k} \) denotes \( u(ih_x, jh_y, kh_z) \).

The fully implicit difference approximations to the partial derivatives of equation (7.2.1) i.e.,

\[
\left[ \frac{\partial^2 U}{\partial x^2} \right]_{(i,j,k)}^{(n+1)} = \frac{u_{i+1,j,k}^{(n+1)} - 2u_{i,j,k}^{(n+1)} + u_{i-1,j,k}^{(n+1)}}{h_x^2}
\]

(7.2.6)
and analogous expressions for $\frac{\partial^2 U}{\partial y^2}$, $\frac{\partial^2 U}{\partial z^2}$ are used to derive the following finite difference analogue to (7.2.1) at the point 

$$(i_{h_x}, j_{h_y}, k_{h_z})$$

$$u(n+1)_{i-1,j,k} - 2u^{(n+1)}_{i,j,k} + u^{(n+1)}_{i+1,j,k} + u^{(n+1)}_{i,j-1,k} - 2u^{(n+1)}_{i,j,k} + u^{(n+1)}_{i,j,k+1} + \frac{h_x^2}{2} u^{(n+1)}_{i,j,k} + \frac{h_y^2}{2} u^{(n+1)}_{i,j,k} + \frac{h_z^2}{2} u^{(n+1)}_{i,j,k} + R_L = 0 \quad (7.2.7)$$

where the local truncation error $R_L$ of (7.2.7), assuming that only the second order differences are retained is given by

$$R_L = \frac{-1}{12} \left[ \frac{h_x^2}{2} \frac{\partial^4 U}{\partial x^4} + h_y^2 \frac{\partial^4 U}{\partial y^4} + h_z^2 \frac{\partial^4 U}{\partial z^4} \right] (i,j,k) \quad (7.2.8)$$

Ordering the $(M-1)^3$ internal grid points of $\Omega_h$ with increasing values of $j$, then $i$, then $k$ (see Figure 2.4) the following system of equations can be obtained in matrix notation:

$$\Omega u^{(n+1)} = s \quad (7.2.9)$$

where $\Omega$ is a real, square, symmetric, seven-diagonal, sparse matrix of the following form:

$$\Omega = \begin{bmatrix} A_1 & -\Gamma \\ -\Gamma & A_2 & -\Gamma \\ & -\Gamma & A_3 \\ & & \ddots & \ddots & \ddots \\ & & \ddots & \ddots & -\Gamma \\ & & & -\Gamma & A_{m-1} \\ & & & & -\Gamma & A_m \end{bmatrix} \quad (7.2.10)$$

where $\Gamma$ is the following diagonal matrix of order $m^2 = (M-1)^2$

$$\Gamma = \text{diag} \left\{ \frac{1}{h_x^2}, \frac{1}{h_y^2}, \ldots, \frac{1}{h_z^2} \right\} \quad (7.2.11)$$

and $A_i$, $i \in [1,m]$ are matrices of order $m^2$, given by:
where $\Delta$ is the following diagonal matrix of order $m$.

$$\Delta = \text{diag} \left( \frac{1}{h_x^2}, \frac{1}{h_y^2}, \ldots, \frac{1}{h_y^2} \right)$$

and $B_i$, $i \in [1, m^2]$ are matrices of order $m$, given by:

$$B_i = \begin{bmatrix} C & -\theta \\ -\theta & C \end{bmatrix}, \quad i \in [1, m^2]$$

where $C = 2 \left[ \frac{1}{h_x^2} + \frac{1}{h_y^2} + \frac{1}{h_z^2} \right]$ and $\theta = \frac{1}{h_x^2}$. (7.2.15)

The solution vector $u^{(n+1)}$ and the right hand side vector $s$ of the system (7.2.9) are $(m^3 \times 1)$ column vectors. Note that the components of the latter vector depend on the boundary values of $u(x, y, z)$ at the grid points on the boundary planes of the unit cube.

### 7.3 Computational Results

The analytic solution of the problem can be verified by separation of variables and is given by

$$U(x, y, z) = \text{sech} \frac{\pi}{\sqrt{2}} \cdot \sin \pi x \cdot \cosh(\sqrt{2} \pi (y - \frac{1}{2})) \cdot \sin \pi z.$$  (7.3.1)
Values of \( U \) calculated from the theoretical solution (7.3.1) are given in Tables 7.1, 7.2, 7.3.

For a numerical solution, the implicit finite difference scheme defined in (7.2.7), was used as an approximation to the original Elliptic P.D.E. (7.2.1). Since the coefficient matrix \( A \) in (7.2.9) is positive definite, diagonally dominant and seven-diagonal, the resulting system (7.2.9) was solved using the LUBOF-3D Algorithm.

Numerical results are presented in the accompanying Tables 7.1, 7.2, 7.3, where the values of the approximate and analytic solution at the \((M-1)^3\) internal mesh points are given in groups of five decimal figures assuming that \( h_X = h_Y = h_Z = \frac{1}{M} \).

Since the solution is symmetric:

(i) about the planes which pass through the points \( x=0.5, \ y=0.5, \ z=0.5 \) and are parallel to the co-ordinate planes \((Y,Z),(X,Z),(X,Y)\) respectively and

(ii) about the plane which pass through the \((0.5,0.5,0.5)\) point and the co-ordinate axis \( Y \),

the values of the solution is given only for

\[
\begin{align*}
    x &\leq 0.5, \quad y \leq 0.5, \quad z \leq 0.5 \\
    z &\leq x.
\end{align*}
\]

The modulus, relative and maximum modulus errors for the test problem are given in Table 7.4 for mesh sizes \( M=4,5,6,7,8 \) respectively.
\begin{table} \centering \begin{tabular}{|c|c|c|c|} \hline \( \pm Z \) & \(+Y\) & \(X\) & 0.250 & 0.500 \\ \hline \hline 0.250 & 0.250 & 0.18039 & 68353 & 0.25511 & 96511 \\ & & 0.19677 & 51746 & 0.27828 & 21207 \\ & & 0.10719 & 18762 & 0.15159 & 22051 \\ & & 0.12408 & 68064 & 0.17548 & 52445 \\ 0.500 & & & & & \\ \hline 0.500 & 0.250 & 0.36079 & 36707 & & \\ & & 0.39355 & 03492 & & \\ & & 0.21438 & 37524 & & \\ & & 0.24817 & 36127 & & \\ \hline \end{tabular} \caption{TABLE 7.1 \ Analytic (+) and Computed (++) solution of the Elliptic 3D-problem for a grid (3×3×3).} \end{table}

\begin{table} \centering \begin{tabular}{|c|c|c|c|c|} \hline \( \pm Z \) & \(+Y\) & \(X\) & 0.167 & 0.333 & 0.500 \\ \hline \hline 0.167 & 0.167 & 0.12392 & 95735 & 0.21465 & 23179 \\ & & 0.12731 & 96573 & 0.22052 & 41152 \\ & & 0.06897 & 33511 & 0.11946 & 53485 \\ & & 0.07286 & 97233 & 0.12621 & 40457 \\ & & 0.05359 & 59381 & 0.09283 & 08878 \\ & & 0.05749 & 05309 & 0.09954 & 18795 \\ 0.333 & 0.167 & & & 0.37178 & 87206 \\ & & 0.38195 & 89719 & 0.42930 & 46358 \\ & & 0.20692 & 00533 & 0.44104 & 82305 \\ & & 0.21860 & 91399 & 0.23893 & 06970 \\ & & 0.16078 & 78143 & 0.25242 & 80915 \\ & & 0.17241 & 15928 & 0.18566 & 17757 \\ 0.500 & 0.333 & & & & \\ & & 0.49571 & 82941 & 0.49571 & 82941 \\ & & 0.50927 & 86291 & 0.50927 & 86291 \\ & & 0.27589 & 34045 & 0.27589 & 34045 \\ & & 0.29147 & 88531 & 0.29147 & 88531 \\ & & 0.21438 & 37523 & 0.21438 & 37523 \\ & & 0.22988 & 21237 & & \\ \hline \end{tabular} \caption{TABLE 7.2 \ Analytic (+) and Computed (++) solution of the Elliptic 3D-problem for a grid (5×5×5).} \end{table}
<table>
<thead>
<tr>
<th>$Z$</th>
<th>$Y$</th>
<th>0.125</th>
<th>0.250</th>
<th>0.375</th>
<th>0.500</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.125</td>
<td>0.125</td>
<td>0.08603</td>
<td>0.08702</td>
<td>0.15896</td>
<td>0.20769</td>
</tr>
<tr>
<td></td>
<td>0.250</td>
<td>0.05283</td>
<td>0.05410</td>
<td>0.09763</td>
<td>0.12755</td>
</tr>
<tr>
<td></td>
<td>0.375</td>
<td>0.03636</td>
<td>0.03766</td>
<td>0.06719</td>
<td>0.09092</td>
</tr>
<tr>
<td></td>
<td>0.500</td>
<td>0.03139</td>
<td>0.03268</td>
<td>0.06039</td>
<td>0.07891</td>
</tr>
<tr>
<td>0.250</td>
<td>0.125</td>
<td>0.05410</td>
<td>0.05283</td>
<td>0.09997</td>
<td>0.13063</td>
</tr>
<tr>
<td></td>
<td>0.250</td>
<td>0.05410</td>
<td>0.05283</td>
<td>0.09997</td>
<td>0.13063</td>
</tr>
<tr>
<td></td>
<td>0.375</td>
<td>0.03636</td>
<td>0.03766</td>
<td>0.06719</td>
<td>0.09092</td>
</tr>
<tr>
<td></td>
<td>0.500</td>
<td>0.03139</td>
<td>0.03268</td>
<td>0.06039</td>
<td>0.07891</td>
</tr>
<tr>
<td>0.375</td>
<td>0.125</td>
<td>0.05283</td>
<td>0.05410</td>
<td>0.09763</td>
<td>0.12755</td>
</tr>
<tr>
<td></td>
<td>0.250</td>
<td>0.05283</td>
<td>0.05410</td>
<td>0.09763</td>
<td>0.12755</td>
</tr>
<tr>
<td></td>
<td>0.375</td>
<td>0.03636</td>
<td>0.03766</td>
<td>0.06719</td>
<td>0.09092</td>
</tr>
<tr>
<td></td>
<td>0.500</td>
<td>0.03139</td>
<td>0.03268</td>
<td>0.06039</td>
<td>0.07891</td>
</tr>
<tr>
<td>0.500</td>
<td>0.125</td>
<td>0.05410</td>
<td>0.05283</td>
<td>0.09997</td>
<td>0.13063</td>
</tr>
<tr>
<td></td>
<td>0.250</td>
<td>0.05410</td>
<td>0.05283</td>
<td>0.09997</td>
<td>0.13063</td>
</tr>
<tr>
<td></td>
<td>0.375</td>
<td>0.03636</td>
<td>0.03766</td>
<td>0.06719</td>
<td>0.09092</td>
</tr>
<tr>
<td></td>
<td>0.500</td>
<td>0.03139</td>
<td>0.03268</td>
<td>0.06039</td>
<td>0.07891</td>
</tr>
</tbody>
</table>

**TABLE 7.3**

Analytic (+) and Computed (++) solution of the Elliptic 3D-problem for a grid (7×7×7).
<table>
<thead>
<tr>
<th>Grid</th>
<th>Modulus Error (+)</th>
<th>Relative Error (+)</th>
<th>Maximum Modulus Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>3x3x3</td>
<td>0.03378 98604</td>
<td>0.15761 39051</td>
<td>0.03378 98604</td>
</tr>
<tr>
<td>4x4x4</td>
<td>0.02003 10081</td>
<td>0.09388 04662</td>
<td>0.02003 10082</td>
</tr>
<tr>
<td>5x5x5</td>
<td>0.01549 83714</td>
<td>0.07229 26584</td>
<td>0.01558 54487</td>
</tr>
<tr>
<td>6x6x6</td>
<td>0.01093 21870</td>
<td>0.05105 74037</td>
<td>0.01093 21870</td>
</tr>
<tr>
<td>7x7x7</td>
<td>0.00882 26791</td>
<td>0.04115 36743</td>
<td>0.00882 46531</td>
</tr>
</tbody>
</table>

**TABLE 7.4**

The Modulus, Relative and Maximum Modulus Errors for mesh sizes M=4,5,6,7,8. The resulting sparse matrices are of order N=27,64,125,216,343 respectively.

(+) Note that the Modulus and Relative Errors are given at the centre (even mesh size) or at one of the eight grid points near the centre (odd mesh size) of the unit cube.

### SECTION B: THE PARABOLIC 3D-PROBLEM

#### 7.4 STATEMENT OF THE PROBLEM AND FORMATION OF THE DIFFERENCE EQUATIONS

We consider the solution of the boundary-value problem,

\[
\frac{\partial U}{\partial t} = a^2 \left( \frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} + \frac{\partial^2 U}{\partial z^2} \right), \quad (x,y,z) \in \Omega, \quad 0 < t < +\infty, \tag{7.4.1}
\]

where \(a\) is a known constant and \(\Omega\) is a cube of side \(\lambda\) \([0 < x, y, z < \lambda]\), subject to the following boundary conditions:

\[
U|_{x=0} = U|_{x=\lambda} = U|_{y=0} = U|_{y=\lambda} = U|_{z=0} = U|_{z=\lambda} = 0, \quad 0 < t < +\infty \tag{7.4.2}
\]
and initial condition

$$U|_{t=0} = \phi_0, \quad 0<x<\lambda, \quad 0<y<\lambda, \quad 0<x<\lambda. \quad (7.4.3)$$

With $\phi_0 = \text{constant}$, the parabolic P.D.E. (7.4.1) together with boundary and initial conditions (7.4.2), (7.4.3), describes the temperature of a cube, uniformly heated at the initial time $t=0$.

It is easily seen that equation (7.4.1) can be obtained from equation (2.8.1) of the Parabolic 3D-example given in section 2.8, when the coefficients of the latter equation are $A=C=E=a^2$ and $F=l$, the source term is $Q=0$, the initial condition $f=\phi_0$ and boundary conditions $g=0$. Then, proceeding in a similar manner as in section 2.8, we derive the following fully implicit finite difference discretization form of (7.4.1), i.e.,

$$\frac{U^{(\ell+1)}_{i,j,k}-U^{(\ell)}_{i,j,k}}{\Delta t} + O(\Delta t) = a^2 \left\{ \begin{array}{c}
U^{(\ell+1)}_{i-1,j,k} - 2U^{(\ell+1)}_{i,j,k} + U^{(\ell+1)}_{i+1,j,k} \\
U^{(\ell+1)}_{i,j-1,k} - 2U^{(\ell+1)}_{i,j,k} + U^{(\ell+1)}_{i,j+1,k} \\
\frac{h_x^2}{h_y^2} - \frac{h_y^2}{h_x^2} - \frac{h_z^2}{h_x^2 + h_y^2}
\end{array} \right\}$$

$$+ O(h_x^2 + h_y^2 + h_z^2) \quad (7.4.4)$$

where $U^{(\ell+1)}_{i,j,k}$ and $U^{(\ell)}_{i,j,k}$ are the values of the dependent variable at the $(\ell+1)$ and $\ell$ time levels respectively (see Figure 2.9).

Note that the volumetric grid system $R_h$, which covers the region $R_\lambda$, is now defined by

$$R_h = \{(i h_x, j h_y, k h_z, r z) : \begin{array}{l}
0 \leq i \leq M \\
0 \leq j \leq M \\
0 \leq k \leq M \\
0 \leq r \leq T/\Delta t
\end{array}\}.$$  

Then, grouping the system of finite difference equations (7.4.4) in matrix form ordering the $(M-1)^3$ internal mesh points as in section 2.8 (see also Figure 2.4), we obtain:
\[ \Omega \cdot u^{(\ell+1)} = u^{(\ell)} \]  
(7.4.5)

where \( \Omega \) is a real, square, seven-diagonal matrix of order \( N \), semibandwidths \( m \) and \( p \) and is given by (7.2.10), where \( \Gamma \) now is the following diagonal matrix of order \( m^2=(M-1)^2 \)

\[ \Gamma = \text{diag}\{ a_2^2 \Delta t \, h_x^{-2}, a_2^2 \Delta t \, h_y^{-2}, \ldots, a_2^2 \Delta t \, h_z^{-2} \}, \]  
(7.4.6)

and \( A_i, i \in [1,m] \) are matrices of order \( m^2 \) given by (7.2.12) where \( \Delta \), in this case, is given by the following diagonal matrix of order \( m \)

\[ \Delta = \text{diag}\{ a_2^2 \Delta t \, h_x^{-2}, a_2^2 \Delta t \, h_y^{-2}, \ldots, a_2^2 \Delta t \, h_z^{-2} \}, \]  
(7.4.7)

and \( B_i, i \in [1,m^2] \) are matrices of order \( m \) given by (7.2.14), where

\[ C = 1+2a_2^2 \left[ \Delta t \left( \frac{1}{h_x^2} + \frac{1}{h_y^2} + \frac{1}{h_z^2} \right) \right] \] and \( \theta = a_2^2 \Delta t \).  
(7.4.8)

Note that \( u^{(\ell+1)} \) and \( u^{(\ell)} \) of system (7.4.5) are respectively \([m^3 \times 1]\) column vectors, consisting of the unknown approximate solutions \( u_{i,j,k} \) at the \((\ell+1)\) time level and the known boundary values given by (7.4.2), plus the known approximate solutions \( u_{i,j,k} \) at the \( \ell \) time level.

Alternatively, the Crank-Nicolson implicit scheme can be derived and is given by

\[ u_{i,j,k}^{(\ell+1)} = u_{i,j,k}^{(\ell)} + \frac{2}{\Delta t} \left\{ \begin{array}{c}
U_{i-1,j,k}^{(\ell+1)} - 2U_{i,j,k}^{(\ell+1)} + U_{i+1,j,k}^{(\ell+1)} \\
U_{i,j-1,k}^{(\ell+1)} - 2U_{i,j,k}^{(\ell+1)} + U_{i,j+1,k}^{(\ell+1)} \\
U_{i,j,k-1}^{(\ell+1)} - 2U_{i,j,k}^{(\ell+1)} + U_{i,j,k+1}^{(\ell+1)} \\
U_{i,j,k}^{(\ell+1)} - 2U_{i,j,k}^{(\ell+1)} + U_{i,j,k}^{(\ell+1)} \\
0 \end{array} \right\} + O(h_x^2 + h_y^2 + h_z^2), \]  
(7.4.9)
Then, the system of finite difference equations so obtained can be expressed in matrix form, namely,

\[ \Omega_1 u^{(t+1)} = \Omega_2 u^{(t)} \quad , \]

(7.4.10)

with \( \Omega_1, \Omega_2 \) defined as follows:

\( \Omega_1 \) is of the same type and form of matrix \( \Omega \) (defined in (7.2.10)-(7.2.15)) and its expression can be obtained by replacing in (7.2.10) \( \Omega \) by \( \Omega_1 \).

In this case, \( \Gamma \) is given by the following diagonal matrix of order \( m^2 = (M-1)^2 \)

\[ \Gamma = \text{diag}\left\{ \frac{a^2 \Delta t}{h_x^2}, \frac{a^2 \Delta t}{h_y^2}, \ldots, \frac{a^2 \Delta t}{h_z^2} \right\} \]  

(7.4.11)

and \( A_i, i \in [1, m] \) are matrices of order \( m^2 \), given by (7.2.12) where

\( \Delta \), is given by the following diagonal matrix of order \( m \)

\[ \Delta = \text{diag}\left\{ \frac{a^2 \Delta t}{h_x^2}, \frac{a^2 \Delta t}{h_y^2}, \ldots, \frac{a^2 \Delta t}{h_z^2} \right\} \]  

(7.4.12)

and \( B_i, i \in [1, m^2] \) are matrices of order \( m \), given by (7.2.14) where

\[ C = 1 + a^2 \left[ \Delta t \left( \frac{1}{h_x^2} + \frac{1}{h_y^2} + \frac{1}{h_z^2} \right) \right] \quad \text{and} \quad \theta = \frac{a^2 \Delta t}{h_x^2} \]  

(7.4.13)

Similarly, \( \Omega_2 \) is of the same form and type as \( \Omega_1, \Omega \) matrices and its expression can be obtained by replacing \( \Omega \) by \( \Omega_2 \) in (7.2.10).

Then, \( \Gamma \) is given by the diagonal matrix of order \( m^2 = (M-1)^2 \)

\[ \Gamma = \text{diag}\left\{ -\frac{a^2 \Delta t}{h_x^2}, -\frac{a^2 \Delta t}{h_y^2}, \ldots, -\frac{a^2 \Delta t}{h_z^2} \right\} \]  

(7.4.14)

and \( A_i, i \in [1, M] \) are matrices of order \( m^2 \), given by (7.2.12), where

\( \Delta \), is given by the diagonal matrix of order \( m \):

\[ \Delta = \text{diag}\left\{ -\frac{a^2 \Delta t}{h_x^2}, -\frac{a^2 \Delta t}{h_y^2}, \ldots, -\frac{a^2 \Delta t}{h_z^2} \right\} \]  

(7.4.15)

and \( B_i, i \in [1, m^2] \) are matrices of order \( m \), given by (7.2.14) where

\[ C = 1 - a^2 \left[ \Delta t \left( \frac{1}{h_x^2} + \frac{1}{h_y^2} + \frac{1}{h_z^2} \right) \right] \quad \text{and} \quad \theta = -\frac{a^2 \Delta t}{h_x^2} \]  

(7.4.16)
The analytic solution of this problem can be verified [8] by separation of variables and is given by:

\[ U(x, y, z, t) = \left( \frac{4}{\pi} \right)^3 \phi_0 \sum_{k, m, n=0}^{+\infty} \frac{A_{k, m, n}}{(2k+1)(2m+1)(2n+1)} \cdot \]

\[ \cdot \sin \left( \frac{(2k+1)\pi x}{\lambda} \right) \cdot \sin \left( \frac{(2m+1)\pi y}{\lambda} \right) \cdot \sin \left( \frac{(2n+1)\pi z}{\lambda} \right) \]

\[ - \frac{a^2}{\lambda^2} \left[ \frac{1}{((2k+1)^2+(2m+1)^2+(2n+1)^2)} \right] t \]

where \( A_{k, m, n} = e^{-a^2(2k+1)^2+2m+1)^2+(2n+1)^2} t \) \( (7.5.1) \)

The time at which a steady-state solution will occur at the centre of the cube with relative accuracy \( \epsilon > 0 \), (where \( \epsilon \) is an arbitrary positive quantity) can be easily seen to be:

\[ U(\frac{a}{2}, \frac{a}{2}, \frac{a}{2}, t) = \left( \frac{4}{\pi} \right)^3 \phi_0 \sum_{k=0}^{+\infty} \frac{(-1)^k e^{-\frac{a^2(2k+1)^2}{\lambda^2}}}{(2k+1)} . t \]

\[ (7.5.2) \]

Since the series on the right hand side of (7.5.3) satisfies the conditions of Leibnitz's theorem on alternating series, then the remainder of this series does not exceed the value of the first of the residual terms in absolute value, i.e.,

\[ \left| R_n \left( \frac{\lambda}{2}, \frac{\lambda}{2}, \frac{\lambda}{2}, t \right) \right| = \left| \left( \frac{4}{\pi} \right)^3 \phi_0 \sum_{k=n+1}^{+\infty} \frac{(-1)^k}{(2k+1)} e^{-\frac{a^2(2k+1)^2}{\lambda^2}} . t \right| \leq \left| \frac{-\frac{a^2}{\lambda^2} \left[ \frac{1}{((2k+1)^2+(2m+1)^2+(2n+1)^2)} \right] t}{(2k+1)} \right| \]

\[ \leq \left( \frac{4}{\pi} \right)^3 \phi_0 e^{-\frac{a^2(2n+3)^2}{\lambda^2}} . t \]

\[ (7.5.4) \]

The ratio of the sum of all terms of the series (7.5.3), excluding the first and the first term of the same series enables us to estimate the time taken for the steady-state solution, i.e.,

\[ \frac{1}{3} \frac{8\pi a^2}{\lambda^2} . t \]

\[ (7.5.5) \]
where $\tilde{\varepsilon}>0$ is an arbitrary positive quantity.

From (7.5.5) now, we can easily obtain

$$t \geq t^* = -\frac{\lambda^2}{8\pi a^2} \ln 3\tilde{\varepsilon}.$$  (7.5.6)

Then, for all $t$ satisfying the inequality (7.5.6), where $\tilde{\varepsilon}$ is less than the least of the numbers 1 and $\varepsilon/9$, it is given in [8] that a steady-state solution will occur at the centre of the cube with relative accuracy $\varepsilon>0$.

### 7.6 COMPUTATIONAL RESULTS

For a numerical solution, the implicit finite difference schemes defined in (7.4.4), (7.4.9) were used as an approximation to the original P.D.E. (7.4.1), subject to the conditions (7.4.2), (7.4.3) and the obtained systems (7.4.5), (7.4.10) respectively were solved using the LUBOT-3D Algorithm.

For simplicity, without loss of generality, we choose $a=1$, $\phi_0=1$ and $h_x=h_y=h_z=M$. The side $\lambda$ of the cube was fixed $\lambda=1$ and a relative accuracy $\varepsilon=10^{-6}$ was used.

The time step $\Delta t$ is chosen such that $\Delta t=h_x^2$, $\Delta t=h_x^3$ respectively and the values of the theoretical solution $U(x,y,z,t)$ are obtained from (7.5.1), where the first ten terms of the infinite series have been kept.

Numerical results are presented in the accompanying Tables 7.5-7.12, where the values of the analytic and approximate (for both implicit finite difference schemes given in (7.4.4), (7.4.9)) at the internal mesh points of a $(7 \times 7 \times 7)$ grid are given in groups of five decimal figures, for the first four time steps.

Since a great deal of symmetry is involved [specifically the solution is symmetric besides the cases (i), (ii) of section 7.3 and
(iii) about the plane passing through the points 

\((0,0,0), (0.5,0.5,0), (0.5,0.5,0.5)\)

the values of the approximate and analytic solution are given only for 

\[x \leq 0.5, \quad y \leq 0.5, \quad z \leq 0.5,\]  \hspace{1cm} (7.6.1a)

with 

\[z \leq x \quad \text{and} \quad y \geq x.\]  \hspace{1cm} (7.6.1b)

The modulus and maximum modulus error for the test problem at a grid point, after a number of time steps, are given in Tables 7.13, 7.14.
<table>
<thead>
<tr>
<th>Z</th>
<th>X</th>
<th>Y</th>
<th>0.125</th>
<th>0.250</th>
<th>0.375</th>
<th>0.500</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.125</td>
<td>0.1410</td>
<td>13288 (AS)</td>
<td>0.03317 48376 (FI)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.250</td>
<td>0.22829</td>
<td>79664</td>
<td>0.36961</td>
<td>0.2851</td>
<td>0.09094 44228</td>
<td>0.35710 80484</td>
</tr>
<tr>
<td>0.375</td>
<td>0.26162</td>
<td>63273</td>
<td>0.42356</td>
<td>82997</td>
<td>0.48540</td>
<td>34416</td>
</tr>
<tr>
<td>0.500</td>
<td>0.26838</td>
<td>47727</td>
<td>0.43451</td>
<td>01006</td>
<td>0.49794</td>
<td>25949</td>
</tr>
</tbody>
</table>

| 0.250 | 0.59839 | 23773 | 0.46262 94953 |
| 0.375 | 0.68574 | 94284 | 0.78585 94068 |
| 0.500 | 0.70346 | 40067 | 0.80616 00696 | 0.82698 51481 |

| 0.375 | 0.90058 | 40639 | 0.74397 05864 |
| 0.500 | 0.92384 | 83443 | 0.94771 35976 |

| 0.500 | 0.97219 | 53484 | 0.82648 39111 |

**TABLE 7.5**

Analytic (AS) and Computed (FI) [using the Fully Implicit Finite Difference Scheme] solutions of the parabolic 3D-problem for a grid (7x7x7) at the first time step [\( \Delta t = h_x^2 = \left(\frac{1}{8}\right)^2 \)].
<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
<th>0.125</th>
<th>0.250</th>
<th>0.375</th>
<th>0.500</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.125</td>
<td>0.05594 74205 (^{(AS)})</td>
<td>0.03336 05104 (^{(FI)})</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.250</td>
<td>0.09946 93797</td>
<td>0.17864 74296</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.375</td>
<td>0.12491 86973</td>
<td>0.22209 39810</td>
<td>0.27891 68975</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.500</td>
<td>0.13298 89822</td>
<td>0.23640 66489</td>
<td>0.29689 14725</td>
<td>0.31602 44046</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.250</td>
<td>0.31441 85019</td>
<td>0.26743 47779</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.375</td>
<td>0.39486 27183</td>
<td>0.49588 86498</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.500</td>
<td>0.42030 93285</td>
<td>0.52784 57947</td>
<td>0.56186 23921</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.375</td>
<td>0.62276 21437</td>
<td>0.52196 79210</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.500</td>
<td>0.66289 55489</td>
<td>0.70561 53191</td>
<td>0.59680 89702</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TABLE 7.6**

Analytic (AS) and Computed (FI) [using the Fully Implicit, Finite Difference Scheme] solutions of the parabolic 3D-problem for a grid (7x7x7) at the second time step \([\Delta t = h^2_x = (\frac{1}{8})^2]\).
### Table 7.7

Analytic (AS) and Computed (FI) [using the fully Implicit Finite Difference Scheme] solutions of the parabolic 3D-problem for a grid (7x7x7) at the third time step \( [\Delta t= h^2_x= \left(\frac{1}{8}\right)^2] \).
### TABLE 7.8

Analytic (AS) and Computed (FI) [using the Fully Implicit Finite Difference Scheme] solutions of the parabolic 3D-problem for a grid (7x7x7) at the fourth time step $[\Delta t=\Delta h^2_x=(\frac{1}{8})^2]$
<table>
<thead>
<tr>
<th>tZ</th>
<th>tY</th>
<th>tX</th>
<th>0.125</th>
<th>0.250</th>
<th>0.375</th>
<th>0.500</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.125</td>
<td>0.250</td>
<td>0.375</td>
<td>0.500</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.125</td>
<td>0.90874 07215 (AS)</td>
<td>0.01025 41252 (FI)</td>
<td>0.00829 74626 (CN)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.250</td>
<td>0.93369 72325</td>
<td>0.95933 91177</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.125</td>
<td>0.06351 49105</td>
<td>0.75620 45480</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.375</td>
<td>0.93354 95054</td>
<td>0.95918 73336</td>
<td>0.95903 55735</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.250</td>
<td>0.06351 49105</td>
<td>0.75620 45480</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.125</td>
<td>0.08655 01636</td>
<td>0.70657 64549</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.500</td>
<td>0.94602 33010</td>
<td>0.97200 36940</td>
<td>0.97184 99061</td>
<td>0.98483 54596</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.375</td>
<td>0.93462 40646</td>
<td>0.81638 67604</td>
<td>0.88469 20535</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.250</td>
<td>0.09427 26837</td>
<td>0.78891 46167</td>
<td>0.87845 11729</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.125</td>
<td>0.07564 11583</td>
<td>0.82124 41023</td>
<td>0.89042 88769</td>
<td>0.89628 54085</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.500</td>
<td>0.9480 95660</td>
<td>0.79291 04586</td>
<td>0.88293 71840</td>
<td>0.88746 62005</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.250</td>
<td>0.98568 51994</td>
<td>0.982306 03300</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.375</td>
<td>0.98552 92470</td>
<td>0.98537 33192</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.250</td>
<td>0.79375 24478</td>
<td>0.97205 74746</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.125</td>
<td>0.89264 98765</td>
<td>0.98222 22015</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.500</td>
<td>0.98391 07969</td>
<td>0.99853 95684</td>
<td>1.01188 17408</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.375</td>
<td>0.98869 75796</td>
<td>0.97909 06592</td>
<td>0.98628 02112</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.250</td>
<td>0.89857 38761</td>
<td>0.98733 90425</td>
<td>0.99250 76754</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.125</td>
<td>0.88845 41062</td>
<td>0.98733 90425</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.500</td>
<td>0.98521 74160</td>
<td>0.98063 15189</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.375</td>
<td>0.98521 74160</td>
<td>0.98793 75819</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.250</td>
<td>0.98863 15821</td>
<td>0.99938 56821</td>
<td>1.01172 16438</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.125</td>
<td>0.98787 13157</td>
<td>0.99527 49114</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.500</td>
<td>0.99311 55662</td>
<td>0.99834 64881</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.375</td>
<td>0.99311 55662</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.500</td>
<td>0.02523 99508</td>
<td>0.99608 32861</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.500</td>
<td>0.99865 27983</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TABLE 7.9**

Analytic (AS) and Computed [using the Fully Implicit (FI) and the Crank-Nicolson (CN) Implicit Schemes respectively] solutions of the parabolic 3D-problem for a grid (7x7x7) at the first time step [$\Delta t = h^3 = (\frac{1}{8})^3$].
<table>
<thead>
<tr>
<th>ΔZ</th>
<th>ΔY</th>
<th>Δx</th>
<th>0.125</th>
<th>0.250</th>
<th>0.375</th>
<th>0.500</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.125</td>
<td>0.60088 50413 (AS)</td>
<td>0.02193 14898 (FI)</td>
<td>0.01624 86965 (CN)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.250</td>
<td>0.70827 78579</td>
<td>0.83486 43907</td>
<td>0.59292 95464</td>
<td>0.10268 09142</td>
<td>0.62124 65962</td>
<td></td>
</tr>
<tr>
<td>0.375</td>
<td>0.71173 45315</td>
<td>0.83893 88563</td>
<td>0.84303 32068</td>
<td>0.78255 65820</td>
<td>0.82322 70364</td>
<td></td>
</tr>
<tr>
<td>0.500</td>
<td>0.71276 07887</td>
<td>0.84014 85306</td>
<td>0.84424 87849</td>
<td>0.84546 61157</td>
<td>0.80801 71995</td>
<td></td>
</tr>
<tr>
<td>0.125</td>
<td>0.01268 09142</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.250</td>
<td>0.11717 92654</td>
<td>0.67967 25275</td>
<td>0.79509 89785</td>
<td>0.11737 71774</td>
<td>0.71558 31892</td>
<td></td>
</tr>
<tr>
<td>0.375</td>
<td>0.11737 71774</td>
<td>0.67967 25275</td>
<td>0.79509 89785</td>
<td>0.11789 76757</td>
<td>0.68994 18860</td>
<td></td>
</tr>
<tr>
<td>0.500</td>
<td>0.11789 76757</td>
<td>0.68994 18860</td>
<td>0.79509 89785</td>
<td>0.11989 76757</td>
<td>0.72331 78770</td>
<td></td>
</tr>
<tr>
<td>0.250</td>
<td>0.098407 50252</td>
<td>0.69521 55716</td>
<td>0.72567 36314</td>
<td>0.098407 50252</td>
<td>0.69521 55716</td>
<td></td>
</tr>
<tr>
<td>0.375</td>
<td>0.98887 76971</td>
<td>0.99370 38080</td>
<td>0.99615 20884</td>
<td>0.98887 76971</td>
<td>0.99370 38080</td>
<td></td>
</tr>
<tr>
<td>0.500</td>
<td>0.99030 35698</td>
<td>0.99513 66396</td>
<td>0.99657 15372</td>
<td>0.99030 35698</td>
<td>0.99513 66396</td>
<td></td>
</tr>
<tr>
<td>0.375</td>
<td>0.9855 34722</td>
<td>0.94976 86238</td>
<td>0.97270 91668</td>
<td>0.99999 32965</td>
<td>1.00143 51969</td>
<td></td>
</tr>
<tr>
<td>0.500</td>
<td>0.99030 35698</td>
<td>0.99513 66396</td>
<td>0.99657 15372</td>
<td>0.99030 35698</td>
<td>0.99513 66396</td>
<td></td>
</tr>
<tr>
<td>0.375</td>
<td>0.9855 34722</td>
<td>0.94976 86238</td>
<td>0.97270 91668</td>
<td>0.99999 32965</td>
<td>1.00143 51969</td>
<td></td>
</tr>
<tr>
<td>0.500</td>
<td>0.99030 35698</td>
<td>0.99513 66396</td>
<td>0.99657 15372</td>
<td>0.99030 35698</td>
<td>0.99513 66396</td>
<td></td>
</tr>
</tbody>
</table>

**TABLE 7.10**

Analytic (AS) and Computed [using the Fully Implicit (FI) and the Crank-Nicolson (CN) Implicit Schemes respectively] solutions of the parabolic 3D-problem for a grid (7x7x7) at the second time step \([Δt=\frac{3}{8}=(\frac{1}{8})^3]\).
<table>
<thead>
<tr>
<th>x</th>
<th>0.125</th>
<th>0.250</th>
<th>0.375</th>
<th>0.500</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.125</td>
<td>0.42507 22749&lt;sup&gt;(AS)&lt;/sup&gt;</td>
<td>0.03176 98867&lt;sup&gt;(FI)&lt;/sup&gt;</td>
<td>0.02342 10396&lt;sup&gt;(CN)&lt;/sup&gt;</td>
<td></td>
</tr>
<tr>
<td>0.250</td>
<td>0.55347 08093</td>
<td>0.72065 37686</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.375</td>
<td>0.65001 36722</td>
<td>0.73568 33011</td>
<td>0.75102 62807</td>
<td></td>
</tr>
<tr>
<td>0.500</td>
<td>0.56539 08584</td>
<td>0.73617 44213</td>
<td>0.75152 76435</td>
<td>0.75202 93409</td>
</tr>
</tbody>
</table>

TABLE 7.11

Analytic (AS) and Computed [using the Fully Implicit (FI) and the Crank-Nicolson (CN) Implicit Schemes respectively] solutions of the parabolic 3D-problem for a grid (7x7x7) at the third time step [Δt=\(h^3 = \left(\frac{1}{8}\right)^3\)].
<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
<th>0.125</th>
<th>0.250</th>
<th>0.375</th>
<th>0.500</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.125</td>
<td></td>
<td></td>
<td>0.31819</td>
<td>11854&lt;sup&gt;(AS)&lt;/sup&gt;</td>
<td>0.44486</td>
<td>81769</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.03894</td>
<td>09041&lt;sup&gt;(FI)&lt;/sup&gt;</td>
<td>0.11830</td>
<td>24208</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.02962</td>
<td>61535&lt;sup&gt;(CN)&lt;/sup&gt;</td>
<td>0.11956</td>
<td>57318</td>
</tr>
<tr>
<td>0.250</td>
<td></td>
<td></td>
<td>0.62197</td>
<td>73104</td>
<td>0.40002</td>
<td>29509</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.49476</td>
<td>87672</td>
<td>0.49650</td>
<td>38752</td>
</tr>
<tr>
<td>0.375</td>
<td></td>
<td></td>
<td>0.64986</td>
<td>90742</td>
<td>0.59869</td>
<td>34751</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.46481</td>
<td>77187</td>
<td>0.61794</td>
<td>44490</td>
</tr>
<tr>
<td>0.500</td>
<td></td>
<td></td>
<td>0.67901</td>
<td>16178</td>
<td>0.72453</td>
<td>56226</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.46602</td>
<td>38141</td>
<td>0.68077</td>
<td>34973</td>
</tr>
<tr>
<td>0.14867</td>
<td>70481</td>
<td></td>
<td>0.62156</td>
<td>59588</td>
<td>0.68253</td>
<td>99485</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.51440</td>
<td>25609</td>
<td>0.64080</td>
<td>66017</td>
</tr>
<tr>
<td>0.15571</td>
<td>10730</td>
<td></td>
<td>0.74151</td>
<td>43610</td>
<td>0.66474</td>
<td>45068</td>
</tr>
<tr>
<td>0.125</td>
<td></td>
<td></td>
<td>0.86959</td>
<td>63963</td>
<td>0.89691</td>
<td>29012</td>
</tr>
<tr>
<td>0.250</td>
<td></td>
<td></td>
<td>0.61929</td>
<td>80110</td>
<td>0.94933</td>
<td>69848</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.65961</td>
<td>51330</td>
<td>0.82476</td>
<td>12631</td>
</tr>
<tr>
<td>0.375</td>
<td></td>
<td></td>
<td>0.74987</td>
<td>27773</td>
<td>0.90832</td>
<td>20484</td>
</tr>
<tr>
<td>0.76760</td>
<td>83011</td>
<td></td>
<td>0.93008</td>
<td>12300</td>
<td>0.95247</td>
<td>24790</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.91094</td>
<td>99181</td>
<td>0.95180</td>
<td>02967</td>
</tr>
<tr>
<td>0.500</td>
<td></td>
<td></td>
<td>0.68458</td>
<td>36479</td>
<td>0.85674</td>
<td>92738</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.76760</td>
<td>83011</td>
<td>0.89028</td>
<td>97563</td>
</tr>
<tr>
<td>0.500</td>
<td></td>
<td></td>
<td>0.99190</td>
<td>87819</td>
<td>0.99965</td>
<td>01622</td>
</tr>
<tr>
<td>0.375</td>
<td></td>
<td></td>
<td>0.87221</td>
<td>55182</td>
<td>0.95114</td>
<td>65365</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.93599</td>
<td>19914</td>
<td>0.98495</td>
<td>34627</td>
</tr>
<tr>
<td>0.375</td>
<td></td>
<td></td>
<td>0.99448</td>
<td>25578</td>
<td>0.99706</td>
<td>30122</td>
</tr>
<tr>
<td>0.500</td>
<td></td>
<td></td>
<td>0.90663</td>
<td>51897</td>
<td>0.94274</td>
<td>74400</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.95858</td>
<td>69830</td>
<td>0.98184</td>
<td>42759</td>
</tr>
<tr>
<td>0.500</td>
<td></td>
<td></td>
<td>0.99965</td>
<td>01622</td>
<td>0.95114</td>
<td>65365</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.98495</td>
<td>34627</td>
<td>0.99706</td>
<td>30122</td>
</tr>
</tbody>
</table>

**TABLE 7.12**

Analytic (AS) and Computed [using the Fully Implicit (FI) and the Crank-Nicolson (CN) Implicit Schemes respectively] solutions of the parabolic 3D-problem for a grid (7x7x7) at the fourth time step [$\Delta t = h^3 \left(\frac{1}{8}\right)^3$].
<table>
<thead>
<tr>
<th>Time Step</th>
<th>t</th>
<th>Modulus Error at $\left(\frac{1}{8}, \frac{1}{8}, \frac{1}{8}\right)$</th>
<th>Maximum Modulus Error</th>
<th>Modulus Error at $\left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)$</th>
<th>Maximum Modulus Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.016</td>
<td>0.10783 84500</td>
<td>0.30828 25113</td>
<td>0.14571 14373</td>
<td>0.13171 12925</td>
</tr>
<tr>
<td>2</td>
<td>0.031</td>
<td>0.02258 69100</td>
<td>0.13171 12925</td>
<td>0.05931 80900</td>
<td>0.05931 80900</td>
</tr>
<tr>
<td>3</td>
<td>0.047</td>
<td>0.00525 83980</td>
<td>0.01160 43345</td>
<td>0.01160 43345</td>
<td>0.01160 43345</td>
</tr>
<tr>
<td>4</td>
<td>0.063</td>
<td>0.00062 73487</td>
<td>0.01141 54882</td>
<td>0.01340 33480</td>
<td>0.01340 33480</td>
</tr>
<tr>
<td>5</td>
<td>0.078</td>
<td>0.00080 56422</td>
<td>0.01293 42865</td>
<td>0.01293 42865</td>
<td>0.01293 42865</td>
</tr>
<tr>
<td>10</td>
<td>0.156</td>
<td>0.00072 80248</td>
<td>0.00305 54197</td>
<td>0.00009 79028</td>
<td>0.00009 79028</td>
</tr>
<tr>
<td>15</td>
<td>0.234</td>
<td>0.00017 12647</td>
<td>0.00001 59722</td>
<td>0.00001 59722</td>
<td>0.00001 59722</td>
</tr>
<tr>
<td>20</td>
<td>0.313</td>
<td>0.00003 21005</td>
<td>0.00000 25385</td>
<td>0.00000 25385</td>
<td>0.00000 25385</td>
</tr>
<tr>
<td>25</td>
<td>0.391</td>
<td>0.00000 54867</td>
<td>0.00000 03972</td>
<td>0.00000 03972</td>
<td>0.00000 03972</td>
</tr>
<tr>
<td>30</td>
<td>0.469</td>
<td>0.00000 089512</td>
<td>0.00000 0616</td>
<td>0.00000 0616</td>
<td>0.00000 0616</td>
</tr>
<tr>
<td>35</td>
<td>0.547</td>
<td>0.00000 01423</td>
<td>0.00000 0095</td>
<td>0.00000 0095</td>
<td>0.00000 0095</td>
</tr>
<tr>
<td>40</td>
<td>0.625</td>
<td>0.00000 00223</td>
<td>0.00000 00014</td>
<td>0.00000 00014</td>
<td>0.00000 00014</td>
</tr>
<tr>
<td>45</td>
<td>0.703</td>
<td>0.00000 00034</td>
<td>0.00000 00002</td>
<td>0.00000 00002</td>
<td>0.00000 00002</td>
</tr>
<tr>
<td>50</td>
<td>0.781</td>
<td>0.82635×10⁻¹₀</td>
<td>0.00000 0002</td>
<td>0.00000 0002</td>
<td>0.00000 0002</td>
</tr>
<tr>
<td>55</td>
<td>0.859</td>
<td>0.12496×10⁻¹₀</td>
<td>0.00000 0002</td>
<td>0.00000 0002</td>
<td>0.00000 0002</td>
</tr>
</tbody>
</table>

**TABLE 7.13**

Modulus and Maximum Modulus errors of the parabolic 3D-problem for a grid (7x7x7) using the Fully Implicit Difference Formula (7.4.4) $[\Delta t=h^2=\left(\frac{1}{8}\right)^2]$.

<table>
<thead>
<tr>
<th>Time Step</th>
<th>t</th>
<th>Modulus Error</th>
<th>Maximum Modulus Error</th>
<th>Modulus Error</th>
<th>Maximum Modulus Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.002</td>
<td>0.89848 65963</td>
<td>0.2915 66648</td>
<td>0.9309 63163</td>
<td>0.71422 93307</td>
</tr>
<tr>
<td>2</td>
<td>0.004</td>
<td>0.57895 35516</td>
<td>0.1620 85760</td>
<td>0.58096 22618</td>
<td>0.18430 64484</td>
</tr>
<tr>
<td>3</td>
<td>0.006</td>
<td>0.39330 23882</td>
<td>0.02874 93482</td>
<td>0.04850 36258</td>
<td>0.12246 57529</td>
</tr>
<tr>
<td>4</td>
<td>0.008</td>
<td>0.27925 02814</td>
<td>0.07157 57047</td>
<td>0.18207 65008</td>
<td>0.39839 82716</td>
</tr>
<tr>
<td>5</td>
<td>0.010</td>
<td>0.20515 84340</td>
<td>0.16623 81288</td>
<td>0.24178 15310</td>
<td>0.24178 15310</td>
</tr>
<tr>
<td>10</td>
<td>0.020</td>
<td>0.06084 27025</td>
<td>0.18207 65008</td>
<td>0.15661 90681</td>
<td>0.15661 90681</td>
</tr>
<tr>
<td>15</td>
<td>0.029</td>
<td>0.02566 46452</td>
<td>0.12246 57529</td>
<td>0.09182 21288</td>
<td>0.09182 21288</td>
</tr>
<tr>
<td>20</td>
<td>0.039</td>
<td>0.01366 86769</td>
<td>0.09182 21288</td>
<td>0.06749 96002</td>
<td>0.06749 96002</td>
</tr>
<tr>
<td>25</td>
<td>0.049</td>
<td>0.00843 75872</td>
<td>0.04912 94750</td>
<td>0.04912 94750</td>
<td>0.04912 94750</td>
</tr>
<tr>
<td>30</td>
<td>0.059</td>
<td>0.00565 27425</td>
<td>0.03556 15508</td>
<td>0.03556 15508</td>
<td>0.03556 15508</td>
</tr>
<tr>
<td>35</td>
<td>0.068</td>
<td>0.00394 42912</td>
<td>0.02564 80082</td>
<td>0.02564 80082</td>
<td>0.02564 80082</td>
</tr>
<tr>
<td>40</td>
<td>0.078</td>
<td>0.00280 37383</td>
<td>0.01844 58111</td>
<td>0.01844 58111</td>
<td>0.01844 58111</td>
</tr>
<tr>
<td>45</td>
<td>0.088</td>
<td>0.00200 81858</td>
<td>0.01323 16045</td>
<td>0.01323 16045</td>
<td>0.01323 16045</td>
</tr>
<tr>
<td>50</td>
<td>0.098</td>
<td>0.00144 17402</td>
<td>0.01323 16045</td>
<td>0.01323 16045</td>
<td>0.01323 16045</td>
</tr>
<tr>
<td>55</td>
<td>0.107</td>
<td>0.00103 48844</td>
<td>0.01323 16045</td>
<td>0.01323 16045</td>
<td>0.01323 16045</td>
</tr>
<tr>
<td>60</td>
<td>0.117</td>
<td>0.00074 17705</td>
<td>0.01323 16045</td>
<td>0.01323 16045</td>
<td>0.01323 16045</td>
</tr>
</tbody>
</table>

**TABLE 7.14**

Modulus errors of the parabolic 3D-problem for a grid (7x7x7) using the Fully Implicit Difference Formula (7.4.4) $[\Delta t=h^2=\left(\frac{1}{8}\right)^2]$. 
7.7 CONCLUDING REMARKS

The main purpose of the two problems was to test the effectiveness of the derived algorithmic solution process (see section 3.4) by comparing the numerical with the analytic solution, finding the appropriate errors both at each internal mesh point (Elliptic problem) and at each time step (Parabolic problem).

Let $A_p$ be the exact solution of the P.D.E., $A_D$ be the exact solution of the difference equation and $A_N$ be the numerical solution of the difference equation (computed solution). Then, the quantity $(A_p - A_N)$, which represents the total error can be expressed as follows:

$$(A_p - A_N) = (A_p - A_D) + (A_D - A_N)$$

where

$(A_p - A_D)$ is the truncation error caused by finite distances between points of the difference mesh and

$(A_D - A_N)$ is the numerical error caused mainly by round-off (computer in exact arithmetic).

Convergence and Stability analysis conditions are satisfied when $A_D \rightarrow A_p$ as $h_x, h_y, h_z \rightarrow 0$ and when $(A_D - A_N)$ can be considered a "negligible" quantity respectively. Since with the implicit method the stability condition is satisfied [52], a large value of $(A_p - A_N)$ is caused by truncation errors.

In particular, from a comparison of the analytic and numerical solutions of the parabolic 3D-problem the quantity $(A_p - A_N)$ was seen to be considerably large for most values of $\Delta t$ and $h_x, h_y, h_z$.

In this case, as in [44], the truncation error proved to be the main factor of the above growth of total error.

Since the absolute value of the mesh size affects the
truncation error, good results could be expected only for sufficiently small values of $h_x, h_y, h_z$ (Elliptic problem) and $\Delta t$ (Parabolic problem). The experiments verified that reasonable results can be obtained with intervals of this size.

Consequently, the algorithmic solution process proves to be an effective and efficient method, given that the discrepancies between numerical and analytic solutions were caused mainly by truncation errors. The applicability and usefulness of the method can be verified by the presented results for both problems in Tables 7.1-7.14.

Finally, we state that the algorithmic procedure can be used for the solution of the "one-phase" oil reservoir flow problem [59], [45].
CHAPTER 8

ON THE SOLUTION OF MILDLY NON-LINEAR ELLIPTIC P.D.E.'S
8.1 INTRODUCTION

In this final Chapter we present inner and outer iterative procedures for which both the exact and approximate factorization processes developed in sections 3.2, 4.2, respectively are used and a strategy for the solution of mildly non-linear elliptic P.D.E.'s is described.

The applicability of the above factorization processes is considered by introducing the following two iterative procedures,

(i) Procedure I, where the outer iteration is a Picard or Newton iteration and the inner iteration is carried out directly by the LUBOT algorithm. In this case, the usual two level 'inner-outer' iterative scheme reduces to an equivalent one-level iteration.

(ii) Procedure II, where a two-level iteration is involved. The outer iteration is a Picard or modified Newton iteration and the inner iteration is performed by a modified strongly Implicit procedure (the ALUBOT algorithm).

8.2 LINEARIZATION AND QUASI-LINEARIZATION METHODS

We consider the system of non-linear equations

\[ Lu = f(u), \quad u \in \mathcal{R} \]  

where \( L \) is an elliptic partial differential operator (of at least second order), \( f \) is an \( n \)-component column vector given by

\[ f(u) = (f_1(u), f_2(u), \ldots, f_n(u)) \]  

and \( \mathcal{R} \) is a finite, connected, two-dimensional region subject to certain boundary conditions on \( \mathcal{C} \) the boundary of \( \mathcal{R} \).

The method of linearization solves the system (8.2.1) as the limit of the solutions of a sequence of linear equations of the same form as (8.2.1), using the method of successive approximations.
The simplest variant of the method is the Picard's iteration.

Let the sequence of functions \( \{u^{(k)}\} \) satisfy the sequence of linear P.D.E's

\[
Lu^{(k+1)} = f(u^{(k)}), \quad u^{(k)}, u^{(k+1)} \in \mathbb{R}, \quad (8.2.3)
\]

where \( u^{(0)} \) indicates the initial guess, and the same boundary conditions specified for \( u \).

When the sequence \( \{u^{(k)}\} \) converges, let say

\[
\lim_{k \to +\infty} u^{(k)} = \xi_0 \quad \text{with} \quad f(\xi_0) = 0, \quad \xi_0 \in \mathbb{R}, \quad (8.2.4)
\]

the convergence is linear, i.e.,

\[
u^{(k+1)} - u \equiv 0[u^{(k)} - u], \quad \text{as} \quad k \to +\infty . \quad (8.2.5)
\]

The main disadvantages of the method are:

(i) the convergence is relatively slow, geometric at best

and

(ii) an initial guess \( u^{(0)} \) sufficiently close to the solution \( u \) is required for convergence.

Assuming now that \( f \) is continuous and differentiable with respect to \( u \), the right hand side of (8.2.3) can be replaced by the expansion about \( u^{(k)} \) i.e.,

\[
f(u^{(k)}) + [u^{(k+1)} - u^{(k)}]f'(u^{(k)}) . \quad (8.2.6)
\]

Then, from (8.2.3), (8.2.6) we get the Newton's iteration:

\[
Lu^{(k+1)} = f(u^{(k)}) + [u^{(k+1)} - u^{(k)}]f'(u^{(k)}) \quad (8.2.7)
\]

where \( f' \) denotes the Jacobian of \( f \) with respect to \( u^{(k)} \), (see Definition 1.6.3).

The method of Quasilinearization replaces the non-linear equations (8.2.1) by the sequence of linear equations:

\[
Lu^{(k+1)} f'(u^{(k)})u^{(k+1)} = f(u^{(k)}) - f'(u^{(k)})u^{(k)} \quad (8.2.8)
\]

where (8.2.8) is subject to the same boundary conditions as (8.2.1).
When the sequence \( \{u(k)\} \) converges (see (8.2.4)), assuming that there are the required higher order derivatives of \( f(u) \) and the Jacobian \( f' \) be non-singular, then the convergence of the Newton's method is quadratic, i.e.

\[
\lim_{k \to +\infty} u(k+1) - u = 0[(u(k) - u)^2], \quad \text{as} \quad k \to +\infty. \tag{8.2.9}
\]

Definitions 8.4.1

The iterative scheme

\[
A(u(k+1) - u(k)) = -f(u(k)), \quad k = 0, 1, 2, \ldots \tag{8.2.10}
\]

where \( A \) is a non-singular matrix to be defined later, is said to be 'locally convergent' if assuming that \( u(0) \) is sufficiently close to a solution \( \xi_0 \) of \( f(u) = 0 \), then

\[
\lim_{k \to +\infty} u(k) = \xi_0.
\]

Note that:

(i) when \( f'(\xi_0) \) exists, the sufficient and necessary condition for local convergence is

\[
\rho(1 - A^{-1}f'(\xi_0)) < 1, \tag{8.2.11}
\]

where \( \rho \) denotes the spectral radius of the matrix.

It is well known [48] that the smaller the spectral radius, the more rapid the convergence.

(ii) for \( f(u(k)) = Au(k) - G(u(k)) \), where \( G \) is non-linear, the iterative scheme (8.2.10) becomes

\[
Au(k+1) = G(u(k)) \quad \text{(Picard's iteration)}
\]

while, for \( A = f'(u(k)) \) we obtain

\[
f'(u(k))(u(k+1) - u(k)) = -f(u(k)) \quad \text{(Newton iteration)}.
\]

8.3 FORMATION OF THE MILDLY NON-LINEAR ELLIPTIC DIFFERENCE EQUATIONS

We consider a class of mildly non-linear elliptic boundary value problems in two space dimensions of the form,

\[
\frac{3}{3x}(A(x,y) \frac{\partial U}{\partial x}) + \frac{3}{3y}(A(x,y) \frac{\partial U}{\partial y}) = f(x,y,U), \quad (x_i, y_j) \in \text{domain of} \quad \mathbb{R} \tag{8.3.1}
\]
where \( \Lambda, \tilde{\Lambda} \) are non-negative and \( R \) is a finite, connected, two-dimensional region subject to the general boundary condition,

\[
a U + \beta \frac{\partial U}{\partial \zeta} = \gamma, \quad (x_1, y_1) \in C
\]  

(8.3.2)

where \( C \) is the exterior boundary of \( R, \zeta \) denotes the direction of the outward drawn normal and \( a, \beta \) are positive, piecewise continuous on \( C \).

If a rectilinear network of mesh spacings \( h_x, h_y \) in the \( X, Y \) directions respectively is superimposed over the region \( R \), then the central finite difference analogue of \( (8.3.1) \) at the point \((i, j) = (ih_x, jh_y)\) is

\[
\left[ \delta_x (\Lambda_{i,j} \delta_x) + \delta_y (\tilde{\Lambda}_{i,j} \delta_y) \right] u_{i,j} = f(x_i, y_j, u_{i,j}), (x_i, y_j) \in R
\]

(8.3.3)

where \( \delta_x, \delta_y \) denote the usual central difference operators with respect to \( x, y \) respectively (see equations \( (2.6.7)-(2.6.8) \)).

The solution of the set of non-linear algebraic equations \( (8.3.3) \) can be obtained by

(i) the linearized Picard iteration \([1]\) defined by,

\[
\left[ \delta_x (\Lambda_{i,j} \delta_x) + \delta_y (\tilde{\Lambda}_{i,j} \delta_y) \right] u_{i,j}^{(k+1)} = f(x_i, y_j, u_{i,j}^{(k)}), (x_i, y_j) \in R
\]

(8.3.4)

(ii) the quasilinearized Newton second-order iteration \([4]\) given by,

\[
\left[ \delta_x (\Lambda_{i,j} \delta_x) + \delta_y (\tilde{\Lambda}_{i,j} \delta_y) \right] u_{i,j}^{(k+1)} - f'(x_i, y_j, u_{i,j}^{(k)}) u_{i,j}^{(k)} = \]

\[
f(x_i, y_j, u_{i,j}^{(k)}) - f'(x_i, y_j, u_{i,j}^{(k)}) u_{i,j}^{(k)}, (x_i, y_j) \in R
\]

(8.3.5)

where \( f' \) denotes the Jacobian of \( f \) with respect to \( u_{i,j}^{(k)} \).

The resulting large order, sparse, linear systems are of the form,

\[
\Lambda u^{(k+1)} = s(u^{(k)}),
\]

(8.3.6)
where the coefficient matrix A is of the same type and form as in (3.2.2)) and have to be solved many times.

Similarly we consider now a class of mildly non-linear elliptic boundary value problems in three space dimensions of the form:

\[
\frac{\partial}{\partial x}(A(x,y,z)\frac{\partial U}{\partial x}) + \frac{\partial}{\partial y}(\tilde{A}(x,y,z)\frac{\partial U}{\partial y}) + \frac{\partial}{\partial z}(\tilde{A}(x,y,z)\frac{\partial U}{\partial z}) = f(x,y,z,U),
\]

\((x_i, y_j, z_k) \equiv (ih_x, jh_y, kh_z) \in \mathbb{R}, \quad (8.3.7)\)

where \(A, \tilde{A}, \tilde{A}\) are non-negative and \(R\) is a finite, compact, three dimensional region subject to the general boundary conditions (8.3.2) with \((x_i, y_j, z_k) \in C\), the exterior boundary of \(R\).

Proceeding in a similar manner as in the two-dimensional case the following central finite difference analogue of (8.3.7) is obtained at the point \((i,j,k) = (ih_x, jh_y, kh_z)\),

\[
[\delta_x (A_{i,j,k} \delta_x) + \delta_y (\tilde{A}_{i,j,k} \delta_y) + \delta_z (\tilde{A}_{i,j,k} \delta_z)]u_{i,j,k} = f(x_i, y_j, z_k, u_{i,j,k}),
\]

\((x_i, y_j, z_k) \in \mathbb{R}, \quad (8.3.8)\)

where \(\delta\) denotes the usual central difference operator.

Then, the linearized Picard iteration is defined by:

\[
[\delta_x (A_{i,j,k} \delta_x) + \delta_y (\tilde{A}_{i,j,k} \delta_y) + \delta_z (\tilde{A}_{i,j,k} \delta_z)]u_{i,j,k}^{(n+1)} = f(x_i, y_j, z_k, u_{i,j,k}^{(n)}),
\]

\((x_i, y_j, z_k) \in \mathbb{R}; \quad (8.3.9)\)

and the quasilinearized Newton iteration is given by:

\[
[\delta_x (A_{i,j,k} \delta_x) + \delta_y (\tilde{A}_{i,j,k} \delta_y) + \delta_z (\tilde{A}_{i,j,k} \delta_z)]u_{i,j,k}^{(n+1)} = f(x_i, y_j, z_k, u_{i,j,k}^{(n)}) - f'(x_i, y_j, z_k, u_{i,j,k}^{(n)})u_{i,j,k}^{(n+1)},
\]

\((x_i, y_j, z_k) \in \mathbb{R}, \quad u_{i,j,k}^{(n)} = u_{i,j,k}^{(n+1)} - f'(x_i, y_j, z_k, u_{i,j,k}^{(n)})u_{i,j,k}^{(n+1)}, \quad (8.3.10)\)

where \(f'\) denotes the Jacobian of \(f\) with respect to \(u_{i,j,k}^{(n)}\).

Both cases lead to the solution of large order, sparse,
linear systems of the form,
\[ \Omega_{u}^{(n+1)} = s(u^{(n)}) \quad , \]
(8.3.11)
(where the coefficient matrix \( \Omega \) is of the same type and form as in (3.4.2)) and have to be solved many times.

Several techniques [17],[l] have appeared in the literature dealing with the solution of (8.3.4),(8.3.5),(8.3.9),(8.3.10). In the following section the LUBOT algorithm is used for a direct method of solution of the inner iteration in (8.3.6) and after convergence of the outer iteration the desired approximate solution is obtained.

Furthermore, the ALUBOT algorithm in conjunction with the standard stationary iteration method i.e., Simultaneous Displacement method, is incorporated as an inner iteration into (8.3.6) and an experimental estimation of the optimal iterative parameter involved is obtained.

8.4 GENERALIZED LINEAR METHODS: DERIVATION OF NEWTON-LUBOT AND NEWTON-ALUBOT METHODS

Utilizing the LUBOT-ALUBOT algorithms in connection with mildly non-linear equations, we consider the Newton-LUBOT and Newton-ALUBOT composite iterative schemes, where Newton's method is the outer (primary) iteration and the inner (secondary) iteration is carried out by LUBOT or ALUBOT algorithm.

Let us assume that the coefficient matrix \( A \) of the linear system
\[ Au = s \quad , \]
(8.4.1)
can be decomposed or split as \( A=B-R \), where \( B \) is non-singular and under the assumption that the linear system
\[ B.u = s^{*} \]
can be 'easily' solved. It is known that an iterative scheme can
be defined by
\[ Bu_{(r+1)} = Ru_{(r)} + s, \quad r=0,1,2,... \] (8.4.2)
or
\[ B(u_{(r+1)} - u_{(r)}) = s - Au_{(r)}, \quad r=0,1,2,... \] (8.4.3)
where appropriate splittings of matrices \( B \) and \( R \) lead to well
known standard iterative methods i.e., Simultaneous Displacement,
S.O.R., A.D.I., etc.

Let us assume that \( B=L_S U_S^r \) and \( R=L_S U_S - L_U \), where \( L, U \) and
\( L_S, U_S \) are the lower and upper triangular matrices in their full
and sparse forms given by (3.2.3), (3.2.4) and (4.2.2) respectively.

Then, the iterative scheme (8.4.3) can be written as
\[ L_S U_S (u_{(r+1)} - u_{(r)}) = \tau_r, \quad r=0,1,2,... \] (8.4.4)

where
\[ \tau_r = s - Au_{(r)} \] (8.4.5)

The derivation of the composite non-linear-linear methods
can now be considered for any linear iterative method of the
form (8.4.4) and any non-linear iterative method given by
\[ A_k(u_{(k+1)} - u_{(k)}) = -f(u_{(k)}), \quad k=0,1,2,... \] (8.4.6)

where the matrix \( A_k \) can be split as
\[ A_k = B_k - R_k \] (8.4.7)

Then, provided that the matrix \( B_k \) is non-singular we get
\[ A_k^{-1} = (I - B_k^{-1} R_k)^{-1} B_k^{-1} = (I + H_k + H_k^2 + \ldots + H_k^{u_k - 1}) B_k^{-1} \] (8.4.8)

where the first \( u_k \) terms have been retained in the expansion of
\( (I - B_k^{-1} R_k)^{-1} \) and \( I \) is the identity matrix. Combination of (8.4.6),
(8.4.8) leads to the following iterative scheme,
\[ u_{(k+1)} - u_{(k)} = -(I + H_k + \ldots + H_k^{u_k - 1}) B_k^{-1} f(u_{(k)}), \quad k=0,1,2,... \] (8.4.9)

where \( H_k = B_k^{-1} R_k, \quad k=0,1,2,... \).

The explicit procedure (8.4.9) is a composite iteration in
which, at the \( k \)th stage starting from \( u_{(k)} \), \( u_k \) steps of the inner
linear iteration are carried out in order to approximate a solution
of the outer iteration.

The choice,

$$A_k f'(u^{(k)}), \quad B_k = L_S U_S \quad \text{and} \quad R_k = L_S U_S - LU,$$

(8.4.10)

in which only the first term in the expansion in (8.4.8) is retained, leads to the following first order Newton-ALUBOT iterative scheme,

$$L_S U_S (u^{(k+1)} - u^{(k)}) = -f(u^{(k)}), \quad k=0,1,2,...,$$

(8.4.11)

whilst retaining the first two terms in the expansion in (8.4.8) a second order Newton-ALUBOT iteration can be obtained, viz.,

$$L_S U_S (u^{(k+1)} - u^{(k)}) = -[I - (L_S U_S)^{-1}(LU - L_S U_S)]f(u^{(k)}).$$

(8.4.12)

The Newton-LUBOT iterative scheme can be easily derived from (8.4.6), assuming that $A_k$ can be decomposed as $A_k = LU$ and is given by

$$LU(u^{(k+1)} - u^{(k)}) = -f(u^{(k)}), \quad k=0,1,2,....$$ 

(8.4.13)

### 8.5 ITERATIVE PROCEDURES AND NUMERICAL RESULTS

We consider the non-linear elliptic P.D.E.

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = e^U, \quad (x,y) \in R$$

(8.5.1)

where

$$R = \left\{ (x,y), \begin{array}{c} 0 \leq x \leq x_{\max} \\ 0 \leq y \leq y_{\max} \end{array} \right\},$$

(8.5.1a)

subject to the Dirichlet boundary conditions

$$U \equiv 0 \quad \text{and} \quad U \equiv 10$$

(8.5.2)

respectively, with $(x,y) \in C$, exterior boundary of $R$.

The equation (8.5.1) arises in magnetohydrodynamics, (specifically it is of physical interest in diffusion-reaction problems, Vortex problems and electric space charge considerations) and the existence and uniqueness of the solutions to the above boundary value problem are assured by the classical theory.
The linearized Picard and Quasilinearized Newton iterations are outer iterative schemes of the form,

\[ (\delta^2_x + \delta^2_y)u^{(k+1)} = e^{-1}u^{(k)}, \quad (x_i, y_j) \in (ih_x, jh_y) \in \mathbb{R}, \quad (8.5.3) \]

and

\[ (\delta^2_x + \delta^2_y)u^{(k+1)} - [e^{-1}u^{(k)}]u^{(k+1)} = [1 - u^{(k)}]e^{-1}u^{(k)}, \]

respectively. \((x_i, y_j) \in \mathbb{R}, \quad (8.5.4)\)

For notational simplification we assume that \(h_x = h_y = h\) and we denote the resulting set of \((-\frac{x_{\text{max}}}{h_x} - 1)(\frac{y_{\text{max}}}{h_y} - 1)\) numbers \(u_{i,j}^{(k)}\), which are the approximations to the solution at \((ih_x, jh_y)\) at the \(k\)th Picard or Newton iteration, by \(u^{(k)}\).

Then, a columnwise ordering of the points is introduced and the resulting large order, sparse, linear system (8.3.6) is obtained which can be written in the following matrix form,

\[
\begin{bmatrix}
B_1 & C_1 & 0 & & & \\
A_2 & B_2 & C_2 & 0 & & \\
& A_3 & B_3 & C_3 & 0 & \\
& & & \ddots & \ddots & \\
& & & & A_{n-1} & B_{n-1} & C_{n-1} \\
& & & & & A_n & B_n
\end{bmatrix} \begin{bmatrix}
\bar{u}^{(k+1)}
\end{bmatrix} = s(u^{(k)}) \quad (8.5.5)
\]

where \(B_i\) is the \(n \times n\) tridiagonal matrix

\[
B_i \equiv \begin{bmatrix}
b_{i+1} & c_{i+1} & & & & \\
a_{i+2} & b_{i+2} & c_{i+2} & & & \\
a_{i+3} & b_{i+3} & c_{i+3} & & & \\
& & & \ddots & \ddots & \ddots \\
& & & & a_{i+n-1} & b_{i+n-1} & c_{i+n-1} \\
& & & & & a_{i+n} & b_{i+n}
\end{bmatrix}, \quad i=1, 2, \ldots, n \quad (8.5.6)
\]
and $C_i, A_i$ are the diagonal matrices defined by,

$$C_i \equiv \text{diag} \{ \tau_{i+1}, \tau_{i+2}, \ldots, \tau_{i+n} \}, \quad i=1,2,\ldots,n-1,$$

and

$$A_i \equiv \text{diag} \{ v_{i+1}, v_{i+2}, \ldots, v_{i+n} \}, \quad i=2,3,\ldots,n,$$

with

$$\lambda = \begin{cases} 0, & \text{if } i=1 \\ \frac{n \times (i-1)}{1}, & \text{if } i \neq 1. \end{cases}$$

Equivalently the system (8.3.6) can be written as,

\[
\begin{bmatrix}
    b_1 & c_1 & \tau_1 \\
    a_2 & b_2 & c_2 & \tau_2 & 0 \\
    a_3 & b_3 & c_3 & 0 & \tau_{N-m+1} \\
    v_m &  &  &  &  \\
    v_{m+1} &  &  &  & \\
    0 &  &  &  & v_N \\
    a_{N-2} & b_{N-2} & c_{N-2} & 0 & \tau_{N-1} \\
    a_N & b_N & c_N & 0 & \tau_1 \\
    0 &  &  &  & v_N \\
    v_{m+1} &  &  &  & \\
    \vdots & \ddots & \ddots & \ddots & \ddots \\
    v_N &  &  &  & \\
    \end{bmatrix} \begin{bmatrix}
    u_{(k+1)}_1 \\
    u_{(k+1)}_2 \\
    \vdots \\
    u_{(k+1)}_N \\
    s_{(u_{(k)})}_1 \\
    \vdots \\
    s_{(u_{(k)})}_N \\
    \end{bmatrix} = \begin{bmatrix}
    b_1 & c_1 & \tau_1 \\
    a_2 & b_2 & c_2 & \tau_2 & 0 \\
    a_3 & b_3 & c_3 & 0 & \tau_{N-m+1} \\
    v_m &  &  &  &  \\
    v_{m+1} &  &  &  &  \\
    0 &  &  &  & v_N \\
    a_{N-2} & b_{N-2} & c_{N-2} & 0 & \tau_{N-1} \\
    a_N & b_N & c_N & 0 & \tau_1 \\
    0 &  &  &  & v_N \\
    v_{m+1} &  &  &  &  \\
    \vdots & \ddots & \ddots & \ddots & \ddots \\
    v_N &  &  &  &  \\
    \end{bmatrix} \begin{bmatrix}
    s_{(u_{(k)})}_1 \\
    s_{(u_{(k)})}_2 \\
    \vdots \\
    s_{(u_{(k)})}_N \\
    \end{bmatrix}
\]

\[\text{(8.5.10)}\]

where $c_{\rho(m-1)}=0, \quad a_{\rho(m-1)+1}=0, \quad \rho \in [1,m-2]$.

For the Picard iteration we have

$$b_i=1, \quad i \in [1,N]; \quad c_i=-1/4, \quad i \in [1,N-1]; \quad a_i=-1/4, \quad i \in [2,N];$$

\[\text{(8.5.11a)}\]

$$\tau_i=-1/4, \quad i \in [1,N-m+1]; \quad \nu_i=-1/4, \quad i \in [m,N];$$

\[\text{(8.5.11b)}\]

and

$$s(u_i^{(k)}) = -\frac{h^2}{4} u_i^{(k)} e_i^{(k)}, \quad i \in [1,N], \quad$$

\[\text{(8.5.11c)}\]

whilst for the Newton iteration we get

$$b_i=1, \quad i \in [1,N]; \quad c_i = \frac{1}{(4+h^2 e_i^{(k)})}, \quad i \in [1,N-1];$$

\[\text{(8.5.12a)}\]

$$a_i = \frac{1}{(4+h^2 e_i^{(k)})}, \quad i \in [2,N]; \quad \tau_i = \frac{1}{(4+h^2 e_i^{(k)})}, \quad i \in [1,N-m+1];$$

\[\text{(8.5.12b)}\]
\[ v_i = 1/(4 + h^2 e^{i}, \quad i \in [m, N] \quad \text{and} \quad (8.5.12b) \]
\[ s(u_i^{(k)}) = -h^2 e^{i} (1-u_i^{(k)})/(4 + h^2 e^{i}), \quad i \in [1, N]. \quad (8.5.12c) \]

An efficient solution of system (8.5.10) can be obtained as follows:

Procedure I

We apply the LUBOT algorithm, solving accurately the system (8.5.10) for each value of \( k \). This combination of a direct method for the solution of the inner iteration and the use of Picard or Newton iteration for the solution of the outer iteration yields the advantages of a direct method and on the other hand the technique proves to be a one-level iteration. The criterion for terminating the outer iteration was that the maximum of the absolute value of the relative change between consecutive iterations of the functional values be no greater than \( \varepsilon = 10^{-6} \), i.e.,

\[ \max_{i} \left| \frac{u_i^{(k+1)} - u_i^{(k)}}{u_i^{(k+1)}} \right| \leq \varepsilon. \quad (8.5.13) \]

The heuristic character of this termination criterion is particularly useful, if the solution of the considered problem or effective bounds on it, are not known.

For the set of boundary conditions \( U = 0 \), the initial guess was fixed to be \( u^{(0)} = 0 \), while for \( U = 10 \) on the boundary the initial guess \( u^{(0)} = 6 \) was chosen.

The numerical experiments were carried out for the non-linear elliptic P.D.E's \( \nabla^2 U = e^U \) and \( \nabla^2 U = U^2 \), with \( h = \frac{1}{64} \) and \( x_{\max} = \frac{1}{2}, y_{\max} = \frac{1}{4} \).

Numerical results for the Procedure I, with the same accuracy used in both cases, are presented in Table 8.5.1.
<table>
<thead>
<tr>
<th>Non-linear P.D.E.</th>
<th>Method - Outer Iteration</th>
<th>No. of Iterations</th>
<th>Value of $u^{(k)}$ at $\left(\frac{1}{4}, \frac{1}{8}\right)$</th>
<th>Value of $u^{(k)}$ at $\left(\frac{1}{64}, \frac{1}{64}\right)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\nabla^2 U = e^U$</td>
<td>B.C. $U=0$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Picard</td>
<td>4</td>
<td>-0.00707 07165 88</td>
<td>-0.00041 00262</td>
<td></td>
</tr>
<tr>
<td>Newton</td>
<td>3</td>
<td>-0.00707 07165 59</td>
<td>-0.00041 00262</td>
<td></td>
</tr>
<tr>
<td>B.C. $U=10$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Picard (OSC)</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Newton</td>
<td>6</td>
<td>5.65995 02703</td>
<td>8.81975 40932</td>
<td></td>
</tr>
<tr>
<td>$\nabla^2 U = U^2$</td>
<td>B.C. $U=0$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Picard</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Newton</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>B.C. $U=10$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Picard</td>
<td>7</td>
<td>9.35894 18443</td>
<td>9.96069 22215</td>
<td></td>
</tr>
<tr>
<td>Newton</td>
<td>4</td>
<td>9.35894 14802</td>
<td>9.96069 22144</td>
<td></td>
</tr>
</tbody>
</table>

**TABLE 8.5.1**

Procedure I. Numerical results for Picard and Newton iterations for the non-linear P.D.E's $\nabla^2 U = e^U$ and $\nabla^2 U = U^2$. The values of the solution at the centre and near the origin of the considered rectangular region are given.

(OSC) In the case of $\nabla^2 U = e^U$, with $U=10$ on the boundary, although a sufficiently close initial guess to the solution was given, it was found that the conditions for local convergence of the Picard method are not satisfied.
Procedure II

In this case the composite iterative scheme is given by

\[ L_s U_s (u^{(k+1)} - u^{(k)}) = a \frac{r^{(k+1)}}{r^{(r)}} , \]  

(8.5.14)

where the superscript \( k \) denotes the outer iteration index and the subscript \( r \) the inner iteration index, \( L_s \) and \( U_s \) are respectively lower and upper sparse triangular matrices of the form (4.2.2), appropriate to the particular method (Picard or Newton method), \( r^{(k+1)} \) is the well known residual factor \( r^{(k+1)} = s - A u^{(k+1)} \) and \( a \) is a pre-determined acceleration parameter. The inner iteration is now carried out by the ALUBOT algorithm given a sufficient near approximate solution. At this stage, it should be pointed out that

(i) if the approximation to the solution obtained by ALUBOT is not sufficiently good i.e., an improper choice of convergence criteria or value of parameter \( a \), then a good approximation to the original non-linear elliptic P.D.E. cannot be achieved from (8.5.14),

(ii) if \( u^{(k+1)} \) is not a good approximation to the solution of equation (8.5.1), there is no need to obtain a very accurate solution of (8.5.14).

(iii) the better the inner iteration process, the better the overall method.

The termination criterion for the outer loop was the same as in Procedure I, while as stopping criterion for the inner loop the mixed error test was used, i.e.,

\[ \frac{|u^{(k+1)} - u^{(k)}|}{|r^{(k+1)}|} < \varepsilon_I , \]  

(8.5.15)
where the accuracy $\varepsilon_1$ was initially taken $\varepsilon_1 = 10^{-2}$ and then was decreased at each iterative step by $\varepsilon_1/10$ up to $10^{-6}$ where it remained constant during the next iterative steps.

For the Picard iteration, in the case of boundary conditions $U\equiv 0$ an optimal $a = 1.02$ was obtained with the overall number of iterations 21, the number of outer iterations 12 (denoting how many times the criterion (8.5.13) has been applied) and the value at the point $(1/4, 1/8) -0.0070707124$ to eight significant figures. Starting the iterative process with the accuracy $\varepsilon_1 = 10^{-6}$ and then constant throughout the iteration, the same optimum value of parameter $a$ was obtained, with the overall number of iterations 27, the number of outer iterations 9 and the value at the central point $-0.0070707203$. For the Newton iteration the optimal $a = 1.01$ was obtained with overall number of iterations 22, number of outer iterations 13 and value at the central point $-0.0070707141$.

In the case of boundary conditions $U\equiv 10$, for the Newton iteration the optimal parameter $a = 0.96$ was found, with overall number of iterations 21 and number of outer iterations 6. The value at the central point was found $5.6599508873$. For the Picard iteration, as in Procedure I, the conditions for local convergence are not satisfied.

Numerical results for Procedure II are presented in Table 8.5.2.
<table>
<thead>
<tr>
<th>Parameter a</th>
<th>No. of Overall Iterations</th>
<th>No. of Outer Iterations</th>
<th>Value of $u^{(k)}$ at $\left(\frac{1}{4}, \frac{1}{8}\right)$</th>
<th>Value of $u^{(k)}$ at $\left(\frac{1}{64}, \frac{1}{64}\right)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.C. $U=0$, PICARD METHOD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.27</td>
<td>89</td>
<td>48</td>
<td>-0.00707 07164 05</td>
<td>-0.00041 00261 95</td>
</tr>
<tr>
<td>1.26</td>
<td>80</td>
<td>43</td>
<td>-0.00707 07167 66</td>
<td>-0.00041 00262 06</td>
</tr>
<tr>
<td>1.22</td>
<td>56</td>
<td>31</td>
<td>-0.00707 07167 30</td>
<td>-0.00041 00262 01</td>
</tr>
<tr>
<td>1.18</td>
<td>43</td>
<td>25</td>
<td>-0.00707 07165 02</td>
<td>-0.00041 00262 04</td>
</tr>
<tr>
<td>1.14</td>
<td>34</td>
<td>21</td>
<td>-0.00707 07166 70</td>
<td>-0.00041 00261 96</td>
</tr>
<tr>
<td>1.10</td>
<td>28</td>
<td>18</td>
<td>-0.00707 07166 47</td>
<td>-0.00041 00261 95</td>
</tr>
<tr>
<td>1.08</td>
<td>26</td>
<td>17</td>
<td>-0.00707 07166 02</td>
<td>-0.00041 00261 96</td>
</tr>
<tr>
<td>1.06</td>
<td>23</td>
<td>15</td>
<td>-0.00707 07160 72</td>
<td>-0.00041 00262 02</td>
</tr>
<tr>
<td>1.02</td>
<td>21</td>
<td>12</td>
<td>-0.00707 07123 53</td>
<td>-0.00041 00262 30</td>
</tr>
<tr>
<td>1.00</td>
<td>22</td>
<td>13</td>
<td>-0.00707 07133 17</td>
<td>-0.00041 00263 44</td>
</tr>
<tr>
<td>0.98</td>
<td>22</td>
<td>13</td>
<td>-0.00707 07116 67</td>
<td>-0.00041 00261 16</td>
</tr>
<tr>
<td>0.96</td>
<td>23</td>
<td>14</td>
<td>-0.00707 07126 18</td>
<td>-0.00041 00261 32</td>
</tr>
<tr>
<td>B.C. $U=0$, NEWTON METHOD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.20</td>
<td>48</td>
<td>27</td>
<td>-0.00707 07167 17</td>
<td>-0.00041 00261 99</td>
</tr>
<tr>
<td>1.16</td>
<td>38</td>
<td>23</td>
<td>-0.00707 07166 73</td>
<td>-0.00041 00261 97</td>
</tr>
<tr>
<td>1.14</td>
<td>34</td>
<td>21</td>
<td>-0.00707 07166 65</td>
<td>-0.00041 00261 96</td>
</tr>
<tr>
<td>1.12</td>
<td>31</td>
<td>19</td>
<td>-0.00707 07165 38</td>
<td>-0.00041 00262 06</td>
</tr>
<tr>
<td>1.10</td>
<td>28</td>
<td>18</td>
<td>-0.00707 07166 42</td>
<td>-0.00041 00261 95</td>
</tr>
<tr>
<td>1.08</td>
<td>25</td>
<td>16</td>
<td>-0.00707 07164 51</td>
<td>-0.00041 00262 08</td>
</tr>
<tr>
<td>1.06</td>
<td>23</td>
<td>15</td>
<td>-0.00707 07160 07</td>
<td>-0.00041 00262 00</td>
</tr>
<tr>
<td>1.02</td>
<td>22</td>
<td>13</td>
<td>-0.00707 07141 20</td>
<td>-0.00041 00261 57</td>
</tr>
<tr>
<td>1.00</td>
<td>22</td>
<td>13</td>
<td>-0.00707 07128 72</td>
<td>-0.00041 00261 36</td>
</tr>
<tr>
<td>0.96</td>
<td>23</td>
<td>13</td>
<td>-0.00707 07121 51</td>
<td>-0.00041 00261 24</td>
</tr>
<tr>
<td>0.92</td>
<td>24</td>
<td>14</td>
<td>-0.00707 07110 67</td>
<td>-0.00041 00261 05</td>
</tr>
<tr>
<td>0.90</td>
<td>25</td>
<td>14</td>
<td>-0.00707 07118 67</td>
<td>-0.00041 00261 19</td>
</tr>
</tbody>
</table>

**TABLE 8.5.2**

Procedure II. Numerical results of Picard and Newton iterations for the non-linear P.D.E. $v^2u = e^u$, when $U=0$ on the boundary. The values of the solution at the centre and near the origin of the considered rectangular region are given.
<table>
<thead>
<tr>
<th>Parameter a</th>
<th>No. of Overall Iterations</th>
<th>No. of Outer Iterations</th>
<th>Value of $u(k)$ at $\left(\frac{1}{4}, \frac{1}{8}\right)$</th>
<th>Value of $u(k)$ at $\left(\frac{1}{64}, \frac{1}{64}\right)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.28</td>
<td>42</td>
<td>6</td>
<td>5.65995 02715</td>
<td>8.81975 41595</td>
</tr>
<tr>
<td>1.24</td>
<td>37</td>
<td>6</td>
<td>5.65995 02570</td>
<td>8.81975 41075</td>
</tr>
<tr>
<td>1.20</td>
<td>33</td>
<td>6</td>
<td>5.65995 02903</td>
<td>8.81975 40193</td>
</tr>
<tr>
<td>1.16</td>
<td>30</td>
<td>6</td>
<td>5.65995 02821</td>
<td>8.81975 41197</td>
</tr>
<tr>
<td>1.12</td>
<td>25</td>
<td>6</td>
<td>5.65995 03298</td>
<td>8.81975 40804</td>
</tr>
<tr>
<td>1.08</td>
<td>25</td>
<td>6</td>
<td>5.65995 03729</td>
<td>8.81975 40815</td>
</tr>
<tr>
<td>1.04</td>
<td>22</td>
<td>6</td>
<td>5.65995 04826</td>
<td>8.81975 40870</td>
</tr>
<tr>
<td>1.00</td>
<td>22</td>
<td>6</td>
<td>5.65995 06378</td>
<td>8.81975 40920</td>
</tr>
<tr>
<td>0.96</td>
<td>21</td>
<td>6</td>
<td>5.65995 08873</td>
<td>8.81975 40953</td>
</tr>
<tr>
<td>0.92</td>
<td>23</td>
<td>6</td>
<td>5.65995 06605</td>
<td>8.81975 40947</td>
</tr>
<tr>
<td>0.88</td>
<td>23</td>
<td>6</td>
<td>5.65995 09940</td>
<td>8.81975 40968</td>
</tr>
<tr>
<td>0.84</td>
<td>25</td>
<td>6</td>
<td>5.65995 08730</td>
<td>8.81975 40990</td>
</tr>
</tbody>
</table>

**TABLE 8.5.3**

Procedure II. Numerical results of Newton iteration for the non-linear P.D.E. $\nabla^2 U = e^U$, when $U = 10$ on the boundary.

From a comparison of numerical results obtained by application of a composite Newton-S.O.R. iterative scheme given in [4] and experimental results obtained by applying the Newton-ALUBOT iterative scheme (see Tables 8.5.2, 8.5.3) it can be easily seen that for optimal parameters, the former iterative scheme takes up to two times as many iterations as the latter composite method. This result must be considered in the light of the computational effort required to complete the inner iteration which is nearly twice for the latter than the former.

Note that for the Picard iteration the coefficient matrix of system (8.5.10) is factorized once, while the Newton iteration requires the
coefficient matrix to be factorized every iterative step. A reasonable approach, dictating further experimentation, to tackle this problem is the consideration that the coefficient matrix is factorized after a certain number of iterations.

Remarks

(i) The applicability of the LUBOT-ALUBOT algorithms in the non-linear equations can be furthermore extended deriving analogous combined methods, in which the inner iteration is represented by one of LUBOT, ALUBOT algorithms and the outer iteration is one of the Secant or Steffensen methods.

(ii) It is expected that a non-linear ALUBOT iteration can be derived (a reversed composite scheme to Newton-ALUBOT method), in which the one-dimensional Newton method can be used as inner iteration and the ALUBOT as outer iteration.

(iii) It can be easily seen that for a class of mildly non-linear self-adjoint P.D.E.'s the NOBAR algorithm can be incorporated as an inner iteration into (8.3.6), giving rise to a Newton-NOBAR method, (similarly the Secant-NOBAR, Steffensen-NOBAR methods can be derived).

(iv) Finally the extension of the foregoing combined methods to three space dimensions can be considered using the LUBOT-3D, ALUBOT-3D and NB3D,NOBAR-3D (for the self-adjoint case) algorithmic procedures for the solution of the linear system (8.3.11).
Chapter 9

CONCLUDING REMARKS AND DISCUSSION
In this thesis, the aim was to derive direct and iterative algorithmic procedures for the solution of Elliptic and Parabolic P.D.E.'s, in both two and three space dimensions using finite difference approximations.

The particular strategies of the derived algorithmic procedures are that more outermost off-diagonal entries are retained in the solution process in comparison with known variants of Stone's strongly implicit procedure, and which enables one to obtain faster convergence rates.

These new algorithmic procedures are the central framework for the efficient solution of boundary value problems as already discussed in Chapters 2, 7 and 8.

The researches have shown that extremely powerful convergent methods for the solution of large scale problems in scientific computing can be derived. To offset against these advantages are the extra computational work involved in the algorithmic procedures.

Generally speaking we can say that the application of these new computational and algorithmic techniques can be extended to the solution of more general boundary value problems under different boundary conditions in which the region under consideration is also of general shape.

Finally, it should be mentioned that further research work on these topics includes:

(i) Extension of the derived direct and iterative algorithmic procedures for the finite difference of Elliptic and Parabolic P.D.E.'s in n dimensions \( (n>3) \).

(ii) Extension of the presented algorithmic procedures such that more complicated banded matrices derived from finite element discretization, i.e.,
can be factorized exactly and approximately to yield direct and iterative algorithmic procedures for the finite element solution of boundary value problems.

(iii) Combinations of the so-obtained algorithmic procedures with linearized and quasilinearized iterative schemes for the solution of complicated non-linear problems in both two and three space dimensions.
REFERENCES


differential equations", Pergamon Press.

Oxford University Press.


methods", Research report 77.04R, Comp. Sc., Chalmers
University of Technology and University of Göteborg.

gradients for solving linear systems", J. Research, Nat.

[33] Hildebrand, F.B. (1968): "Finite difference equations and
simulations", Prentice-Hall, Inc., Englewood Cliffs,
New Jersey.

[34] Householder, A.S. (1964): "The theory of matrices in numerical
analysis", Blaisdell, New York.

methods", John Wiley and Sons Ltd.

Center for Numerical Analysis, University of Texas, Austin.


procedure for solution of systems of parabolic and elliptic
equations in three dimensions", I. and E.C. Fundamentals,
Vol.8, No.2, pp.281-287.

simultaneous equations by Chebychev extrapolation when the
eigenvalues of the iteration matrix are complex",


unpublished.

for solving elliptic and parabolic partial differential
APPENDIX 1
A TEST PROBLEM

In order to define the elements of the submatrices $r$, $u$ and $d$ from the identities (3.5.6), (3.5.7), in the case of the NB3D algorithm (section 3.5), let us consider the following simplified structure of the coefficient matrix $A$ of order 10.

\[
A = \begin{bmatrix}
  a_1 & b_1 & c_1 & h_1 \\
  b_1 & a_2 & b_2 & c_2 & h_2 \\
  b_2 & a_3 & b_3 & c_3 & h_3 \\
  b_3 & a_4 & b_4 & c_4 \\
  c_1 & b_4 & a_5 & b_5 & c_5 \\
  c_2 & b_5 & a_6 & b_6 & c_6 \\
  c_3 & b_6 & a_7 & b_7 \\
  h_1 & c_4 & b_7 & a_8 & b_8 \\
  h_2 & c_5 & b_8 & a_9 & b_9 \\
  h_3 & c_6 & b_9 & a_{10} \\
\end{bmatrix}
\]

where

\[\bar{\mathbf{d}} = \text{diag} \{d_1, d_2, \ldots, d_{10}\}\] (A1.1)

\[\bar{T} = \begin{bmatrix}
  e_1 & t_{11} & r_{11} \\
  e_2 & t_{21} & t_{12} & r_{21} & r_{12} \\
  e_3 & t_{31} & t_{22} & t_{13} & r_{31} & r_{22} & r_{13} \\
  e_4 + t_{41} & t_{23} & t_{23} & r_{41} & r_{32} & r_{23} \\
  e_5 + t_{51} & r_{51} & r_{52} & r_{53} \\
  e_6 + t_{61} & r_{61} & r_{62} & r_{63} \\
  e_7 + t_{71} & r_{71} & r_{72} & r_{73} \\
  e_8 + t_{81} & r_{81} & r_{82} & r_{83} \\
  e_9 + t_{91} & r_{91} & r_{92} & r_{93} \\
\end{bmatrix}
\]

(A1.2)

(A1.3)
From the identity (3.5.6) we obtain:

\[
\begin{bmatrix}
  h_1 \\
  h_2 \\
  h_3 \\
  c_4 \\
  c_5 \\
  c_6 \\
  b_7
\end{bmatrix}
= \begin{bmatrix}
  d_1 \\
  d_2 \\
  d_3 \\
  d_4 \\
  d_5 \\
  d_6 \\
  d_7
\end{bmatrix}
\begin{bmatrix}
  1 \\
  e_1 \\
  e_2 \\
  e_3 \\
  t_{11} \\
  t_{12} \\
  \vdots
\end{bmatrix}
\begin{bmatrix}
  r_{11} \\
  r_{21} \\
  r_{31} \\
  r_{41} \\
  r_{51} \\
  r_{61} \\
  \vdots
\end{bmatrix}
\begin{bmatrix}
  d_8 \\
  d_9 \\
  d_{10}
\end{bmatrix}
\]

\[
\begin{bmatrix}
  d_1 d_8 r_{11} \\
  d_2 d_8 (e_1 r_{11} + r_{21}) \\
  d_3 d_8 (e_2 r_{21} + r_{31}) \\
  d_4 d_8 (e_3 r_{31} + r_{41}) \\
  d_5 d_8 [(\frac{4}{5} t_{k1} r_{12}^1 + e_4 r_{41} + r_{51})] \\
  d_6 d_8 [(\frac{4}{5} t_{k2} r_{k1+1}^1 + e_5 r_{51} + r_{61})] \\
  d_7 d_8 [(\frac{4}{5} t_{k3} r_{k2+1}^1 + e_6 r_{61} + r_{71})]
\end{bmatrix}
\]

\[
\begin{bmatrix}
  d_2 d_9 r_{12} \\
  d_3 d_9 (e_2 r_{12} + r_{22}) \\
  d_4 d_9 (e_3 r_{22} + r_{32}) \\
  d_5 d_9 [(\frac{4}{5} t_{k1} r_{k-1,2} + e_4 r_{32} + r_{42})] \\
  d_6 d_9 [(\frac{4}{5} t_{k2} r_{k-1,3} + e_5 r_{42} + r_{52})] \\
  d_7 d_9 [(\frac{4}{5} t_{k3} r_{k-1,2} + e_6 r_{52} + r_{62})]
\end{bmatrix}
\]

\[
\begin{bmatrix}
  d_3 d_{10} r_{13} \\
  d_4 d_{10} (e_3 r_{13} + r_{23}) \\
  d_5 d_{10} [(\frac{4}{5} t_{k1} r_{k-2,3} + e_4 r_{33} + r_{43})] \\
  d_6 d_{10} [(\frac{4}{5} t_{k2} r_{k-2,4} + e_5 r_{43} + r_{53})] \\
  d_7 d_{10} [(\frac{4}{5} t_{k3} r_{k-2,5} + e_6 r_{53} + r_{63})]
\end{bmatrix}
\]

(A1.4)
From identity (3.5.7) we get

\[
\begin{bmatrix}
  a_8 & b_8 \\
  b_8 & a_9 & b_9 \\
  b_9 & a_10
\end{bmatrix} =
\begin{bmatrix}
  d_8 \\
  d_9 \\
  d_{10}
\end{bmatrix} \cdot
\begin{bmatrix}
  r_{11} & r_{21} & r_{31} & r_{41} & r_{51} & r_{61} (e_7 + r_{71}) \\
  r_{12} & r_{22} & r_{32} & r_{42} & r_{52} & r_{62} \\
  r_{13} & r_{23} & r_{33} & r_{43} & r_{53} \\
  (e_7 + r_{71}) & r_{62} & r_{53}
\end{bmatrix}
\begin{bmatrix}
  r_{11} \\
  r_{21} \\
  r_{31} \\
  r_{41} \\
  r_{51} \\
  r_{61} (e_7 + r_{71})
\end{bmatrix} +
\begin{bmatrix}
  d_8 \\
  d_9 \\
  d_{10}
\end{bmatrix}
\]

\[
\begin{bmatrix}
  d_8 \\
  d_9 \\
  d_{10}
\end{bmatrix} +
\begin{bmatrix}
  1 & (e_8 + r_{72}) & r_{63} \\
  (e_8 + r_{72}) & 1 & (e_9 + r_{73}) \\
  r_{63} & (e_9 + r_{73}) & 1
\end{bmatrix}
\begin{bmatrix}
  d_8 \\
  d_9 \\
  d_{10}
\end{bmatrix} =
\begin{bmatrix}
  d_8^2 [1 + \frac{6}{1} \sum_{k=1}^{r_{k+1}} (e_7 + r_{71})^2] \\
  d_8^2 [\frac{6}{1} \sum_{k=1}^{r_{k+1}} r_{k2} + e_7 r_{62} + e_8 + r_{72}] \\
  d_8^2 [\frac{5}{1} \sum_{k=1}^{r_{k+2}} r_{k2} + e_8 + r_{72}]
\end{bmatrix}
\begin{bmatrix}
  d_8^2 [1 + \frac{6}{1} \sum_{k=1}^{r_{k+1}} (e_8 + r_{72})^2] \\
  d_8^2 [\frac{6}{1} \sum_{k=1}^{r_{k+1}} r_{k2} + (e_8 + r_{72})^2] \\
  d_8^2 [\frac{6}{1} \sum_{k=1}^{r_{k+1}} r_{k2} + (e_8 + r_{72})^2]
\end{bmatrix}
\begin{bmatrix}
  d_8^2 [1 + \frac{6}{1} \sum_{k=1}^{r_{k+1}} (e_7 + r_{72})^2] \\
  d_8^2 [\frac{6}{1} \sum_{k=1}^{r_{k+1}} r_{k2} + e_7 r_{62} + e_8 + r_{72}] \\
  d_8^2 [\frac{5}{1} \sum_{k=1}^{r_{k+2}} r_{k2} + e_7 r_{53} + e_9 + r_{73}]
\end{bmatrix}
\]

(A1.5)
On equating (A1.4) we obtain the elements of the submatrix \( r \) as follows:

\[
\begin{align*}
  r_{11} &= h_1/d_1 d_8 \\
  r_{21} &= -e_1 r_{11} \\
  r_{31} &= -e_2 r_{21} \\
  r_{41} &= c_4/d_4 d_8 e_3 r_{31} \\
  r_{51} &= -e_4 r_{41} - \frac{4}{1} t_{k1} r_{k1} \\
  r_{61} &= -e_5 r_{51} - \frac{4}{1} t_{k2} r_{k2+1,1} \\
  r_{71} &= -e_6 r_{61} - \frac{4}{1} t_{k3} r_{k2+2,1} \\
  r_{12} &= h_2/d_2 d_9 \\
  r_{22} &= -e_2 r_{12} \\
  r_{32} &= -e_3 r_{22} \\
  r_{42} &= c_5/d_5 d_9 e_4 r_{32} - \frac{4}{2} t_{k1} r_{k-1,2} \\
  r_{52} &= -e_5 r_{42} - \frac{4}{1} t_{k2} r_{k2} \\
  r_{62} &= -e_6 r_{52} - \frac{4}{1} t_{k3} r_{k+1,2} \\
  r_{13} &= h_3/d_3 d_{10} \\
  r_{23} &= -e_3 r_{13} \\
  r_{33} &= -e_4 r_{23} - \frac{4}{3} t_{k1} r_{k-2,3} \\
  r_{43} &= c_6/d_6 d_{10} e_5 r_{33} - \frac{4}{2} t_{k2} r_{k-1,3} \\
  r_{53} &= -e_6 r_{43} - \frac{4}{1} t_{k3} r_{k3} \\
  r_{63} &= -e_7 r_{53} - \frac{5}{1} t_{k2} r_{k+2,1} r_{k3} \\
  r_{73} &= b_9/d_9 d_{10} e_8 r_{72} - \frac{6}{1} t_{k1} r_{k+1,2} r_{k3} \\
  d_8 &= a_8^{\frac{1}{2}} [1 + \frac{6}{1} t_{k1} r_{k} (e_7 r_{71})^2]^\frac{1}{2} \\
  d_9 &= a_9^{\frac{1}{2}} [1 + \frac{6}{1} t_{k2} r_{k} (e_8 r_{72})^2]^\frac{1}{2} \\
  d_{10} &= a_{10}^{\frac{1}{2}} [1 + \frac{6}{1} t_{k3} r_{k} (e_9 r_{73})^2]^\frac{1}{2}
\end{align*}
\]

The elements of submatrices \( u \) and \( \tilde{d} \) are obtained by equating (A1.5) and are given by:

\[
\begin{align*}
  r_{72} r_{e_8} &= b_8/d_8 d_9 e_7 r_{62} - \frac{6}{1} t_{k1} r_{k+1,2} r_{k2} \\
  d_9 &= a_9^{\frac{1}{2}} [1 + \frac{6}{1} t_{k2} r_{k} (e_8 r_{72})^2]^\frac{1}{2} \\
  r_{63} &= -e_7 r_{53} - \frac{5}{1} t_{k2} r_{k+2,1} r_{k3} \\
  r_{73} r_{e_9} &= b_9/d_9 d_{10} e_8 r_{63} - \frac{6}{1} t_{k1} r_{k+1,2} r_{k3} \\
  d_{10} &= a_{10}^{\frac{1}{2}} [1 + \frac{6}{1} t_{k3} r_{k} (e_9 r_{73})^2]^\frac{1}{2} \\
\end{align*}
\]
APPENDIX 2
SUBROUTINE LUBOT(N,M,B,A,C,U,V)
C THIS SUBROUTINE IS A FACTORIZATION OF A SQUARE MATRIX OF
C ORDER N. THE COEFFICIENT MATRIX IS NON-SYMMETRIC, POSITIVE
C DEFINITE AND (QUINDIAGONAL OF BANDWIDTH M (3<M<N),
C B,C,A,U,V ARE VECTORS CONTAINING RESPECTIVELY THE DIAGONAL,
C UPPER AND LOWER \text{ CO-O-DIAGONAL AND M-TH DIAGONAL ELEMENTS.}
C THE COEFFICIENT MATRIX IS FACTORIZED INTO LU WHERE
C L,U (WITH UNIT DIAGONAL ELEMENTS) ARE STRICTLY LOWER AND
C UPPER TRIANGULAR MATRICES WITH NON-ZERO ELEMENTS IN SUB,
C SUPER DIAGONAL RESPECTIVELY AND RETAINING M1=M-1 UUTERMOST
C OFF-DIAGONAL ENTRIES. THREE RESULT VECTORS OF LENGTH N AND
C TWO ARRAYS OF DIMENSION [M-1,N-M+1] ARE USED AS WORKSPACE.
C
N1=N-M+1.
DIMENSION B(N),A(N),C(N),U(N),V(N)
CUM1ON /BLOCK2/G(N),W(N),D(N)
CUMHON /BLOCK4/H(M1,N1),E(M1,N1)
C
FACTORIZATION OF TRIDIAGONAL MATRIX A11
W(1)=B(1)
D(1)=A(2)
G(1)=C(1)/W(1)
DO 5 I=2,M-2
W(I)=D(I-1)-D(I-1)*G(I-1)
D(I)=A(I+1)
5 G(I)=C(I)/W(I)
W(M-1)=B(M-1)-D(M-2)*G(M-2)
C
CALCULATION OF ELEMENTS OF SUBMATRICES U12,L12, U22, L22 .
DO 6 J=1,N-M+1
E(1,J)=V(J+M-1)
H(1,J)=U(J)/W(J)
G(M+J-2)=C(H+J-2)/W(M+J-2)
D(M+J-2)=A(H+J-1)
IF (J.GT.M-2) GO TO 22
7 H(I,J)=D(I+J-2)*H(I-1,J)/W(I+J-1)
22 IF (J.EQ.1) GO TO 23
IF (J.LT.M-1) IP=2
IF (J.LE.M-1) IP=M-J+1
DO 8 I=IP,M-1
Z=0
9 Z=Z+E(K,J)*H(K-I+M,J-M)
E(I,J)=G(I+J-2)*E(I-1,J)-Z
Z=0
DO 12 K=1,I-1
Z=Z+E(K,I)*H(K,J)
8 CONTINUE
23 I=M-1
Z=0
DO 13 K=1,I
Z=Z+E(K,J)*H(K,J)
13 CONTINUE
W(M+J-1)=B(M+J-1)-D(I+J-1)*H(I,J)-G(I+J-1)*E(I,J)-D(I+J-1)*
*G(1+J-1)-Z
6 CONTINUE
SUBROUTINE FBSUBS(N,M,S)
  
  C  THIS SUBROUTINE SOLVES THE SET (LU)Y=S OF N LINEAR EQUATIONS.
  C  WHERE L, U (WITH UNIT DIAGONAL ELEMENTS) ARE STRICTLY LOWER
  C  AND UPPER TRIANGULAR MATRICES OF BANDWIDTH M, THE NON-ZERO
  C  ELEMENTS ARE ON SUB, SUPER DIAGONAL ELEMENTS RESPECTIVELY AND
  C  RETAIN M=M-1 OUTERMOST OFF-DIAGONAL ENTRIES. THE SOLUTION
  C  IS EFFECTED BY A FORWARD-BACKWARD SUBSTITUTION PROCESS WHERE
  C  THE INPUT VECTOR S IS OVERWRITTEN SUCCESSIVELY BY THE
  C  INTERMEDIATE SOLUTION (OBTAINED BY FORWARD SUBSTITUTION) AND
  C  THE FINAL SOLUTION (OBTAINED BY BACK SUBSTITUTION). N1=N=M+1.
  C
  DIMENSION S(N)
  COMMON /BLOCK2/G(N),W(N),B(N)
  COMMON /BLOCK4/T(M1,N1),E(M1,N1)
  
  S(1)=S(1)/W(1)
  DO 7 I=M+1
  7 S(I)=(S(I)-B(I-1)*S(I-1))/W(I)
  DO 8 I=1,N
  8 Z=0
  DO 9 K=I-M-1,1,-1
  9 Z=Z+B(K-1*M,I-I+1)*S(K)
  8 S(I)=(S(I)-B(I-1)*S(I-1)-Z)/W(I)
  
  IF (M.LT.N/2) GO TO 34
  IF (M.EQ.N) GO TO 35
  IF ((M.EQ.N/2)) GO TO 36
  IF ((M,EQ.N/2)) GO TO 37
  
  CASE : M=[N/2+1].
  DO 10 I=1,N-M
  10 I=N-I
  Z=0
  DO 11 J=I+1,N
  11 Z=Z+T(I-J*M,J-M+1)*S(J)
  10 CONTINUE
  S(I)=S(I)-G(I)*S(I+1)-Z
  11 CONTINUE
  DO 12 I=1,2*M-N-1
  I=N-I
  Z=0
  DO 13 J=M,N
  13 Z=Z+T(I-J*M,J-M+1)*S(J)
  12 CONTINUE
  S(I)=S(I)-G(I)*S(I+1)-Z
  13 CONTINUE
  DO 14 I=1,N-M
  I=N-M-I
  Z=0
  DO 15 J=M+1
  15 Z=Z+T(I-J*M,J-M+1)*S(J)
  14 CONTINUE
  S(I)=S(I)-G(I)*S(I+1)-Z
  16 CONTINUE
  IF (((N-N/2+2).EQ.0).AND.((M,EQ.N/2))) GO TO 37
  
  CASE : M=[N/2+1].
  DO 16 I=1,M-1
  I=N-I
  Z=0
  DO 17 J=I+1,N
  17 Z=Z+T(I-J*M,J-M+1)*S(J)
  16 CONTINUE
  RETURN
END
16 \[ S(I) = S(I) - G(I) \times S(I+1) - Z \]
DO 18 I=1,N-2,M+1
17 I=N-M+1,II
Z=0
DO 19 J=I+1,II
18 Z= Z+T(I-J+1,J-M+1)*S(J)
CONTINUE
19 S(I) = S(I) - G(I) \times S(I+1) - Z
DO 20 II=1,M-1
I=M-II
Z=0
DO 21 J=M,II
20 Z= Z+T(I-J+1,J-M+1)*S(J)
CONTINUE
21 S(I) = S(I) - G(I) \times S(I+1) - Z
GO TO 36

CASE : N IS EVEN AND M=N/2.
37 DO 22 II=1,N-M-1
I=N-II
Z=0
DO 23 J=I+1,N
22 Z= Z+T(I-J+1,J-M+1)*S(J)
I=M
Z=0
DO 24 J=M+1,N-1
23 Z= Z+T(I-J+1,J-M+1)*S(J)
S(I) = S(I) - G(I) \times S(I+1) - Z
DO 25 II=1,M-1
I=M-II
Z=0
DO 26 J=M,II
24 Z= Z+T(I-J+1,J-M+1)*S(J)
25 S(I) = S(I) - G(I) \times S(I+1) - Z
CONTINUE
RETURN
END


INTEGER P
COMMON /BLOCK2/G(N), D(N), W(N)
COMMON H(M1,N1), T(M2,N2), E(M1,N1), F(M2,N2)

W(1)=B(1)
D(1)=A(2)
C(1)=C(1)/W(1)
DU 5 I=2, M-2
W(I)=B(I)-D(I-1)*G(I-1)
D(I)=A(I+1)
G(I)=C(I)/W(I)
W(M-1)=B(M-1)-D(M-2)*G(M-2)
DU 6 J=1, P-1
E(I,J)=V(J+M-1)
H(I,J)=U(J)/W(J)
G(M+J-2)=C(M+J-2)/W(M+J-2)
D(M+J-2)=A(M+J-1)
IF (J, GT, M-2) GO TO 4
DU 7 I=2, M-J
E(I,J)=-G(I+J-2)*E(I-1,J)
H(I,J)=-D(I+J-2)*H(I-1,J)/W(I+J-1)
IF (J, LE, M-1) IP=2
IF (J, LE, M-1) IP=M-J+1
DU 8 I=IP, M-1
Z=0
DU 3 K=1, I-1
Z=Z+E(K,I)*H(K+I-1,J)
E(I,J)=-G(I+J-2)*E(I-1,J)-Z
Z=0
DU 12 K=1, I-1
Z=Z+E(K+I-1,J)*H(K+I-1,J)
H(I,J)=-D(I+J-2)*H(I-1,J)-Z)/W(I+J-1)
CONTINUE
18 I=M-1
Z=0
DU 13 K=1, I
Z=Z+H(K,J)*E(K,J)*H(K,J)
W(M+J-1)=B(M+J-1)-D(I+J-1)*H(I,J)*G(I+J-1)*E(I,J)-D(I+J-1)*G(I+J-1)+Z
CONTINUE
6 GENERATION OF ELEMENTS OF SUBMATRICES U12, L21.
DU 22 J=1, N-P+1
SUBROUTINE FB/SUBS3D(N,P,M,S)
THIS SUBROUTINE SOLVES THE SET (I)Y=S OF N LINEAR EQUATIONS,
WHERE L,U (WITH UNIT DIAGONAL ELEMENTS) ARE STRICTLY LOWER
AND UPPER TRIANGULAR MATRICES OF BANDWIDTHS M AND P, THE
NON-ZERO ELEMENTS ARE ON SUB/SUPER DIAGONAL ELEMENTS
RESPECTIVELY, RETAINING M1=M-1 AND M2=P-1 OUTERMOST OFF-
DIAGONAL ENTRIES. THE SOLUTION IS EFFECTED BY A FORWARD-
BACKWARD SUBSTITUTION PROCESS WHERE THE INPUT VECTOR S IS
OVERWRITTEN SUCCESSIVELY BY THE INTERMEDIATE SOLUTION
(OBTAINED BY FORWARD SUBSTITUTION) AND THE FINAL SOLUTION
(OBTAINED BY BACK SUBSTITUTION). N1=N=M+1 , N2=N-P+1.
DIMENSION S(N)
COMMON /BLOCK2/GS(N),B(N),W(N)
COMMON H(M1,N1),T(M2,N2),E(M1,N1),F(M2,N2)
INTEGER P
S(1)=S(1)/W(1)
44 S(I)=(S(I)-B(I-1)*S(I-1))/W(I)
IF (M.EQ.N) GO TO 396
DO 78 I=M+1, P-1
Z=0
DO 79 K=I-M+1, I-1
Z=Z+E(K-I*M, I-M+1)*S(K)
78 S(I)=(S(I)-B(I-1)*S(I-1)-Z)/W(I)
396 DO 87 I=P, N
Z=0
DO 88 K=I-M+1, I-1
88 Z=Z+E(K-I*M, I-M+1)*S(K)
Z1=0
DO 89 K=I-P+1, I+1
89 Z1=Z1+E(K-I*M, I-M+1)*S(K)
S(I)=(S(I)-B(I-1)*S(I-1)-Z1)/W(I)
IF (M.LT.(N/2+1)) GO TO 64
IF (M.EQ.N) GO TO 65
IF (P.GT.(N/2+1)) GU TO 183
CASE M>N/2+1], P DOES NOT EXIST
DO 47 I=1, N-M
I=N-M
Z=0
DO 48 J=I+1, N
48 Z=Z+H(I-J*N, J-M+1)*S(J)
47 S(I)=S(I)-GS(I)*S(I+1)-Z
65 DO 49 I=1,2*M-N-1
I=M+1
Z=0
DO 51 J=I+1, N
51 Z=Z+H(I-J*N, J-M+1)*S(J)
49 S(I)=S(I)-GS(I)*S(I+1)-Z
IF (M.EQ.N) GO TO 66
DO 52 I=1, N-M
I=N-M+1-I
Z=0
DO 53 J=I+1, N
53 Z=Z+H(I-J*N, J-M+1)*S(J)
52 S(I)=S(I)-GS(I)*S(I+1)-Z
GU TO 66
CASE : M>IN/2+1 ] , P>IN/2+1 ]
183 IF (P.EQ.N) GO TO 252
DU 231 I=1, N-P
I = N - 11
Z = 0
Z1 = 0
DO 232 J = I + 1, N
Z = Z + H(I - J + M, J - M + 1) * S(J)
232 Z1 = Z1 + T(I - J + P, J - P + 1) * S(J)
S(I) = S(I) - G5(I) * S(I + 1) - Z - Z1
IF (M, EQ, N) GO TO 253
DO 233 II = I, P - M
I = P - 1
Z = 0
DO 234 J = I + 1, N
Z = Z + H(I - J + M, J - M + 1) * S(J)
Z1 = 0
DO 235 J = M, N
Z1 = Z1 + T(I - J + P, J - P + 1) * S(J)
S(I) = S(I) - G5(I) * S(I + 1) - Z - Z1
IF (M, EQ, N) GO TO 66
DO 238 II = M, P - 1
I = N - II
Z = 0
Z1 = 0
DO 237 J = M, N
Z = Z + H(I - J + M, J - M + 1) * S(J)
Z1 = Z1 + T(I - J + P, J - P + 1) * S(J)
S(I) = S(I) - G5(I) * S(I + 1) - Z - Z1
IF (M, EQ, N) GO TO 66
DO 242 II = P, N - 1
I = N - II
Z = 0
Z1 = 0
DO 243 J = M, I + M - 1
Z = Z + H(I - J + M, J - M + 1) * S(J)
Z1 = Z1 + T(I - J + P, J - P + 1) * S(J)
S(I) = S(I) - G5(I) * S(I + 1) - Z - Z1
64 IF (P, GT, N, OR, P, LE, M) GO TO 821
IF (P, LT, (N - 2 + 1)) GO TO 444
IF ((N - N/2 * 2) .EQ, 0) .AND. (M, EQ, N/2)) GO TO 67
CASE: M < [N/2 + 1] ; p >= [N/2 + 1].
IF (M, EQ, N) GO TO 254
DO 94 II = 1, N - P
I = N - II
Z = 0
Z1 = 0
DO 95 J = I + 1, N
Z1 = Z1 + T(I - J + P, J - P + 1) * S(J)
95 Z = Z + H(I - J + M, J - M + 1) * S(J)
S(I) = S(I) - G5(I) * S(I + 1) - Z - Z1
DO 96 II = N - P + 1, M - 1
I = N - II
z = 0
DO 97 J = I + 1, N
97 Z = Z + H(I - J + M, J - M + 1) * S(J)
Z(1) = 0
DO 98 J = P, N
98 Z = Z + T(I - J + P, J = P + 1) * S(J)
S(I) = S(I) - GS(I) * S(I + 1) - Z - Z(1)
DO 101 II = M, N - M
I = N - II
Z = 0
DO 102 J = I + 1, I + 1
102 Z = Z + H(I - J + M, J - M + 1) * S(J)
Z(1) = 0
DO 103 J = P, N
103 Z = Z + T(I - J + P, J = P + 1) * S(J)
S(I) = S(I) - GS(I) * S(I + 1) = Z - Z(1)
DO 104 II = N - M + 1, P - 1
I = N - II
Z = 0
DO 105 J = M, I + M - 1
105 Z = Z + H(I - J + M, J - M + 1) * S(J)
Z(1) = 0
DO 106 J = P, N
106 Z = Z + T(I - J + P, J = P + 1) * S(J)
S(I) = S(I) - GS(I) * S(I + 1) - Z - Z(1)
IF (P, EQ, N) GO TO 66
DO 121 II = P, N - 1
I = N - II
Z = 0
DO 122 J = M, I + M - 1
122 Z = Z + H(I - J + M, J - M + 1) * S(J)
Z(1) = 0
DO 123 J = P, I + P - 1
123 Z = Z + T(I - J + P, J = P + 1) * S(J)
S(I) = S(I) - GS(I) * S(I + 1) - Z - Z(1)
GO TO 66
CASE : M = N/2 , P > [N/2 + 1] .
67 IF (P, EQ, N) GO TO 256
DO 61 II = 1, N - P
I = N - II
Z = 0
Z = 1
DO 62 J = I + 1, N
Z = Z + H(I - J + M, J - M + 1) * S(J)
62 Z = Z + T(I - J + P, J = P + 1) * S(J)
61 S(I) = S(I) - GS(I) * S(I + 1) - Z - Z(1)
256 DO 141 II = 1, P - M - 1
I = P - II
Z = 0
DO 142 J = I + 1, N
142 Z = Z + H(I - J + M, J - M + 1) * S(J)
Z(1) = 0
DO 145 J = P, N
145 Z = Z + T(I - J + P, J = P + 1) * S(J)
141 S(I) = S(I) - GS(I) * S(I + 1) - Z - Z(1)
I = M
Z = 0
DO 146 J = M + 1, N - 1
146 Z = Z + H(I - J + M, J - M + 1) * S(J)
Z(1) = 0
DO 147 J=P,N
147 Z1=Z1+T(I-J+P, J-P+1)*S(J)
S(I)=S(I)-GS(I)*S(I+1)-Z-Z1
DO 146 II=1,M+P-N+1
I=M-1
Z=0
DO 149 J=M,1+M-1
149 Z=Z+H(I-J+M, J-M+1)*S(J)
Z1=0
DO 150 II=M,1+M-1
Z=0
DO 156 II=M,1+M-1
Z1=Z1+T(I-J+P, J-P+1)*S(J)
S(I)=S(I)-GS(I)*S(I+1)-Z-Z1
IF (P.EQ.N) GO TO 66
DO 152 II=P,N-1
I=N-1
Z=0
DO 156 II=M,1+M-1
Z1=Z1+T(I-J+P, J-P+1)*S(J)
S(I)=S(I)-GS(I)*S(I+1)-Z-Z1
GO TO 66
CASE: M<=N/2+1, P<=(N/2+1)
GO TO 107
C
IF (((N-N/2-Z).EQ.0).AND.((P.EQ.N/2)) GO TO 107
P=I+1, Q=N,
P1=I+1, Q1=N
DO 54 II=1,M-1
I=N-1
Z=0
Z1=0
DO 55 J=M+1,N
Z=Z+H(I-J+M, J-M+1)*S(J)
Z1=Z1+T(I-J+P, J=P+1)*S(J)
CONTINUE
54 S(I)=(S(I)-GS(I)*S(I+1)-Z-Z1)
DO 56 II=M, P-1
I=N-1
Z=0
DO 57 J=M+1,N
Z=Z+H(I-J+M, J-M+1)*S(J)
CONTINUE
Z1=0
DO 91 J=M+1,N
Z1=Z1+T(I-J+P, J=P+1)*S(J)
S(I)=S(I)-GS(I)*S(I+1)-Z-Z1
DO 124 II=P,N-1
I=N-1
Z=0
DO 125 J=M+1,N
Z=Z+H(I-J+M, J-M+1)*S(J)
Z1=0
DO 126 J=M+1, I+1
Z1=Z1+T(I-J+P, J=P+1)*S(J)
S(I)=S(I)-GS(I)*S(I+1)-Z-Z1
DO 124 II=N, P+1
I=N-1
Z=0
DO 128 J=M+1, I+1
Z=Z+H(I-J+M, J-M+1)*S(J)
Z1=0
**Cf--**  
CASE:  \( M \leq \lfloor N/2+1 \rfloor, P = N/2 \)

```plaintext
DO 107 I = 1, N - P - 1  
       Z = 0  
       Z1 = 0  
       DO 109 J = I + 1, N  
          Z = Z + H(I - J + P, J - P + 1) * S(J)  
          Z1 = Z1 + T(I - J + P, P - 1) * S(J)  
       END  
       DO 111 J = I + 1, N  
          Z = Z + H(I - J + M, J - M + 1) * S(J)  
          Z1 = Z1 + T(I - J + P, P + 1) * S(J)  
       END  
       Z = 0  
       Z1 = 0  
       DO 114 J = I + 1, N  
          Z = Z + H(I - J + M, J - M + 1) * S(J)  
          Z1 = Z1 + T(I - J + P, P - 1) * S(J)  
       END  
       Z = 0  
       Z1 = 0  
       DO 117 J = P, I + P - 1  
          Z = Z + H(I - J + P, J - P + 1) * S(J)  
          Z1 = Z1 + T(I - J + P, P - 1) * S(J)  
       END  
       Z = 0  
       Z1 = 0  
       DO 120 J = M + P, I + P - 1  
          Z = Z + H(I - J + P, J - P + 1) * S(J)  
          Z1 = Z1 + T(I - J + P, P - 1) * S(J)  
       END  
GO TO 66  
```

**Cf--**  
CASE:  \( M \leq \lfloor N/2+1 \rfloor \), \( P \) does not exist

```plaintext
DO 381 II = 1, M - 1  
```
I = N - 1
Z = 0

DO 382 J = I + 1, N
382 Z = Z + H(I - J + M, J - M + 1) * S(J)
383 S(I) = S(I) - GS(I) * S(I + 1) - Z

DO 385 I = I, N - 2 + M + 1
I = N - M + 1 - I
Z = 0

DO 384 J = I + 1, I + M - 1
384 Z = Z + H(I - J + M, J - M + 1) * S(J)
385 S(I) = S(I) - GS(I) * S(I + 1) - Z

DO 386 I = I, M - 1
I = M - I
Z = 0

DO 387 J = M, I + M - 1
387 Z = Z + H(I - J + M, J - M + 1) * S(J)
388 S(I) = S(I) - GS(I) * S(I + 1) - Z
GO TO 66

CASE: M = N/2, P DOES NOT EXIST.

DO 388 I = I, N - M - 1
I = N - I
Z = 0

DO 389 J = I + 1, N
389 Z = Z + H(I - J + M, J - M + 1) * S(J)
389 S(I) = S(I) - GS(I) * S(I + 1) - Z

DO 391 J = I + 1, N - 1
391 Z = Z + H(I - J + M, J - M + 1) * S(J)
391 S(I) = S(I) - GS(I) * S(I + 1) - Z

DO 392 I = I, M - 1
I = M - I
Z = 0

DO 393 J = M, I + M - 1
393 Z = Z + H(I - J + M, J - M + 1) * S(J)
393 S(I) = S(I) - GS(I) * S(I + 1) - Z

CONTINUE
RETURN
END
SUBROUTINE NB3D(N, M, P, A, B, C, H, S)
C THIS SUBROUTINE SOLVES THE SET WX = S OF N LINEAR EQUATIONS
C (N > 4) WHERE THE COEFFICIENT MATRIX IS SYMMETRIC, POSITIVE
C A, B, C, H, S ARE VECTORS CONTAINING RESPECTIVELY THE
C DIAGNOSTIC, CO DIAGNOSTIC, M-TH DIAGNOSTIC, P-TH DIAGNOSTIC AND
C CONSTANT ELEMENTS. THE MATRIX IS FACTORIZED INTO --DTTD--
C WHERE D IS A DIAGNOSTIC MATRIX AND T IS A REAL, UPPER TRIANGULAR
C MATRIX WITH UNIT DIAGNOSTIC ELEMENTS, NON-ZERO ELEMENTS IN
C SUPER DIAGNOSTIC AND RETAINING R1 = M-1 AND R2 = P-1 OUTERMOST
C OFF-DIAGNOSTIC ENTRIES. T^T DENOTES THE TRANSPOSE OF T. THE
C INPUT VECTORS A, B ARE OVERWRITTEN DURING THE FACTORIZATION
C STAGE SO THAT VECTOR A CONTAINS THE DIAGNOSTIC ELEMENTS OF
C MATRIX D AND VECTOR B CONTAINS THE CO DIAGNOSTIC ELEMENTS OF
C MATRIX T. TWO RESULT VECTORS U, D OF LENGTH N AND TWO ARRAYS
C OF DIMENSION [P-1, N-M+1] ARE USED AS WORKSPACE. THE SOLUTION
C IS EFFECTED BY A FORWARD-BACKWARD SUBSTITUTION PROCESS. THE
C NORMALIZED R.H.S. VECTOR IS OVERWRITTEN SUCCESSIVELY BY THE
C INTERMEDIATE SOLUTION (OBTAINED BY FORWARD SUBSTITUTION) AND
C THE FINAL SOLUTION (OBTAINED BY BACK SUBSTITUTION). THE
C VECTORS A, B, C, H, S ARE GENERATED BY THE CALLING SUBROUTINE
C GEVECT. N1 = P-1, N2 = N-M+1, N3 = N-P+1.
C DIMENSION A(N), B(N), C(N2), H(N3), S(N), D(N), U(N)
C DIMENSION F(N, N2), G(N, N2)
C INTEGER P
C CALL GEVECT(N, P, M, A, B, C, H, S)
C TEST FOR THE STRUCTURE OF THE COEFFICIENT MATRIX.
C IF (M. NE. N) GO TO 871
C MM = N
MP = N-M+1
GO TO 173
871 W = M
MM = 1
IC = 0
DO 151 I = 1, N, N-M+1
IF (C(I), EQ, 0) IC = IC + 1
151 CONTINUE
IND = 0
DO 171 I = 1, N, P+1
IF (H(I), EQ, 0) IND = IND + 1
171 CONTINUE
IS = IC + IND
ISUM = 2 * N - P + 1
IF (IS, EQ, ISUM) GO TO 999
IF (IC, NE, N-M+1) GO TO 172
CASE: NORMBAND WITH BANDWIDTH P.
MP = N-P+1
IP = P
IRG = IP
MP = N-P+1
DO 153 I = 1, N-P+1
153 C(I) = H(I)
GO TO 173
CASE: NORMBAND WITH BANDWIDTH M.
MP = N-M+1
C STAGE I: FACTORIZATION.

173 \(A(1) = \text{SQRT}(A(1))\)
DO 1 \(I = 2, \ldots, N\)
Z = B\((I-1)/A(I-1)\)
A\((1) = \text{SQRT}(A(1) - Z^2)\)
1 B\((I-1) = Z/A(I)\)
IF (N.EQ.N) GO TO 177
DO 2 \(I = 2, \ldots, M\)
Z = Z + B/(A(I) - U(I-1) * U(I-1))
2 U\((I) = -B(I+J-2) * U(I-1)\)
IF (J.EQ.1) GO TO 8
IF (J.GT.1) GO TO 652
DO 3 \(I = 1, \ldots, P\)
Z = Z + U(I) * F(I+J-2, I+J-1)
3 U\((I) = -B(I+J-2) * U(I-1) - Z\)
GO TO 24

562 G(I,J) = M(I,J) / (A(J) * A(J) * A(J-1))
V = B/P(J-2, A(P+J-2) * A(P+J-1))
IF (J.GT.(H-2)) GO TO 11
DO 33 \(I = 2, \ldots, M\)
G(I,J) = B(I+J-2) * G(I-1, J)
IF (J.EQ.1, AND, I.EQ.I) G(I,J) = G(I,J) + C(I+J-1) / (A(I+J-1) * A(I+J-1))
GO TO 24
CONTINUE

C GENERATION OF ELEMENTS OF MATRICES U AND R (FROM THE
C PARTITIONED COEFFICIENT MATRIX A).
31 IF (J.GT.(H-1)) IP = 2
IF (J.LE.(H-1)) IP = M - J + 1
DO 16 \(I = 1, \ldots, P\)
IF (I.GE.(P-J) OR, J.GT.P-2) GO TO 654
Z = Z + G(K+I-1, J) * F(K, I+J-1)
17 Z = Z + G(K+I-1, J) * F(K, I+J-1)
GO TO 24
23 DO 26 \(K = 1, \ldots, P\)
Z = Z + G(K, J) * F(K, I+J-1)
26 Z = Z + G(K, J) * F(K, I+J-1)
24 G(I,J) = -B(I+J-2) * G(I-1, J) - Z
IF (I.EQ.I1) G(I,J)=G(I,J)+C(I+J-1)/(A(I+J-1)*A(P+J-1))  
GO TO 16

654 Z=0
60 DO 21 K=1,P-2
21 Z=Z+G(K-I*P, I+J-P)*G(K, J)
   G(I,J)=G(I,J)+C(I+J-1)/(A(I+J-1)*A(P+J-1))  
   GO TO 16

CONTINUE
I=I+1
W=V+G(I,J)

15 Z=0
6 DO 161 K=1,P-2
161 Z=Z+G(K, J)*G(K, J)
   A(P+J-1)=SQRT(D(P+J-1)/(1+Z+W**2))
   IF (ABS(A(P+J-1)-U(P+J-1)),LT.1.0E-12) GO TO 191
   U(P+J-1)=A(P+J-1)
GO TO 562

191 IF (J.EQ.0) U(P-1)=D(P-1)
   U(P+J-1)=A(P+J-1)
   B(P+J-2)=B(P+J-2)*(A(P+J-2)*A(P+J-1))
   A(P-1)=D(P-1)

656 IF (I.GT.N-W+1) GO TO 311
   M=W
   DO 314 I=1,P-1
314 M=M+G(I,J)
   DO 314 J=1,N-M+1
314 F(I,J)=0

STAGE II: NORMALIZATION.

311 DO 37 I=1,N
37 S(I)=S(I)/A(I)

STAGE III: SOLUTION (FORWARD-BACKWARD SUBSTITUTION).

44 DO 44 I=2,H+1
44 S(I)=S(I-1)+S(I-1)
   IF (I.EQ.N) GO TO 396
   Z=0
   DO 79 K=I-M+1,H+1
79 Z=Z+F(K-I+H+1)*S(K)
   S(I)=S(I-1)+S(I-1)-Z
396 DO 88 I=P,N
38 DO 88 I=1,P-1
88 Z=Z+F(K-I+H+1)*S(K)
Z=0

CASE: M>(N/2+1), P DOES NOT EXIST

DO 47 II=1,N-M
47 S(I)=S(I)-B(I)*S(I)
48 Z=Z+F(I-J+H+1)*S(J)
47 S(I)=S(I)-B(I)*S(I-1)-Z
65 DO 49 II=1,2*M-N+1
   I=M-II
   Z=0

DO 51 J=M+N
   Z=Z+F(I-J+M,J-M+1)*S(J)
51 S(I)=S(I)+B(I)*S(I+1)-Z
   IF (H, EQ, N) GO TO 66
   DU 52 II=I, M=M-1
   I=N-M+1-1
   Z=0
   ZI=0
   DU 53 J=M+1
   Z=Z+F(I-J+M,J-M+1)*S(J)
52 S(I)=S(I)+B(I)*S(I+1)-Z
   GO TO 66
6 CASE : I I>=[N/2]+1; P>[N/2]+1
183 IF (P, EQ, N) GO TO 252
   DU 231 II=1, N=P
   I=N-1
   Z=0
   ZI=0
   DO 232 J=I+1, N
   Z=Z+F(I-J+H,J-M+1)*S(J)
232 ZI=ZI+G(I-J+P,J-P+1)*S(J)
231 S(I)=S(I)+B(I)*S(I+1)-Z-Z1
252 IF (H, EQ, N) GO TO 253
   DU 233 II=1, P-M
   I=P-I
   Z=0
   ZI=0
   DO 234 J=I+1, N
   Z=Z+F(I-J+H,J-M+1)*S(J)
234 ZI=ZI+G(I-J+P,J-P+1)*S(J)
   ZI=ZI+G(I-J+P,J-P+1)*S(J)
233 S(I)=S(I)+B(I)*S(I+1)-Z-Z1
253 DO 236 II=N-M+1, M-1
   I=N-1
   Z=0
   ZI=0
   DO 237 J=M, N
   Z=Z+F(I-J+H,J-M+1)*S(J)
237 ZI=ZI+G(I-J+P,J-P+1)*S(J)
236 S(I)=S(I)+B(I)*S(I+1)-Z-Z1
   IF (M, EQ, N) GO TO 66
   DU 238 II=1, P-1
   I=N-1
   Z=0
   ZI=0
   DO 239 J=I+1, N
   Z=Z+F(I-J+H,J-M+1)*S(J)
239 ZI=ZI+G(I-J+P,J-P+1)*S(J)
   ZI=0
   DO 241 J=M, N
241 ZI=ZI+G(I-J+P,J-P+1)*S(J)
238 S(I)=S(I)+B(I)*S(I+1)-Z-Z1
   IF (P, EQ, N) GO TO 66
   DO 242 II=P, N-1
   I=N-1
   Z=0
   ZI=0
   DO 243 J=N+1, I+H-1
   Z=Z+F(I-J+H,J-M+1)*S(J)
243 ZI=ZI+G(I-J+P,J-P+1)*S(J)
242 S(I)=S(I)+B(I)*S(I+1)-Z-Z1
   GO TO 66
64 IF (P, GT, N, OR, P, LE, M) GO TO 821
IF (P, LT, (N/2+1)) GO TO 444
IF ((C(N-N/2*2).EQ.0).AND.(M.EQ.N/2)) GO TO 67

CASE: M<\(\lfloor N/2+1 \rfloor \), \(P>\lfloor N/2+1 \rfloor \).

CASE: M<\(\lfloor N/2+1 \rfloor \), \(P>\lfloor N/2+1 \rfloor \).

IF (P.EQ.N) GO TO 254
DO 94 II=1, N=P
I=N-II
Z=0
Z1=0
DO 95 J=I+1, N
Z=Z+F(I-J+M,J-M+1)*S(J)
95 Z=Z+F(I-J+M,J-M+1)*S(J)
94 S(I)=S(I)-B(I)*S(I+1)-Z-Z1
254 DO 96 II=N, P+1; M=1
I=N-II
Z=0
DO 97 J=I+1, N
Z=Z+F(I-J+M,J-M+1)*S(J)
Z1=0
DO 98 J=P, N
Z1=Z1+G(I-J+P,J=M+1)*S(J)
98 S(I)=S(I)-B(I)*S(I+1)-Z-Z1
97 Z=Z+F(I-J+M,J-M+1)*S(J)
96 DO 101 II=N, H-M
I=N-II
Z=0
DO 102 J=I+1, I+1
102 Z=Z+F(I-J+M,J-M+1)*S(J)
101 Z1=0
DO 103 J=P, N
Z1=Z1+G(I-J+P,J=M+1)*S(J)
103 S(I)=S(I)-B(I)*S(I+1)-Z-Z1
102 Z=Z+F(I-J+M,J-M+1)*S(J)
101 Z1=0
DO 105 J=M+1, I+M-1
105 Z=Z+F(I-J+M,J-M+1)*S(J)
104 Z1=0
DO 106 J=P, N
Z1=Z1+G(I-J+P,J=M+1)*S(J)
106 S(I)=S(I)-B(I)*S(I+1)-Z-Z1
105 Z=Z+F(I-J+M,J-M+1)*S(J)
104 IF (P.EQ.N) GO TO 66
DO 121 II=P, N-1
I=N-II
Z=0
DO 122 J=M+1, H-1
122 Z=Z+F(I-J+M,J-M+1)*S(J)
121 Z1=0
DO 123 J=P, I+P-1
123 Z1=Z1+G(I-J+P,J=M+1)*S(J)
122 S(I)=S(I)-B(I)*S(I+1)-Z-Z1
121 GO TO 66

CASE: M=\(\lfloor N/2 \rfloor \), \(P>\lfloor N/2+1 \rfloor \).

IF (P.EQ.N) GO TO 256
DO 61 II=1, N-P
I=N-II
Z=0
Z1=0
DO 62 J=I+1, N
Z=Z+F(I-J+M,J-M+1)*S(J)
61 S(I)=S(I)-B(I)*S(I+1)-Z-Z1
60 Z1=Z1+G(I-J+P,J=M+1)*S(J)
62 S(I)=S(I)-B(I)*S(I+1)-Z-Z1
61 IF (P.EQ.N) GO TO 256
DO 61 II=1, N-P
I=N-II
Z=0
Z1=0
DO 62 J=I+1, N
Z=Z+F(I-J+M,J-M+1)*S(J)
60 Z1=Z1+G(I-J+P,J=M+1)*S(J)
256  DO 141 I=1,P-M-1
    I=P-11
    Z=0
  142  DO 142 J=1+1,N
    Z=Z+F(I-J+M,J-M+1)*S(J)
    Z1=0
    DO 145 J=P,N
  145  Z1=Z1+G(I-J+P,J-P+1)*S(J)
  141  S(I)=S(I)-B(I)*S(I+1)-Z-Z1
    I=H
    Z=0
  146  DO 146 J=M+1,N-1
    Z=Z+F(I-J+M,J-M+1)*S(J)
    Z1=0
    DO 147 J=P,N
  147  Z1=Z1+G(I-J+P,J-P+1)*S(J)
  148  S(I)=S(I)-B(I)*S(I+1)-Z-Z1
    IF (P.EQ.N) GO TO 60
    DU 154 II=P,N-1
    I=N-II
    Z=0
  149  DO 149 J=M,N-H-1
    Z=Z+F(I-J+M,J-M+1)*S(J)
    Z1=0
    DO 158 J=1+N
  158  Z1=Z1+G(I-J+P,J-P+1)*S(J)
  144  IF (((N-N/2*2). EQ, O). AND. ( P. EO. N/2)) GO TO 107
    DO 54 II=1,II-1
    I=N-II
    Z=0
    Z1=0
  55  DO 55 J=1+1,N
    Z=Z+F(I-J+M,J-M+1)*S(J)
    Z1=0
    Z=Z+F(I-J+M,J-M+1)*S(J)
  54  S(I)=S(I)-B(I)*S(I+1)-Z-Z1
    DO 56 II=M-1
    P=1
    I=H
    Z=0
  56  DO 57 J=1+1,I=N-1
    Z=Z+F(I-J+M,J-M+1)*S(J)
    Z1=0
    Z=Z+F(I-J+M,J-M+1)*S(J)
  50  S(I)=S(I)-B(I)*S(I+1)-Z-Z1
    DO 124 II=P,N-1
    I=N-II
    Z=0
  125  J=I+1,I=N-1

C--- CASE: M<(N/2+1), P<(N/2+1).
  444  IF ((N-N/2*2).EQ, O). AND, ( P. EO. N/2)) GO TO 107
    DU 54 II=1,II-1
    I=N-II
    Z=0
    Z1=0
  55  DO 55 J=1+1,N
    Z=Z+F(I-J+M,J-M+1)*S(J)
    Z1=0
    Z=Z+F(I-J+M,J-M+1)*S(J)
  54  S(I)=S(I)-B(I)*S(I+1)-Z-Z1
    DO 56 II=M-1
    P=1
    I=H
    Z=0
  56  DO 57 J=1+1,I=N-1
    Z=Z+F(I-J+M,J-M+1)*S(J)
    Z1=0
    Z=Z+F(I-J+M,J-M+1)*S(J)
  50  S(I)=S(I)-B(I)*S(I+1)-Z-Z1
    DO 124 II=P,N-1
    I=N-II
    Z=0
  125  J=I+1,I=N-1
125 \[ Z = Z + F(I - J + M, J - M + 1) * S(J) \]
\[ Z1 = 0 \]
DO 126 J=I+1, I+ P=1
126 \[ Z1 = Z1 + G(I - J + P, J + P + 1) * S(J) \]
124 \[ S(I) = S(I) - B(I) * S(I+1) - Z - Z1 \]
DO 127 II=N- P+1, N=H
I=N-1
Z=0
DO 128 J=I+1, I+ M=1
128 \[ Z = Z + F(I - J + M, J - M + 1) * S(J) \]
\[ Z1 = 0 \]
DO 129 J= P, I+ P=1
129 \[ Z1 = Z1 + G(I - J + P, J + P + 1) * S(J) \]
127 \[ S(I) = S(I) - B(I) * S(I+1) - Z - Z1 \]
DO 58 II=1, II=1
I=M-1
Z=0
DO 59 J=M-1, I+ M=1
59 \[ Z = Z + F(I - J + M, J - M + 1) * S(J) \]
\[ Z1 = 0 \]
DO 93 J= P, I+ P=1
93 \[ Z1 = Z1 + G(I - J + P, J + P + 1) * S(J) \]
58 \[ S(I) = S(I) - B(I) * S(I+1) - Z - Z1 \]
GO TO 66
CR----- CASE : M < [N/2+1], P=N/2
107 DO 108 II=1, N= P+1
I=N-1
Z=0
108 \[ Z = Z + F(I - J + M, J - M + 1) * S(J) \]
\[ Z1 = 0 \]
DO 109 J=M-1, I+ M=1
109 \[ Z = Z + F(I - J + M, J - M + 1) * S(J) \]
107 \[ S(I) = S(I) - B(I) * S(I+1) - Z - Z1 \]
I=P
Z=0
DO 110 J=M-1, I+ M=1
110 \[ Z = Z + F(I - J + M, J - M + 1) * S(J) \]
\[ Z1 = 0 \]
DO 112 J= P+1, N=1
112 \[ Z1 = Z1 + G(I - J + P, J + P + 1) * S(J) \]
111 \[ S(I) = S(I) - B(I) * S(I+1) - Z - Z1 \]
DO 113 II=N- P+1, N=H-1
I=N-1
Z=0
DO 114 J=M-1, I+ M=1
114 \[ Z = Z + F(I - J + M, J - M + 1) * S(J) \]
\[ Z1 = 0 \]
DO 115 J= P, I+ P=1
115 \[ Z1 = Z1 + G(I - J + P, J + P + 1) * S(J) \]
113 \[ S(I) = S(I) - B(I) * S(I+1) - Z - Z1 \]
I=M
Z=0
DO 116 J=M+1, N=1
116 \[ Z = Z + F(I - J + M, J - M + 1) * S(J) \]
\[ Z1 = 0 \]
DO 117 J= P, I+ P=1
117 \[ Z1 = Z1 + G(I - J + P, J + P + 1) * S(J) \]
S(I) = S(I) - B(I) * S(I+1) - Z - Z1
DO 118 II=N+1, N=1
I=N-1
DO 119 J = M, I + M - 1
119 Z = Z + F (I - J + M, J - M + 1) * S (J)
Z = 0
DO 131 J = P, I + P - 1
131 Z = Z + G (I - J + P, J = P + 1) * S (J)
S (I) = S (I) - B (I) * S (I + 1) - Z * Z
GO TO 66

821 IF ((N = N / 2) .EQ. 0 .AND. N .EQ. N / 2) GO TO 822
CASE: M = [N/2+1], P DOES NOT EXIST
DO 381 I = 1, M - 1
381 Z = 0
DO 382 J = I + 1, N
382 Z = Z + F (I - J + M, J - M + 1) * S (J)
S (I) = S (I) - B (I) * S (I + 1) - Z
DO 383 I = 1, N - Z * M + 1
I = N - M + 1 - I
Z = 0
DO 384 J = I + 1, I + M - 1
384 Z = Z + F (I - J + M, J - M + 1) * S (J)
S (I) = S (I) - B (I) * S (I + 1) - Z
DO 386 I = 1, M - 1
I = M - I
Z = 0
DO 387 J = M, I + M - 1
387 Z = Z + F (I - J + M, J - M + 1) * S (J)
S (I) = S (I) - B (I) * S (I + 1) - Z
GO TO 66
CASE: M = N / 2, P DOES NOT EXIST.
822 DO 388 I = 1, N - M - 1
388 Z = 0
DO 389 J = I + 1, N
389 Z = Z + F (I - J + M, J = M + 1) * S (J)
S (I) = S (I) - B (I) * S (I + 1) - Z
I = M
Z = 0
DO 391 J = I + 1, N - 1
391 Z = Z + F (I - J + M, J = M + 1) * S (J)
S (I) = S (I) - B (I) * S (I + 1) - Z
DO 392 I = 1, M - 1
I = M - I
Z = 0
DO 393 J = M, I + M - 1
393 Z = Z + F (I - J + M, J - M + 1) * S (J)
392 S (I) = S (I) - B (I) * S (I + 1) - Z
66 DO 76 I = 1, N
76 S (I) = S (I) / A (I)
RETURN
999 WHITE (2800)
800 FORMAT (1H15X,"*** GIVEN MATRIX IS A TRIDIAGONAL MATRIX***")
RETURN
END
SUBROUTINE ALUBOT(N, M, IR, B, A, C, U, V)
C THIS SUBROUTINE IS AN APPROXIMATE FACTORIZATION OF A SQUARE
C MATRIX OF ORDER N. THE COEFFICIENT MATRIX IS NON-SYMMETRIC,
C POSITIVE DEFINITE AND QUINDIAGONAL OF BANDWIDTH M (3< M < N).
C B, C, A, U, V ARE VECTORS CONTAINING RESPECTIVELY THE DIAGONAL,
C UPPER AND LOWER TRI DIAGONAL AND M-TH TRIAGONAL ELEMENTS.
C THE COEFFICIENT MATRIX IS FACTORIZED INTO LS US WHERE LS
C US (WITH UNIT DIAGONAL ELEMENTS) ARE STRICTLY LOWER AND UPPER
C TRIANGULAR MATRICES WITH NON-ZERO ELEMENTS IN SUB, SUPER
C DIAGONAL RESPECTIVELY AND RETAINING IR OF OUTERMOST
C OFF-DIAGONAL ENTRIES. THREE RESULT VECTORS OF LENGTH N AND
C TWO ARRAYS OF DIMENSION [IR:N:M+1] ARE USED AS WORKSPACE.
C N=1=M+1 ,
DIMENSION B(N), A(N), C(N), U(N), V(N)
COMMON /BLOCK2/G(N), R(N), D(N)
COMMON /BLOCK4/IR(N1), E(IR,N1)
C FACTORIZATION OF TRIDIAGONAL MATRIX A11
W(1)=B(1)
D(1)=A(2)
G(1)=C(1)/W(1)
DO 5 IM2=2,M-2
W(IM)=W(IM-2)*D(IM-2)*G(IM-2)
5 D(IM)=A(IM+1)
C CALCULATION OF ELEMENTS OF SUBMATRICES U12, L21, U22, L22.
DO 6 JM1=N+1
E(I,J)=W(JM)+W(J)
H(I,J)=W(JM)*H(JM-1)
G(JM-2)=C(JM-1)/W(JM-2)
D(JM-2)=A(JM-1)
IF (IR=J+1, LT, 2) GO TO 22
DO 7 IM2=IR,J+1
E(I,J)=W(JM)*E(JM-1)*E(JM-2)
7 H(I,J)=W(JM)*H(JM-1)*H(JM-2)
22 IF (J, G, IR ) IP=2
IF (J, LE, IR) IP=IR-J+2
IR1=IR+1
DO 8 IMP=IR
Z=0
DO 9 KM1=1,M-1
Z=Z+E(KM1)*H(KM1)+E(KM1)*H(KM1)+E(KM1)*H(KM1)
8 CONTINUE
C 23 I=IR
Z=0
DO 13 KM1=1,M-1
Z=Z+E(KM1)*H(KM1)
13 CONTINUE
SUBROUTINE FBSUBS(N,M,IR,S)

THIS SUBROUTINE SOLVES THE SET (LS US)Y=S OF N-LINEAR
EQUATIONS. THE SOLUTION IS EFFECTED BY A FORWARD-BACKWARD
SUBSTITUTION PROCESS WHERE THE INPUT VECTOR S IS OVERWRITTEN
SUCCESSIVELY BY THE INTERMEDIATE SOLUTION (OBTAINED BY FORWARD)
SUBSTITUTION) AND THE FINAL SOLUTION (OBTAINED BY BACK
SUBSTITUTION). N=MAX(N-M+1, 1).

DIMENSION S(N), C
COMMON /BLOCK1/G(N), W(N), B(N)
COMMON /BLOCK2/T(1R, N), E(1R, N)
S(1)=S(1)/W(1)
DO 7 I=2, M+1
7 S(I)=S(I)= B(I-1) * S(I-1) / W(I)
DO 8 I=M+1, N
Z=0
DO 9 K=1, M+1, I-I+1
9 Z=Z+E(K-I+1, I-I+1) * S(K)
8 S(I)=S(I)= S(I-1) * S(I-1) / S(I)
IF (M .LT. N/2+1) GO TO 34
IF (M .EQ. N) GO TO 35
CASE 11 > [N/2+1]

DO 10 II=1, N-M+1
I=N-II
Z=0
DO 11 J=I+1, N
IF (I-J+1, GT. IR) GO TO 11
Z=Z+T(I-J+1, J-I+1) * S(J)
10 CONTINUE
11 S(I)= S(I)= G(I) * S(I-1) * Z
DO 12 II=1, 2*M-N+1
I=N-II
Z=0
DO 13 J=I, N
IF (I-J+1, GT. IR) GO TO 13
Z=Z+T(I-J+1, J-I+1) * S(J)
12 CONTINUE
13 S(I)= S(I)= G(I) * S(I-1) * Z
IF (M .EQ. N) GO TO 36
DO 14 II=1, N-M+1
I=N-M+1-I
Z=0
DO 15 J=M+1, I+H-1
IF (I-J+1, GT. IR) GO TO 15
Z=Z+T(I-J+1, J-I+1) * S(J)
14 CONTINUE
15 S(I)= S(I)= G(I) * S(I-1) * Z
GO TO 36
34 IF (((N-N/2+2), EQ, 0).AND. (M .EQ. N/2)) GO TO 37
CASE 11 < [N/2+1]

DO 16 II=1, M+1
I=N-II
Z=0
DO 17 J=I+1, N
IF (I-J+1, GT. IR) GO TO 17
GO TO 36
Z = Z + T(I = J + M, J = M + 1) * S(J)

CONTINUE

S(I) = S(I) + G(I) * S(I + 1) - Z
DO 18 II = I, N - 2 * M + 1
I = N - M + 1 - II
Z = 0
DO 19 J = I + 1, I + H - 1
IF (I - J + H, GT, IR) GO TO 19
Z = Z + T(I - J + II, J = M + 1) * S(J)
CONTINUE

S(I) = S(I) + G(I) * S(I + 1) - Z
DO 20 II = I, M + 1
I = H - II
Z = 0
DO 21 J = I + M + 1, M - 1
IF (I - J + M, GT, IR) GO TO 21
Z = Z + T(I - J + II, J = M + 1) * S(J)
CONTINUE

S(I) = S(I) + G(I) * S(I + 1) - Z
GO TO 36

CASE: N IS EVEN AND M = N/2.

DO 22 II = I, N - M - 1
I = N - II
Z = 0
DO 23 J = I + 1, N
Z = Z + T(I + J + II, J = M + 1) * S(J)
22 S(I) = S(I) + G(I) * S(I + 1) - Z
I = H
Z = 0
DO 24 J = M + 1, N - 1
Z = Z + T(I - J + II, J = M + 1) * S(J)
S(I) = S(I) + G(I) * S(I + 1) - Z
DO 25 II = I, M - 1
I = H - II
Z = 0
DO 26 J = H, I + M - 1
Z = Z + T(I + J + II, J = M + 1) * S(J)
S(I) = S(I) + G(I) * S(I + 1) - Z
25 CONTINUE
RETURN
END
SUBROUTINE ALUBOT3D(N,P;M;A;B;C;U;V;IR;IS)
C THIS SUBROUTINE IS AN APPROXIMATE FACTORIZATION OF A SQUARE
C MATRIX OF ORDER N. THE COEFFICIENT MATRIX IS NON-SYMMETRIC,
C POSITIVE DEFINITE, SEVEN DIAGONAL OF BANDWIDTHS M AND P
C THE DIAGONAL, UPPER AND LOWER OFF DIAGONAL, M-TH DIAGONAL AND
C P-TH DIAGONAL ELEMENTS. THE COEFFICIENT MATRIX IS
C APPROXIMATELY FACTORIZED INTO LS US WHERE LS,US (WITH UNIT
C DIAGONAL ELEMENTS) ARE STRICTLY LOWER AND UPPER TRIANGULAR
C MATRICES WITH NONZERO ELEMENTS IN SUB/SUPER DIAGONAL
C RESPECTIVELY, RETAINING IR1 [1,H-1], IR2 [1,P-1] OUTERMOST
C OFF-DIAGONAL ENTRIES. THREE RESULT VECTORS OF LENGTH N AND
C FOUR ARRAYS OF DIMENSION [IR1,N,H-1] AND [IR2,N,P-1] ARE USED
C AS WORKSPACE. N1=N-M+1, N2=N-P+1.
COMMON /BLOCK2/G(N),D(N),W(N)
DIMENSION A(N),B(N),C(N),U(N),V(N),R(N),S(N)
INTEGER P
COMMON /BLOCK7/IRI,IR2
COMMON H(IR1,N1),T(IR2,N2),E(IR1,N1),F(IR2,N2)
IR1=IR1+1
IR2=IR2+1
W(1)=B(1)
D(1)=A(2)
G(1)=C(1)/W(1)
DO 5 IR2=IR1
W(M-I)=D(I-I)*G(I-1)
D(I)=A(I+1)
5 G(I)=C(I)/W(I)
W(M-I+1)=D(M-I)*G(M-I)
DO 6 J=I+1,N
E(I,J)=W(J)/W(I)
H(I,J)=MU(J)/W(J)
G(I+J-2)=C(I+J-2)/W(I+J-2)
D(I+J-2)=A(H+J-1)
IF (IR1, EQ, 1) GO TO 18
6 IF (J, GT, IR1-1) GO TO 4
DO 7 IR2=IR1
E(I,J)=G(I+J-2)*E(I-1,J)
H(I,J)=D(I-J-1)*H(I-J-1,J)/W(I+J-1)
7 IF (J, EQ, 1, OR, IR1, EQ, 0) GO TO 18
4 IF (J, EQ, 1, OR, IR1, EQ, 1) GO TO 18
IF (J, LE, IR1) IP=2
DO 8 IR1=1,IP
3 Z=O
DO 9 K=1,IP
Z=Z+E(K,J)*H(K-I+IR1S,I+J-IR1S)
E(I,J)=G(I+J-2)*E(I-1,J)-Z
9 Z=O
DO 10 K=1,IP
Z=Z+E(K-J)*H(I+K-1,I-1,J-1)-Z)/W(I+J-1)
8 CONTINUE
18 IR1=IR1
Z=O
DO 19 K=1,IR1
IF (K, GT, IR1) GO TO 13
13 CONTINUE
6 CONTINUE

C GENERATION OF U12
DO 22 J = IPN + 1, N + 1
T(I,J) = R(J) / W(J)
G(P+J-2) = C*(P+J-2) / W(P+J-2)
F(1,J) = S(P+J-1)
D(P+J-2) = A(P+J-1)
IF (J, EQ, 1) GO TO 33
IF (J, GT, IR2-1) GO TO 17
DO 23 I = 2, IR2S-J
T(I,J) = D(I,J) / W(I+J-1)
F(I,J) = G(I,J) * F(I,J)
D(P+J-1) = A(P+J-1)
IF (I, NE, 1) GO TO 53
GO TO 93
55 T(I,J) = T(I,J) + U(I,J-1) / W(I+J-1)
F(I,J) = F(I,J) + V(P,J-1)
23 CONTINUE
17 IF (J, EQ, 1, OR, IR2, EQ, 1) GO TO 33
IF (J, GT, IR2) IP = 2
IF (J, LE, IR2) IP = IR2 - J + 2
DO 24 I = IP, IR2
IF (I, EQ, 1, OR, J, EQ, 1, AND, J, LE, P - 2) GO TO 66
Z = 0
Z1 = 0
IF (I, LT, M) GO TO 25
DO 28 K = 1, M - 1
IF (K, GT, IR1) GO TO 28
Z = Z + E(K, I + J - M) * T(K, I + J - M)
Z1 = Z1 + F(K, I + J - M) * H(K, I + J - M)
28 CONTINUE
GO TO 27
25 DO 26 K = 1, I - 1
IF (K - I + IR1S, LT, 1, OR, I + J - IR1S, LT; 1) GO TO 26
Z = Z + E(K - I + IR1S, I + J - IR1S) * T(K, J)
Z1 = Z1 + F(K, J) * H(K - I + IR1S, I + J - IR1S)
26 CONTINUE
27 T(I,J) = D(I,J) * T(I - 1, J) / W(I + J - 1)
F(I,J) = G(I,J) * F(I,J)
IF (I, EQ, P - M + 1, AND, J, NE, 1) GO TO 53
GO TO 24
53 T(I,J) = T(I,J) + U(I,J-1) / W(I+J-1)
F(I,J) = F(I,J) + V(P,J-1)
GO TO 24
C GENERATION OF U12, U22
65 IF (J, EQ, 1) GO TO 33
66 Z = 0
Z1 = 0
DO 29 K = 1, I - 1
IF (K - I + IR2S, LT, 1) GO TO 29
Z = Z + F(K - I + IR2S, I + J - IR2S) * T(K, J)
Z1 = Z1 + F(K, J) * T(K - I + IR2S, I + J - IR2S)
29 CONTINUE
T(I,J) = D(I,J) * T(I - 1, J) / W(I + J - 1)
F(I,J) = G(I,J) * F(I,J)
IF (I, NE, P = M + 1) GO TO 24
T(I,J) = T(I,J) + U(I+J-1) / W(I+J-1)
SUBROUTINE FBSUBS3D(N,P,M,S)
C THIS SUBROUTINE SOLVES THE SET (LS US)Y=S OF N-LINEAR
C EQUATIONS, THE SOLUTION IS EFFECTED BY A FORWARD-BACKWARD
C SUBSTITUTION PROCESS WHERE THE INPUT VECTOR S IS OVERWRITTEN
C SUCCESSIVELY BY THE INTERMEDIATE SOLUTION (OBTAINED BY FORWARD
C SUBSTITUTION) AND THE FINAL SOLUTION (OBTAINED BY BACK
C SUBSTITUTION), N1=N-M+1, N2=N-P+1.
DIMENSION S(N)
COMMON /BLOCK2/GS(N),B(N),W(N)
COMMON H(IR1,N1),T(IR2,N2),E(IR1,N1),F(IR2,N2)
COMMON /BLOCK7/IR1,IR2
INTEGER P
S(1)=S(1)/W(1)
DO 44 I=2,N-1
44 S(I)=(S(I)-B(I-1)*S(I-1))/W(I)
DO 78 I=P,N-1
Z=0
DO 79 K=I-M+1,I-M+1
79 Z=Z+E(K-I+1,K)*S(K)
78 S(I)=(S(I)-H(I-N)*S(I-N)+Z)/W(I)
DO 87 I=P,N
Z=0
DO 88 K=I-M+1,I-M+1
88 Z=Z+E(K-I+1,K)*S(K)
87 S(I)=(S(I)-B(I-1)*S(I-1)+Z)/W(I)
IF (H,LT,(N/2+1)) GO TO 64
IF (H,EQ.N) GO TO 65
IF (P,GT.(N/2+1)) GO TO 183
C-- CASE M>(N/2+1), P DOES NOT EXIST
DO 47 II=1,N-M
47 S(I)=S(I)
I=N-M
RETURN
END
Z = 0
DO 48 J = I + 1, N
IF (I - J + M, GT, IR1) GO TO 48
Z = Z + H(I - J + M, J - M + 1) * S(J)
48 CONTINUE

47 S(I) = S(I) - GS(I) * S(I + 1) - Z
DO 49 I = 1, 2 * M - N + 1
I = M
Z = 0
DO 51 J = M, N
IF (I - J + M, GT, IR1) GO TO 51
Z = Z + H(I - J + M, J - M + 1) * S(J)
51 CONTINUE

49 S(I) = S(I) - GS(I) * S(I + 1) + Z
IF (H, EQ, N) GO TO 66
DO 52 I = 1, N - M
I = N - M + 1 - I
Z = 0
DO 53 J = M, I + M - 1
IF (I - J + M, GT, IR1) GO TO 53
Z = Z + H(I - J + M, J - M + 1) * S(J)
53 CONTINUE

52 S(I) = S(I) - GS(I) * S(I + 1) + Z
GO TO 66

CASE : N > (N/2 + 1), P > (N/2 + 1)
183 IF (P, EQ, N) GO TO 252
DO 231 I = 1, N - P
I = N - I
Z = 0
Z1 = 0
DO 232 J = I + 1, N
IF (I - J + M, GT, IR1) GO TO 932
Z = Z + H(I - J + M, J - M + 1) * S(J)
932 IF (I - J + P, GT, IR2) GO TO 232
Z1 = Z1 + T(I - J + P, J - P + 1) * S(J)
232 CONTINUE

231 S(I) = S(I) - GS(I) * S(I + 1) + Z = Z1
DO 233 I = 1, P - M
I = P - I
Z = 0
DO 234 J = I + 1, N
IF (I - J + M, GT, IR1) GO TO 234
Z = Z + H(I - J + M, J - M + 1) * S(J)
234 CONTINUE
Z1 = 0
DO 235 J = M, N
IF (I - J + P, GT, IR2) GO TO 235
Z1 = Z1 + T(I - J + P, J - P + 1) * S(J)
235 CONTINUE

233 S(I) = S(I) - GS(I) * S(I + 1) + Z = Z1
DO 236 I = N - M + 1, M + 1
I = N - I
Z = 0
Z1 = 0
DO 237 J = M, N
IF (I - J + M, GT, IR1) GO TO 937
Z = Z + H(I - J + M, J - M + 1) * S(J)
937 IF (I - J + P, GT, IR2) GO TO 237
Z1 = Z1 + T(I - J + P, J - P + 1) * S(J)
CONTINUE
S(I)=S(I)-GS(I)*S(I+1)-Z-Z1
IF (M.EQ.N) GO TO 66
DO 238 II=M, P-1
I=N-II
Z=0
DO 239 J=M, I+H-1
IF (I-J+M, GT, IR1) GO TO 239
Z=Z+H(I-J+M, J-M+1)*S(J)
CONTINUE
Z1=0
DO 241 J=M, N
IF (I-J+P, GT, IR2) GO TO 241
Z1=Z1+T(I-J+P, J-P+1)*S(J)
CONTINUE
S(I)=S(I)-GS(I)*S(I+1)-Z-Z1
IF (P.EQ.N) GO TO 66
DO 242 II=P, N-1
I=N-II
Z=0
Z1=0
DO 243 J=M, I+H-1
IF (I-J+M, GT, IR1) GO TO 243
Z=Z+H(I-J+M, J-M+1)*S(J)
CONTINUE
S(I)=S(I)-GS(I)*S(I+1)-Z-Z1
GO TO 66
64 IF (P.GT.N.OR.P.LE.M) GO TO 821
IF (P.LT.(N/2+1)) GO TO 444
IF (((N-N/2*2).EQ.0).AND.((M.EQ.N/2))) GO TO 67
C--- CASE: \( M < \frac{N}{2} + 1 \), \( P > \frac{N}{2} + 1 \).
IF (P.EQ.N) GO TO 254
DO 94 II=1, N- P
I=N-II
Z=0
Z1=0
DO 95 J=II+1, N
IF (I-J+M, GT, IR1) GO TO 95
Z1=Z1+T(I-J+P, J-P+1)*S(J)
CONTINUE
S(I)=S(I)-GS(I)*S(I+1)-Z-Z1
DO 96 II=N, P+1, M-1
I=N-II
Z=0
DO 97 J=II+1, N
IF (I-J+M, GT, IR1) GO TO 97
Z=Z+H(I-J+M, J-M+1)*S(J)
CONTINUE
Z1=0
DO 98 J=P, N
IF (I-J+P, GT, IR2) GO TO 98
Z1=Z1+T(I-J+P, J-P+1)*S(J)
CONTINUE
S(I)=S(I)-GS(I)*S(I+1)-Z-Z1
DO 101 II=N, N-M
I=N-II
```
Z=0
DO 102 J=I+1, I+M-1
IF (I-J+M, GT, IR1) GO TO 102
Z=Z+H(I-J+M, J-M+1)*S(J)
102 CONTINUE
Z1=0
DO 103 J= P, N
IF (I-J+P, GT, IR2) GO TO 103
Z1=Z1+T(I-J+ P, J-M+1)*S(J)
103 CONTINUE
S(I)=S(I)+G5(I)*S(I+1)=Z-Z1
DO 104 II=N-M+1, P=1
I=N-I1
Z=0
DO 105 J=M, I+M-1
IF (I-J+M, GT, IR1) GO TO 105
Z=Z+H(I-J+M, J-M+1)*S(J)
105 CONTINUE
S(I)=S(I)+G5(I)*S(I+1)=Z-Z1
DO 106 J=P, N
IF (I-J+P, GT, IR2) GO TO 106
Z1=Z1+T(I-J+ P, J+1)*S(J)
106 CONTINUE
S(I)+G5(I)*S(I+1)=Z-Z1
IF (P, EQ; N) GO TO 66
DO 121 II=P, N-1
I=N-I1
Z=0
DO 122 J=M, I+M-1
IF (I-J+M, GT, IR1) GO TO 122
Z=Z+H(I-J+M, J-M+1)*S(J)
122 CONTINUE
S(I)+G5(I)*S(I+1)=Z-Z1
GO TO 66
CASE 1: H=N/2 , P=[N/2]+1.
67 IF (P, EQ; N) GO TO 256
DO 61 II=1, N-P
I=N-I1
Z=0
DO 62 J=I+1, N
IF (I-J+M, GT, IR1) GO TO 281
Z=Z+H(I-J+M, J-M+1)*S(J)
281 IF (I-J+P, GT, IR2) GO TO 62
Z1=Z1+T(I-J+ P, J+1)*S(J)
62 CONTINUE
S(I)+G5(I)*S(I+1)=Z-Z1
DO 141 II=1, P=M-1
I=R-I1
Z=0
DO 142 J=I+1, N
IF (I-J+M, GT, IR1) GO TO 142
Z=Z+H(I-J+H, J-M+1)*S(J)
142 CONTINUE
Z1=0
```
DO 145 J=JP1
IF (I=J+P,GT,IR2) GO TO 145
Z=Z1+T(I=J+P,J=P+1)*S(J)
145 CONTINUE
S(I)=S(I)-GS(I)*S(I+1)-Z-Z1
I=I+1
Z=0
DO 146 J=H+1,N-1
IF (I=J+H,GT,IR1) GO TO 146
Z=Z+H(I=J+H,J=H+1)*S(J)
146 CONTINUE
Z=0
DO 147 J=PN
IF (I=J+P,GT,IR2) GO TO 147
Z1=Z1+T(I=J+P,J=P+1)*S(J)
147 CONTINUE
S(I)=S(I)-GS(I)*S(I+1)-Z-Z1
DO 148 II=1,MP=N-1
I=I+1
Z=0
DO 149 J=NI+1
IF (I=J+I,GT,IR1) GO TO 149
Z=Z+H(I=J+I,J=I+1)*S(J)
149 CONTINUE
Z=0
DO 150 J=PN
IF (I=J+P,GT,IR2) GO TO 150
Z1=Z1+T(I=J+P,J=P+1)*S(J)
150 CONTINUE
S(I)=S(I)-GS(I)*S(I+1)-Z-Z1
DO 151 ll=1,I-1
I=NI
Z=0
DO 152 J=NP
IF (I=J+N,GT,IR1) GO TO 152
Z=Z+H(I=J+N,J=N+1)*S(J)
152 CONTINUE
Z=0
DO 153 J=IP+1
IF (I=J+P,GT,IR2) GO TO 153
Z1=Z1+T(I=J+P,J=P+1)*S(J)
153 CONTINUE
S(I)=S(I)-GS(I)*S(I+1)-Z-Z1
GO TO 66
C; -m CASE: M<[(N/2)+1], P<[(N/2)+1],
444 IF (((N-N/2+2),EQ,0).AND.(( P,EQ,N/2)) GO TO 107
DO 54 II=N+1,M-1
I=I+1
Z=0
Z=0
DO 55 J=I+1,N
IF (I=J+P,GT,IR2) GO TO 348
Z1=Z1+T(I=J+ P,J= P+1)*S(J)
348 IF (I=J+H,GT,IR1) GO TO 55
Z=Z+H(I=J+H,J=H+1)*S(J)
55 CONTINUE
S(I)=S(I)-GS(I)*S(I+1)-Z-Z1
DO 56 II=M, P-1
I=I-1

Z = 0
DO 57 J = 1 + 1, I + M - 1
IF (I - J + M, GT, IR1) GO TO 57
Z = Z + H(I - J + M, J - M + 1) * S(J)
57 CONTINUE
Z = 0
DO 91 J = 1 + 1, N
IF (I - J + P, GT, IR2) GO TO 91
Z = Z + T(I - J + P, J - P + 1) * S(J)
91 CONTINUE
S(I) = S(I) * GS(I) * S(I + 1) - Z - Z
DO 124 I = 1 + 1, P - 1
I = N - I
Z = 0
DO 126 J = 1 + 1, I + M - 1
IF (I - J + M, GT, IR1) GO TO 126
Z = Z + H(I - J + M, J - M + 1) * S(J)
124 CONTINUE

Z = 0
DO 126 J = 1 + 1, I + P - 1
IF (I - J + P, GT, IR2) GO TO 126
Z = Z + T(I - J + P, J - P + 1) * S(J)
126 CONTINUE
S(I) = S(I) * GS(I) * S(I + 1) - Z - Z
DO 127 I = 1 + 1, P + 1, N - M
I = N - I
Z = 0
DO 128 J = 1 + 1, I + M - 1
IF (I - J + M, GT, IR1) GO TO 128
Z = Z + H(I - J + M, J - M + 1) * S(J)
128 CONTINUE
Z = 0
DO 129 J = 1 + 1, I + P - 1
IF (I - J + P, GT, IR2) GO TO 129
Z = Z + T(I - J + P, J - P + 1) * S(J)
129 CONTINUE
S(I) = S(I) * GS(I) * S(I + 1) - Z - Z
DO 58 I = 1 + 1, M - 1
I = H - I
Z = 0
DO 59 J = 1 + 1, I + M - 1
IF (I - J + M, GT, IR1) GO TO 59
Z = Z + H(I - J + M, J - M + 1) * S(J)
59 CONTINUE
Z = 0
DO 93 J = 1 + 1, I + P - 1
IF (I - J + P, GT, IR2) GO TO 93
Z = Z + T(I - J + P, J - P + 1) * S(J)
93 CONTINUE
S(I) = S(I) * GS(I) * S(I + 1) - Z - Z
GO TO 66
C ---- CASE : N < [N/2 + 1], P = N/2
107 DO 108 I = 1 + 1, P - 1
I = N - I
Z = 0
DO 109 J = 1 + 1, N
IF (I - J + P, GT, IR2) GO TO 409
Z = Z + T(I - J + P, J - P + 1) * S(J)
108 CONTINUE
S(I) = S(I) * GS(I) * S(I + 1) - Z - Z
GO TO 66

Cr = ---
Z = Z + H(I-J+M, J=M+1) * S(J)
109 CONTINUE
108 S(I) = S(I) + S(I+1) = Z = Z1
I = P
Z = 0
DO 111 J = I+1, N
IF (I-J+M, GT, IR1) GO TO 111
Z = Z + H(I-J+M, J=M+1) * S(J)
111 CONTINUE
Z1 = 0
DO 112 J = P+1, N-1
IF (I-J+P, GT, IR2) GO TO 112
Z1 = Z1 + T(I-J+P, J=M+1) * S(J)
112 CONTINUE
S(I) = S(I) + S(I+1) = Z = Z1
DO 113 I = N-M, P+1, N-1
I = N-1
Z = 0
DO 114 J = I+1, N
IF (I-J+M, GT, IR1) GO TO 114
Z1 = Z1 + T(I-J+P, J=M+1) * S(J)
114 CONTINUE
S(I) = S(I) + S(I+1) = Z = Z1
I = M
Z = 0
DO 116 J = M+1, N-1
IF (I-J+M, GT, IR1) GO TO 116
Z1 = Z1 + T(I-J+P, J=M+1) * S(J)
116 CONTINUE
Z1 = 0
DO 117 J = P, I+ P-1
IF (I-J+P, GT, IR2) GO TO 117
Z1 = Z1 + T(I-J+P, J=M+1) * S(J)
117 CONTINUE
S(I) = S(I) + S(I+1) = Z = Z1
DO 118 I = N-M+1, N-1
I = N-1
Z = 0
DO 119 J = M+1, I+1
IF (I-J+M, GT, IR1) GO TO 119
Z1 = Z1 + T(I-J+P, J=M+1) * S(J)
119 CONTINUE
Z1 = 0
DO 131 J = P, I+ P-1
IF (I-J+P, GT, IR2) GO TO 131
Z1 = Z1 + T(I-J+P, J=M+1) * S(J)
131 CONTINUE
S(I) = S(I) + S(I+1) = Z = Z1
DO 821 (N = N/2*2).EQ.0, AND. M.EQ.N/2) GO TO 822
CASE: M < (N/2+1), P DOES NOT EXIST
DO 381 II = M, I+1, H-1
I = N-M
Z = 0
DO 382 J = I+1, N
IF (I-J+H,GT,IR1) GO TO 382
Z=Z+H(I-J+M,J=M+1)*S(J)

382 CONTINUE
383 S(I)=S(I)-GS(I)*S(I+1)-Z
DO 383 II=1,N-2*M+1
I=N-M+1-II
Z=0
DO 384 J=I+1,I+II-1
IF (I-J+H,GT,IR1) GO TO 384
Z=Z+H(I-J+M,J=M+1)*S(J)

384 CONTINUE
385 S(I)=S(I)-GS(I)*S(I+1)-Z
DO 385 II=1,N-1
I=M-II
Z=0
DO 387 J=I+1,I+H-1
IF (I-J+H,GT,IR1) GO TO 387
Z=Z+H(I-J+M,J=M+1)*S(J)

387 CONTINUE
386 S(I)=S(I)-GS(I)*S(I+1)-Z
GO TO 66

CASE : M=N/2 , P DOES NOT EXIST.
388 S(I)=S(I)-GS(I)*S(I+1)-Z
DO 388 II=1,N-1
I=N-II
Z=0
DO 389 J=I+1,N
IF (I-J+H,GT,IR1) GO TO 389
Z=Z+H(I-J+M,J=M+1)*S(J)

389 CONTINUE
389 S(I)=S(I)-GS(I)*S(I+1)-Z
DO 389 II=1,N-1
I=H+II
Z=0
DO 391 J=I+1,N-1
IF (I-J+H,GT,IR1) GO TO 391
Z=Z+H(I-J+M,J=M+1)*S(J)

391 CONTINUE
390 S(I)=S(I)-GS(I)*S(I+1)-Z
DO 390 II=1,H-1
I=H-II
Z=0
DO 393 J=H,I+M-1
IF (I-J+H,GT,IR1) GO TO 393
Z=Z+H(I-J+M,J=M+1)*S(J)

393 CONTINUE
392 S(I)=S(I)-GS(I)*S(I+1)-Z
66 CONTINUE
RETURN
END
THE NORMALISED IMPLICIT CONJUGATE GRADIENT METHOD
IN TWO SPACE DIMENSIONS.

AN ACCELERATED METHOD FOR SYMMETRIC SYSTEMS, DERIVED FROM LINEAR LAPLACE IN UNIT SQUARE, $W \times W$ OF $N$ LINEAR EQUATIONS WHERE THE COEFFICIENT MATRIX $W$ IS SYMMETRIC, POSITIVE DEFINITE OF SEMIBANDWIDTH $M (3 < M < N)$, THE APPROXIMATE FACTORIZATION ALGORITHM NOBAR (I.E., SUBROUTINES APNOFA, GEDRHS, FBSUBS) RETAINING $R$ OUTERMOST OFF-DIAGONAL ENTRIES, IS USED AND THE ITERATIVE PROCESSES IS ACCELERATED (BY THE CONJUGATE GRADIENT METHOD). THE FOLLOWING CALLING SUBROUTINES ARE USED:

- RESID: CALCULATES THE PRODUCT $W \times X$ IF $IX = 1$; OTHERWISE CALCULATES THE RESIDUAL $R = W \times X$.
- ERME: CALCULATES THE ERROR MEASURES.


$Y$ IS A VECTOR OF DIMENSION $N$ CONTAINING THE COMPUTED SOLUTION. $VS, SI$ ARE WORKING VECTORS EACH OF DIMENSION $N$. DIMENSION $A(N), B(N), C(N), S(N), Y(N), VS(N), RI(N), SI(N)$ COMMON/ BLOCK1/N, M, HS, ANUM
COMMON/ BLOCK4/X(N)

INTEGER Q
READ (1, 10) N, M, RMS, IPAR
10 FORMAT (515)

IF IPAR = 1, THE SOLUTION IS A VECTOR OF $N$-RANDOM NUMBERS IN [0, 1]; OTHERWISE THE SOLUTION IS $X(1) = 0, X(1 + 1) = R, X(M + 1) = 1$.

IF (IPAR = 1) GO TO 12
14 WRITE (2, 1719) N, M, R, HS, IPAR
159 FORMAT(H1, 5X, 'ORDER OF MATRIX N = ', 15//6X, 'SEMIBANDWIDTH M = ', 15//6X, 'OUTERMOST OFF DIAGONAL ENTRIES R = ', 15//6X, 'EPS = ', 1.0E6)
550 WRITE (2, 270) EPS
270 FORMAT (H1, 5X, 'ACCURACY EPS = ', 1.20, 11//)

K = 0
Q = 0
DO 17 I = 1, N
239

17 \( Y(I) = GV \)

C \text{Y=HOLDS-- THE INITIAL APPROXIMATE SOLUTION Y0 .}
CALL GEVECT(N, M, HS, IX, A, B, C, S)
CALL RESID(Y, IX, S)

C \text{S=HOLDS--RO}
DO 24 I=1, N

24 \( RI(I) = S(I) \)
C \text{RI=HOLDS--R0}
CALL APNODA(N, M, R, A, B, C)
CALL GEDRHS(N, A, S)
CALL FBSUBS(N, M, R, B, S)

C \text{S=HOLDS--R0}
DO 28 I=1, N

28 \( SI(I) = S(I) \)
C \text{CALCULATION OF AI, Y,RI .}
IX=1

C
C \text{THE INNER LOOP--*--.}
C
C \text{CALCULATION OF PRODUCT W*SI .}
25 CALL RESID(SI, IX, VS)
C \text{SI=HOLDS--SO, VS=HOLDS--W*SO .}
Q=0
C \text{CALCULATION OF AI}
ANUM=0
DENOM=0
DO 41 I=1, N
ANUM = ANUM + RI(I) * S(I)
C \text{RI=HOLDS--R0, BI=HOLDS--R0}
41 DENOM = DENOM + SI(I) * VS(I)
C \text{VS=HOLDS--W*SO} 
IF (DENOM .LE. 0) GO TO 55
AI = ANUM / DENOM
WRITE (2, '210) AI
210 FORMAT (1H, 5X, 'SCALAR AI= ', E20, 11/

C \text{CALCULATION OF Y'RI .}
DO 19 I=1, N
Y(I) = Y(I) + AI * SI(I)

19 \( RI(I) = RI(I) + AI * VS(I) \)
C \text{RI=HOLDS--RI (I+1)}
C \text{TEST OF THE CONVERGENCE WITH THE RECURSIVE RESIDUAL:}
DO 23 I=1, N
IF (ABS(RI(I)) .LT. EPS) Q=Q+1
23 CONTINUE
CALL ERHES(RI, Y, S1, S2, S3, S4)
WRITE (Z, 290) S1, S2, S3, S4
290 FORMAT (1H, 5X, 'LOG10 OF SQRT (RI*(TS*TS)--(-1)*RI) = ',
* E20, 11/5X, 'LOG10 OF EUCL. NORM OF RESIDUALS RI(I)= ',
* E20, 11/5X, 'LOG10 OF R=(XK-X) = ', E20, 11/5X,
* 'LOG10 OF EUCL. NORM OF (XK-X) = ', E20, 11/

K=K+1
IF (Q .EQ. N) GO TO 35
IF (K .GE. KM) GO TO 45
DO 21 I=1, N
21 VS(I) = RI(I)
C \text{VS=HOLDS--RI (I+1)}
CALL FBSUBS(N, M, R, B, RI)
C \text{RI=HOLDS--RI (I+1)}
C \text{CALCULATION OF BI}
SUBROUTINE ANOFA(N,M,IR,A,B,C)
C THIS SUBROUTINE IS AN APPROXIMATE NORMALIZED FACTORIZATION
C OF A SQUARE MATRIX OF ORDER N. THE COEFFICIENT MATRIX IS
C SYMMETRIC, POSITIVE DEFINITE, QUINTIDIAGONAL OF SEMIBANDWIDTH
C M (3<M<N), A,B,C ARE VECTORS CONTAINING RESPECTIVELY THE
C DIAGONAL, CO DIAGONAL AND M-TH DIAGONAL ELEMENTS. THE MATRIX
C IS FACTORIZED INTO DS TS+ TS DS, WHERE DS IS A DIAGONAL
C MATRIX AND TS IS A REAL UPPER TRIANGULAR MATRIX WITH UNIT
C DIAGONAL ELEMENTS, NON-ZERO ELEMENTS IN SUPER DIAGONAL,
C RETAINING IR E[1'M-1] OUTERMOST OFF-DIAGONAL ENTRIES.
C TS+ DENOTES THE TRANSPOSE OF TS. ONE RESULT VECTOR OF LENGTH
C (M-1) AND ONE ARRAY OF DIMENSION [IR,N=M+1] ARE USED AS

ANUI11 =0
DO 22 I=1,N
22 ANUI11 =ANUI11 +VS(I)*RI(I)
BI =ANUI11 /ANUH
WRITE (2,220) BI

220 FORMAT (1H5,5X,'SCALEAR BI = ',E20.11/10X,'ERROR MEASURES'/)
C -- CALCULATION OF SI1 --
DO 42 I=1,N
42 SI(I)=R(I)+BI*SI(I)
C SI(I)--HOLDS--S(I+1), RI--HOLDS--R*(I+1)
DO 31 I=1,N
31 RI(I)=VS(I)
C SI(I)--HOLDS--R*(I+1), RI(I)--HOLDS--R(I+1)
GO TO 35

C -- TERMINATION OF INNER LOOP --

35 WRITE (2,190) K
190 FORMAT(1H5,5X,'NUMBER OF ITERATIONS K=',K15//6X7
* 'FINAL SOLUTION Y(I) '/)
DO 46 L=1,K
J=I-L+1
46 WRITE(2,170) (Y((J-1)=L+1),I=1,K)
170 FORMAT(1H6,5X,'IF (R.EQ.1,OR.R.EQ.4) GO TO 944
GO TO 945

944 WRITE (2,260)
260 FORMAT (1H5,5X,'RECURSIVE RESIDUAL RI(I)'/)
DO 48 L=1,K
J=I-L+1
48 WRITE (2,170) (RI((J-1)=L+1),I=1,K)
945 EPS =EPS/100.0
IF (EPS.LT.1.0E-10) GO TO 65
GO TO 550

65 R =R+1
IF (R.LE.4) GO TO 666
STOP 11

45 WRITE (2,260) KM
60 FORMAT (1H5,5X,'NO CONVERGENCE AFTER ',K5,' ITERATIONS')
STOP 111
55 WRITE (2,230)
230 FORMAT (1H5,5X,'THE COEFF. MATRIX IS NOT POSITIVE DEFINITE')
STOP 1111
END

220 FORMAT (1H5,5X,'SCALEAR BI = ',E20.11/10X,'ERROR MEASURES')
DO 42 I=1,N
42 SI(I)=R(I)+BI*SI(I)
DO 31 I=1,N
31 RI(I)=VS(I)
GO TO 35
WRITE (2,190) K
190 FORMAT(1H5,5X,'NUMBER OF ITERATIONS K=',K15//6X7
* 'FINAL SOLUTION Y(I) '/)
DO 46 L=1,K
J=I-L+1
46 WRITE (2,170) (Y((J-1)=L+1),I=1,K)
170 FORMAT(1H6,5X,'IF (R.EQ.1,OR.R.EQ.4) GO TO 944
GO TO 945
944 WRITE (2,260)
260 FORMAT (1H5,5X,'RECURSIVE RESIDUAL RI(I)'/)
DO 48 L=1,K
J=I-L+1
48 WRITE (2,170) (RI((J-1)=L+1),I=1,K)
945 EPS =EPS/100.0
IF (EPS.LT.1.0E-10) GO TO 65
GO TO 550
65 R =R+1
IF (R.LE.4) GO TO 666
STOP 11
45 WRITE (2,260) KM
60 FORMAT (1H5,5X,'NO CONVERGENCE AFTER ',K5,' ITERATIONS')
STOP 111
55 WRITE (2,230)
230 FORMAT (1H5,5X,'THE COEFF. MATRIX IS NOT POSITIVE DEFINITE')
STOP 1111
END
SUBROUTINE GEDRHS(N, A, S)
C THIS SUBROUTINE FORMS THE VECTOR Z WHERE D*Z=S.
C D IS A DIAGONAL MATRIX OF ORDER N, WHOSE NON-ZERO DIAGONAL
C ENTRIES ARE STORED IN A VECTOR A OF N ELEMENTS. THE INPUT
C VECTOR S IS OVERWRITTEN BY THE RESULT.
DIMENSION A(N), S(N)
DO 6 I = 1, N
6 S(I) = S(I) / A(I)
RETURN
END

SUBROUTINE FBSUBS(N, N, IR, B, S)
C THIS SUBROUTINE SOLVES THE SET (T*S) + Y = S OF N LINEAR
C EQUATIONS WHERE T*S IS AN UPPER TRIANGULAR MATRIX OF BANDWIDTH
C N WITH UNIT ELEMENTS ON THE DIAGONAL. T*S DENOTES THE
C TRANSPOSE OF T*S. THE NON-ZERO ELEMENTS ARE ON SUPER DIAGONAL
C AND IR ENTRIES OUTERMOST OFF-DIAGONAL ENTRIES ARE RETAINED.
C THE SOLUTION IS EFFECTED BY FORWARD AND BACKWARD SUBSTITUTION
C PROCESS WHERE THE INPUT VECTOR S IS OVERWRITTEN SUCCESSIVELY
C BY THE INTERMEDIATE SOLUTION OBTAINED BY FORWARD SUBSTITUTION
C
C AND THE FINAL SOLUTION (OBTAINED BY BACK SUBSTITUTION).
C
DIMENSION B(N), S(N)
COMMON T(IR, N)
DO 7 I=2, M-1
7 S(I) = S(I) * B(I-1) * S(I-1)
DO 8 I=M; N
Z=0
8 DO 9 K=I-M+1, I-M+IR
Z= Z + T(K-I+M, I-M+1) * S(K)
9 S(I) = S(I) * B(I-1) * S(I-1) - Z
IF (M.LT. (N/2+1)) GO TO 34
IF (M.EQ. N) GO TO 35
CASE : H > [N/2+1].
DO 10 II=1, N-M
I=N-II
Z=0
10 DO 11 J=I+1, N
IF (I-J+M.GT. IR) GO TO 11
Z= Z + T(I-J+M, J-M+1) * S(J)
11 CONTINUE
S(I) = S(I) * B(I) * S(I+1) - Z
DO 12 II=1, 2*M-N-1
I=M-II
Z=0
12 DO 13 J=I+1, N
IF (I-J+M.GT. IR) GO TO 13
Z= Z + T(I-J+M, J-M+1) * S(J)
13 CONTINUE
S(I) = S(I) * B(I) * S(I+1) - Z
IF (M.EQ. N) GO TO 36
DO 14 II=1, N-M
I=N-M+1-II
Z=0
14 DO 15 J=M+I, I+M-1
IF (I-J+M.GT. IR) GO TO 15
Z= Z + T(I-J+M, J-M+1) * S(J)
15 CONTINUE
S(I) = S(I) * B(I) * S(I+1) - Z
GO TO 36
CASE : H < [N/2+1].
DO 16 II=1, H-1
I=N-II
Z=0
16 DO 17 J=I+1, N
IF (I-J+M.GT. IR) GO TO 17
Z= Z + T(I-J+M, J-M+1) * S(J)
17 CONTINUE
S(I) = S(I) * B(I) * S(I+1) - Z
DO 18 II=1, N-2*M+1
I=N-M+1-II
Z=0
18 DO 19 J=I+1, I+M-1
IF (I-J+M.GT. IR) GO TO 19
Z= Z + T(I-J+M, J-M+1) * S(J)
19 CONTINUE
S(I) = S(I) * B(I) * S(I+1) - Z
DO 20 II=1, H-1
I=H-II
SUBROUTINE RESID (X, IND, S)
C THIS SUBROUTINE CALCULATES THE CORRESPONDING TO THE SET W*X A\(\text{\textsc{e}}\) RESIDUALS WHERE THE COEFFICIENT MATRIX IS OF ORDER N\(\text{\textsc{e}}\)
C SYMMETRIC, POSITIVE DEFINITE, QUINDIAGONAL OF SEMIBANDWIDTH M
C AND A, B, C, S ARE VECTORS CONTAINING RESPECTIVELY THE DIAGONAL,
C CO DIAGONAL, M-TH DIAGONAL AND CONSTANT ELEMENTS. THE
C RESIDUAL IS OVERWRITTEN BY VECTOR S. IF INDEX IND=1 THEN THE
C SUBROUTINE CALCULATES THE PRODUCT S OF THE GIVEN SPARSE
C MATRIX W BY THE INPUT VECTOR X.
C DIMENSION A(N), B(N), C(N), S(N), X(N)
COMMON /BLOCK1/N, M, IND, A, B, C, S
CALL GEVCT(N, M, IND, A, B, C, S)
S(1) = S(1) + ((A(1)*X(1) + B(1)*X(2) + C(1)*X(M+1-1))
IF (M LT (N/2+1)) GO TO 15
C CASE : N IS EVEN AND M=N/2.
DO 2 1=2, N-1
S(1) = S(1) - (B(1-1)*X(I-1)+A(I)*X(I)+B(I)*X(I+1)+C(I)*X(M+I-1))
DO 3 I=M+2, N-1
S(1) = S(1) + (B(I-1)*X(I-1)+A(I)*X(I)+B(I)*X(I+1))
DO 4 I=M+1
S(1) = S(1) - (C(I-I+1)*X(I-M+1)+B(I-1)*X(I-1)+A(I)*X(I)+
*B(I)*X(I+1))
CONTINUE
GO TO 35
15 IF (M .EQ. N/2) GO TO 25
CASE : H < N/2.
DO 5 I=M+2, M-1
  5 S(I) = S(I) - (B(I-1) * X(I-1) + A(I) * X(I) + B(I) * X(I+1) + C(I) * X(I+M-1))
      S(I) = S(I) - (C(I-M+1) * X(I-M+1) + B(I-1) * X(I-1) + A(I) * X(I) +
              * B(I) * X(I+1) + C(I) * X(I+M-1))
      CONTINUE
    DU 6 I=N-M+2, N-1
    S(I) = S(I) - (C(I-M+1) * X(I-M+1) + B(I-1) * X(I-1) + A(I) * X(I) +
                   * B(I) * X(I+1) + C(I) * X(I+M-1))
  7 CONTINUE
GO TO 35
CASE : H = N/2.
25 DO 8 I=M+2, M-1
  8 S(I) = S(I) - (B(I-1) * X(I-1) + A(I) * X(I) + B(I) * X(I+1) + C(I) * X(I+M-1))
      S(I) = S(I) - (C(I-N+M+1) * X(I-N+M+1) + B(I-1) * X(I-1) + A(I) * X(I) +
                   * B(I) * X(I+1) + C(I) * X(I+M-1))
      CONTINUE
    DU 9 I=N-M+2, N-1
    S(I) = S(I) - (C(I-N+M+1) * X(I-N+M+1) + B(I-1) * X(I-1) + A(I) * X(I) +
                   * B(I) * X(I+1))
  11 CONTINUE
35 S(N) = S(N) - (C(N-M+1) * X(N-M+1) + B(N-1) * X(N-1) + A(N) * X(N))
      IF (IND.EQ.1) GO TO 20
      RETURN
  20 DU 12 I=1, N
  12 S(I) = -S(I)
      RETURN
END
THE NORMALISED IMPLICIT CONJUGATE GRADIENT METHOD
IN THREE SPACE DIMENSIONS.

AN ACCELERATED METHOD FOR SYMMETRIC SYSTEMS, DERIVED FROM
LINEAR LAPLACE IN UNIT CUBE: \( W \times x = s \) OF \( n \) LINEAR EQUATIONS
WHERE THE COEFFICIENT MATRIX \( W \) IS SYMMETRIC, POSITIVE
DEFINITE OF SEMIBANDWIDTHS \( m \) AND \( p \) \((4 \leq m \leq p < n)\).

THE APPROXIMATE FACTORIZATION ALGORITHM NUBAR3D (i.e.,
SUBROUTINES: APNOFA3D, GEDRNS, FBSUBS3D) RETAINING \( m \) AND \( p \)
OUTERMOST OFF-DIAGONAL ENTRIES IS USED AND THE ITERATIVE
PROCESS IS ACCELERATED (BY THE CONJUGATE GRADIENT METHOD).

THE FOLLOWING CALLING SUBROUTINES ARE USED:

GVECT: GENERATES THE DIAGONAL VECTORS \( A, B, C \) OF THE
COEFFICIENT MATRIX AND THE R.H.S. VECTOR \( s \).

RESID3D: CALCULATES THE PRODUCT \( W \times x \) IF \( i x = \) 1, OTHERWISE
CALCULATES THE RESIDUAL \( r = W \times x \).

ERROR: CALCULATES THE ERROR MEASURES.

THE \( n, A, G \) SUBROUTINE G05AAF GENERATES THE \( n \) PSEUDO-RANDOM
NUMBERS BETWEEN \((0, 1)\).

\( Y \) IS A VECTOR OF DIMENSION \( n \) CONTAINING THE COMPUTED
SOLUTION, \( x \). \( s \) AND \( s \) ARE WORKING VECTORS EACH OF DIMENSION \( n \).

DIMENSION \( A(n), b(n), c(n), h(n), s(n), y(n), vs(n), ri(n), si(n) \)
COMMON / BLOCK1/ \( n, m, p, \) ANUM
COMMON / BLOCK4/ \( x(n) \)
COMMON / BLOCK5/ \( irf, irg \)
INTEGER \( q, p \)
READ (1, 10) \( n, p, irf, irg, ms, index \)
10 FORMAT (6I5)
   \( i h = h - 1 \)
   \( i h = i h \times i h \)
   IF (INDEX = 1, THE SOLUTION IS A VECTOR OF \( n \) RANDOM NUMBERS IN
   \( (0, 1) \), OTHERWISE THE SOLUTION IS \( x(1) = 0, i e, h + 1, x(h + 1) = 1 \).
   IF (INDEX, NE. 1) GO TO 5
   DU 6 = 1, 1, n
   ZZ = G05AAF(Y1)

6   \( x(1) = z z \)
   WRITE (2, ?77)
7 FORIAT (1H, 5X, \"THE EXACT SOLUTION\")
   DO 8 \( l = 1, i h \)
   \( j = i h - l + 1 \)
   8 WRITE (2, 170) \( x((j - 1) * i h + 1), i = 1, i h \)
   GO TO 75
5 DO 4 \( i = 1, n \)
4 \( x(1) = 0 \)
   \( x(m + 1) = 1 \)
75 \( m p = p \)
   \( m = n \)
666 WRITE (2, 159) \( n, p, irf, irg, ms, index \)
159 FORIAT (1H, 5X, \"ORDER OF MATRIX \( n = 15 / 6 \), SEMIBANDWIDTH \( p = 1 \),
   \(* 15 / 6 \), \"OUTERMOST OFF-DIAGONAL ENTRIES \( irf = 15 / 6 \),
   \(* 15 / 6 \), \"OUTERMOST OFF-DIAGONAL ENTRIES \( irg = 15 / 6 \),
   \(* 15 / 6 \), \"INDEX = 15 / 6 \)
   \( \epsilon p s = 1.0 e 4 \b \)
550 WRITE (2, 270) EPS
270 FORIAT (1H, 5X, \"ACCURACY EPS = 1, 20, 11. \")
   \( k = 0 \)
   \( q = 0 \)
C \textit{KII} IS AN UPPER LIMIT ON THE NUMBER OF ITERATIONS.
\begin{verbatim}
KII=0
IX=0
GV=0.0
DO 17 I=1,N
17 Y(I)=GV

C Y--HOLDS-- THE INITIAL APPROXIMATE SOLUTION Y0.
CALL GEVECT(N,P,MS,IX,A,B,C,H,S)
CALL RESID3D(Y,IX,S)
C S--HOLDS-- R 0
DO 24 I=1,N
24 RI(I)=S(I)
C RI--HOLDS-- R 0
CALL APNUFA3D(N,P,H,IRF,IRG,A,B,C,H)
CALL GEDRHS(N,A,S)
CALL FBSUBS3D(N,P,H,B,S)
C S--HOLDS-- R 0
DO 28 I=1,N
28 SI(I)=S(I)
C SI--HOLDS-- R 0

C CALCULATION OF AI, Y, RI.
IX=1

C --- THE INNER LOOP ---

C CALCULATION OF PRODUCT W*SI.
25 CALL RESID3D(SI,IX,VS)
C SI--HOLDS-- S0, VS--HOLDS-- W*S0.
Q=0
C CALCULATION OF AI
ANUI=0
DENOM=0
DO 41 I=1,N
ANUM =ANUM+RI(I)*S(I)
C RI--HOLDS-- R 0, SI--HOLDS-- R 0
41 DENOM=DENOM+SI(I)*VS(I)
C VS--HOLDS-- W*S0.
IF (DENOM.LE.0) GO TO 55
AI =ANUM /DENOM
WRITE (2,210) AI
210 FORIAT (1H,5X,'SCALAR AI = ',E20.11/
C CALCULATION OF Y, RI.
DO 19 I=1,N
Y(I) =Y(I)*AI*SI(I)
19 RI(I) =RI(I)-AI*VS(I)
C RI--HOLDS-- R(I+1)
C TEST OF THE CONVERGENCE WITH THE RECURSIVE RESIDUAL.
DO 23 I=1,N
IF (ABS(RI(I)).LT.EPS) Q=Q+1
23 CONTINUE
CALL ERMES(RI,Y,S1,S2,S3,S4)
WRITE (2,730) S1,S2,S3,S4
730 FORIAT (1H,5X,'LOG<10> OF SUM (RI*(T+S+TS)**(-1))*RI)',
* E20.11/5X,'LOG<10> OF EUCL. NORM OF RESIDUALS RI(I)= ',
* E20.11/5X,'LOG<10> OF R=(XX-X) = ',E20.11/5X,
* 'LOG<10> OF EUCL. NORM OF (XX-X) = ',E20.11/
K=K+1
IF (Q.EQ.N) GO TO 35
IF (K.GE.KM) GO TO 45
DO 21 I=1,N
\end{verbatim}
SUBROUTINE APNOFA3D(N,P,H,B,RI)
C THIS SUBROUTINE IS AN APPROXIMATE NORMALIZED FACTORIZATION
C OF A SQUARE MATRIX OF ORDER N. THE COEFFICIENT MATRIX IS
C SYMMETRIC, POSITIVE DEFINITE, SEVEN DIAGONAL MATRIX OF
C BANDWIDTHS H AND P (4<H<P<N), A, B, C, H ARE VECTORS EACH OF
C N-ELEMENTS CONTAINING RESPECTIVELY THE DIAGONAL, CO DIAGONAL.
C
M-TH DIAGUNAL AND P-TH DIAGUNAL ELEMENTS. THE MATRIX IS
FACTORIZED INTO DS T+S DS DS WHERE DS IS A DIAGONAL MATRIX
AND T+S IS A REAL UPPER TRIANGULAR MATRIX OF BANDWIDTHS M AND
P, WITH UNIT DIAGONAL ELEMENTS, NON-ZERO ELEMENTS IN SUPER
DIAGONAL, RETAINING IRF [1, M-1] AND IRG [1, P-1] OUTERMOST
OFF-DIAGONAL ENTRIES. T+S DENOTES THE TRANPOSE OF T+S.
THREE RESULT VECTORS OF LENGTH N AND TWO ARRAYS OF DIMENSION
[IRF, N-M+1], [IRG, N-P+1] ARE USED AS WORKSPACE.
N1 = N-M+1, N2 = N-P+1.
DIMENSION A(N), B(N), C(N), H(N), U(N), XX(N), YY(N)
COMMON F(IRF, N1), Y(IRG, N2)
INTEGER P
IX = 1
IRF1 = IRF + 1
IRG1 = IRG + 1
CALL GVECT (N, P, M, IX, A, B, C, H, U)
DO 11 I = P, N
XX(I-1) = B(I-1)
11 YY(I) = A(I)
A(I) = SQRT(A(I))
DO 1 J = 1, M-1
Z = B(I-1)/A(I-1)
A(I) = SQRT(A(I) - Z*Z)
1 B(I-1) = Z/A(I)
DO 6 J = 1, M-1
U(I) = C(J)/A(J)
V = B(H+J-2)/A(H+J-2)
IF (J GT IRF) GO TO 7
DO 2 I = 2, IRF1 - J
2 U(I) = B(I+J-2)*U(I-1)
DO 6 J = 1, M-1
IF (J LE IRF) GO TO 3
IF (J LE IRF) IP = IRF - J + 2
DO 3 I = IP, IRF
3 Z = Z + U(K)*U(K)
A(M+J-1) = SQRT(A(H+J-1) - Z - (U(H-1) + V)*V)
B(H+J-2) = V/A(H+J-2)
DO 6 I = 1, IRF
6 F(I, J) = U(I)/A(M+J-1)
14 CONTINUE
DO 18 I = P-1, N-1
18 B(I) = XX(I)
DO 19 I = P, N
19 A(I) = YY(I)
DO 13 I = 1, N
13 U(I) = A(I)
XX(P-1) = U(P-1)
M1 = P - M + 1
DO 12 J = 1, N-P+1
JJ = J + P - 1
C INITIAL GUESS VALUE OF D(P+J-1) = D(P+J-2)
A(P+J-1) = A(P+J-2)
562 Y(1, J) = H(J)/(A(J)*A(P+J-1))
V = B(P+J-2)/(A(P+J-2)*A(P+J-1))
SUBROUTINE GEURHS(N, A, S)

C THIS SUBROUTINE FORMS THE VECTOR Z, WHERE D=Z=S.

DO 33 I=2,IRF1-J
Y(I,J)=B(I+J-2)*Y(I-1,J)
IF (J.EQ.1,AND.I.EQ.M1) Y(I,J)=Y(I,J)*C(I+J-1)/(A(I+J-1)*
*A(P+J-1))
33 CONTINUE

IF (J,GT.IRF-1) GO TO 31

DO 33 I=2,IRF1-J
Y(I,J)=B(I+J-2)*Y(I-1,J)
IF (J.EQ.1,AND.I.EQ.1) Y(I,J)=Y(I,J)+C(I+J-1)/(A(I+J-1)*
*A(P+J-1))
31 CONTINUE

IF (J,GT.IP) GO TO 22

IF (J,LE.IP) IP=IRF-J+2
DO 16 I=IP,IRG
IF (I,GT.P+J-1,OR,J,GT.P-2) GO TO 335
22 Z=0
IF (I,GT.M) GO TO 23
DO 17 K=1,I-1
Z=Z+Y(K+I-M,J)*F(K+I+J-M,J)
17 CONTINUE
GO TO 24

23 DO 26 K=1,J-1
Z=Z+Y(K,J)*F(K-I+IRF1, I+J-IRF)
26 CONTINUE

Y(I,J)=B(I+J-2)*Y(I-1,J)=Z
IF (I,GT.M1) Y(I,J)=Y(I,J)+C(I+J-1)/(A(I+J-1)*A(P+J-1))
GO TO 16

C GENERATION OF MATRIX U (FROM PARTITIONED COEFF. MATRIX A).

335 Z=0
DO 21 K=1,I-1
221 Z=Z+Y(K-I+IRG1, I+J-IRG1)*Y(K+J,K)
21 CONTINUE

Y(I,J)=B(I+J-2)*Y(I-1,J)*Z
IF (I,GT.M1) Y(I,J)=Y(I,J)+C(I+J-1)/(A(I+J-1)*A(P+J-1))
715 CONTINUE

16 CONTINUE
15 Z=0
W=V
IF (I,GT.P-1) W=V+Y(I,J)
DO 161 K=1,P-2
IF (K,GT.IRG) GO TO 161
Z=Z+Y(K,J)*Y(K+J,K)
161 CONTINUE

Z2=0
DO 717 K=1,M-2
IF (K,GT.IRF) GO TO 717
Z2=Z2+F(K+J,K)*F(K,J)
717 CONTINUE

974 A(P+J-1)=SQRT(Y(P+J-1)/(A(P+J-1)+XX(P+J-2))
XX(P+J-2)=A(P+J-2)
GO TO 562
191 IF (J,GT.1) XX(P+J)=U(P+J-1)
XX(P+J-1)=A(P+J-1)
121 B(P+J-1)=B(P+J-2)/(A(P+J-2)+A(P+J-1))
12 CONTINUE
RETURN
CD IS A DIAGONAL MATRIX OF ORDER N, WHOSE NON-ZERO DIAGONAL ENTRIES ARE STORED IN A VECTOR A OF N-ELEMENTS. THE INPUT VECTOR S IS OVERWRITTEN BY THE RESULT.

**DIMENSION** A(N), S(N)

**DO 2 I = 1, N**

S(I) = S(I)/A(I)

**RETURN**

**END**

**SUBROUTINE FBSUBS3D(N,P,H,Q,S)**

**THIS SUBROUTINE SOLVES THE SET (TS*TS)*Y=S OF N-LINEAR EQUATIONS, WHERE TS IS AN UPPER TRIANGULAR MATRIX OF BANDWIDTHS H AND P, WITH UNIT ELEMENTS ON THE DIAGONAL.**

**TS** DENOTES THE TRANSPOSE OF TS. THE NON-ZERO ELEMENTS ARE ON SUPER DIAGONAL AND IRF(I,J-1), IRF(I,P-1) OUTERMOST OFF-DIAGONAL ENTRIES ARE RETAINED. THE SOLUTION IS EFFECTED BY A FORWARD-BACKWARD SUBSTITUTION PROCESS WHERE THE INPUT VECTOR S IS OVERWRITTEN SUCCESSIVELY BY THE INTERMEDIATE SOLUTION (OBTAINED BY FORWARD SUBSTITUTION) AND THE FINAL SOLUTION (OBTAINED BY BACK SUBSTITUTION). N1=N-H+1, N2=N-P+1.

**DIMENSION B(N), S(N)**

**COMMON /BLOCK5/IRF,P**

**INTEGER P**

**DO 44 I = 2, N**

S(I) = S(I) - B(I-1)*S(I-1)

**IF (H.EQ.N) GO TO 396**

**DO 78 I = H, P-1**

Z = 0

**DO 79 K = I-H+1, I-H+IRF**

Z = Z + F(K-I+H, I-M+1)*S(K)

**78 S(I) = S(I) - B(I-1)*S(I-1) - Z**

**396 DO 87 I = P, N**

Z = 0

**DO 88 K = I-H+1, I-H+IRF**

Z = Z + F(K-I+H, I-M+1)*S(K)

**87 S(I) = S(I) - B(I-1)*S(I-1) - Z**

**88 Z = Z + F(K-I+H, I-M+1)*S(K)**

Z1 = 0

**DO 39 K = I-P+1, I-P+IRF**

Z1 = Z1 + G(K-I+P, I-P+1)*S(K)

**89 S(I) = S(I) - B(I-1)*S(I-1) - Z**

**89 Z = Z + F(K-I+H, I-M+1)*S(K)**

**IF (H.LT. (N/2+1)) GO TO 64**

**IF (H.EQ.N) GO TO 65**

**IF (P.GT. (N/2+1)) GO TO 183**

**CASE : H>(N/2+1), P DOES NOT EXIST**

**DO 47 I = 1, N-H**

**I = H-1**

Z = 0

**DO 48 J = I+1, N**

**IF (J-I+H, GT, IRF) GO TO 48**

Z = Z + F(I-J+H, J-M+1)*S(J)

**CONTINUE**

**47 S(I) = S(I) - B(I)*S(I-1) - Z**

**48 Z = Z + F(I-J+H, J-M+1)*S(J)**

**49**

**DO 51 J = 1, 2*M-N-1**

I = M-1

Z = 0

**DO 51 J = M, N**

**IF (I-J+H, GT, IRF) GO TO 51**

Z = Z + F(I-J+H, J-M+1)*S(J)
51 CONTINUE
49 S(I) = S(I) - B(I) * S(I+1) - Z
IF (H.EQ.N) GO TO 66
DO 52 II = 1, N-I
I = N-M+1 - II
Z = 0
DO 53 J = M, I + M - 1
IF (I-J+M, GT. IEF) GO TO 53
Z = Z + F(I-J+M, J-M+1) * S(J)
53 CONTINUE
52 S(I) = S(I) - B(I) * S(I+1) - Z
GO TO 66
C---- CASE : I>N/2+1, P>N/2+1
183 IF (P, EQ. N) GO TO 252
DO 231 II = 1, N - P
I = N - I
Z = 0
Z1 = 0
DO 232 J = I + 1, N
IF (I-J+M, GT. IEF) GO TO 261
Z = Z + F(I-J+M, J-M+1) * S(J)
261 IF (I-J+P, GT. IRE) GO TO 232
Z1 = Z1 + G(I-J+P, J-P+1) * S(J)
232 CONTINUE
231 S(I) = S(I) - B(I) * S(I+1) - Z - Z1
252 IF (H, EQ. N) GO TO 253
DO 233 II = 1, P - M
I = P - I
Z = 0
DO 234 J = I + 1, N
IF (I-J+M, GT. IEF) GO TO 234
Z = Z + F(I-J+M, J-M+1) * S(J)
234 CONTINUE
Z1 = 0
DO 235 J = M, N
IF (I-J+P, GT. IRE) GO TO 235
Z1 = Z1 + G(I-J+P, J-P+1) * S(J)
235 CONTINUE
233 S(I) = S(I) - B(I) * S(I+1) - Z - Z1
253 DO 236 II = N - M + 1, M + 1
I = N - I
Z = 0
Z1 = 0
DO 237 J = M, N
IF (I-J+M, GT. IEF) GO TO 266
Z = Z + F(I-J+M, J-M+1) * S(J)
266 IF (I-J+P, GT. IRE) GO TO 237
Z1 = Z1 + G(I-J+P, J-P+1) * S(J)
237 CONTINUE
236 S(I) = S(I) - B(I) * S(I+1) - Z - Z1
IF (H, EQ. N) GO TO 66
DO 238 II = H, P - 1
I = N - I
Z = 0
DO 239 J = M, I + M - 1
IF (I-J+H, GT. IEF) GO TO 239
Z = Z + F(I-J+H, J-H+1) * S(J)
239 CONTINUE
Z1 = 0
DO 241 J = M, N
IF (I-J+P, GT, IRG) GO TO 241
Z1 = Z1 + G(I-J+P, J-P+1)*S(J)

241 CONTINUE

238 S(I) = S(I) - B(I)*S(I+1) - Z - Z1
IF (P, EQ, N) GO TO 65
DO 242 II = P, N-1
I = N-II
Z = 0
Z1 = 0
DO 243 J = H, I+1
IF (I-J+H, GT, IRG) GO TO 271
Z = Z + F(I-J+H, J-H+1)*S(J)

271 IF (I-J*P, GT, IRG) GO TO 243
Z1 = Z1 + G(I-J+P, J-P+1)*S(J)

243 CONTINUE

242 S(I) = S(I) - B(I)*S(I+1) - Z - Z1
GO TO 66

64 IF (P, GT, N, OR, P, LE, M) GO TO 821
IF (P, LT, (N/2+1)) GO To 444
IF (((N-N/2*2), EQ, 0), AND, (M, EQ, N/2)) GO To 67
CASE: M < [N/2+1], P > [N/2+1]. IF (P, EQ, N) GO TO 274
DO 94 II = P, N-1
I = N-II
Z = 0
Z1 = 0
DO 95 J = I+1, N
IF (I-J+P, GT, IRG) GO TO 211
Z1 = Z1 + G(I-J+P, J-P+1)*S(J)

211 IF (I-J+M, GT, IRF) GO TO 95
Z = Z + F(I-J+M, J-M+1)*S(J)

95 CONTINUE

94 S(I) = S(I) - B(I)*S(I+1) - Z - Z1
DO 96 II = N-M, P+17M-1
I = N-II
Z = 0
DO 97 J = I+1, N
IF (I-J+P, GT, IRG) GO TO 97
Z = Z + F(I-J+P, J-P+1)*S(J)

97 CONTINUE

96 S(I) = S(I) - B(I)*S(I+1) - Z - Z1
DO 101 II = H, N-H
I = N-II
Z = 0
DO 102 J = I+1, I+1
IF (I-J+H, GT, IRF) GO TO 102
Z = Z + F(I-J+H, J-H+1)*S(J)

102 CONTINUE

101 S(I) = S(I) - B(I)*S(I+1) - Z - Z1
DO 104 II = N-H+1, P-1
I = N - 1
Z = 0
DO 105 J = M + 1, I + M - 1
IF (I - J + M, GT, IRF) GO TO 105
Z = Z + F(I - J + M, J + M - 1) * S(J)
105 CONTINUE
Z1 = 0
DO 106 J = P, N
IF (I - J + P, GT, IRG) GO TO 106
Z1 = Z1 + G(I - J + P, J + P - 1) * S(J)
106 CONTINUE
104 S(I) = S(I) - B(I) * S(I + 1) - Z - Z1
IF (P, EQ, N) GO TO 106
DO 121 I = P, N - 1
I = N - I
Z = 0
DO 124 J = 0, I - M - 1
IF (I - J + M, GT, IRF) GO TO 124
Z = Z + F(I - J + M, J + M - 1) * S(J)
124 CONTINUE
Z1 = 0
DO 125 J = I + 1, N
IF (I - J + M, GT, IRF) GO TO 125
Z1 = Z1 + G(I - J + P, J + P - 1) * S(J)
125 CONTINUE
121 S(I) = S(I) - B(I) * S(I + 1) - Z - Z1
GO TO 66
C--- CASE : M = N/2 , P >= [N/2 + 1] .
67 IF (P, EQ, N) GO TO 256
DU 01 I = 1, N - P
I = N - I
Z = 0
Z1 = 0
DO 62 J = I + 1, N
IF (I - J + M, GT, IRF) GO TO 281
Z = Z + F(I - J + M, J + M - 1) * S(J)
281 IF (I - J + P, GT, IRG) GO TO 62
Z1 = Z1 + G(I - J + P, J + P - 1) * S(J)
62 CONTINUE
61 S(I) = S(I) - B(I) * S(I + 1) - Z - Z1
256 DU 141 I = 1, P - M - 1
I = P - I
Z = 0
DO 142 J = I + 1, N
IF (I - J + M, GT, IRF) GO TO 142
Z = Z + F(I - J + M, J + M - 1) * S(J)
142 CONTINUE
Z1 = 0
DO 145 J = P, N
IF (I - J + P, GT, IRG) GO TO 145
Z1 = Z1 + G(I - J + P, J + P - 1) * S(J)
145 CONTINUE
141 S(I) = S(I) - B(I) * S(I + 1) - Z - Z1
I = M
Z = 0
DO 146 J = M + 1, N - 1
IF (I - J + M, GT, IRF) GO TO 146
Z = Z + F(I - J + M, J + M - 1) * S(J)
146 CONTINUE
Z1 = 0
DO 147 J=M+P,N
IF (I-J+P, GT, IRF) GO TO 147
Z = Z + G(I-J+P, J-P+1) * S(J)
147 CONTINUE
S(I) = S(I) - B(I) * S(I+1) - Z - Z1
DO 148 I = I, N + P - M - 1
I = M - II
Z = 0
DO 149 J = M + I + M - 1
IF (I-J+M, GT, IRF) GO TO 149
Z = Z + F(I-J+M, J-M+1) * S(J)
149 CONTINUE
Z = 0
DO 150 I = N - II
Z = 0
DO 151 J = I, N + P - M - 1
IF (I-J+M, GT, IRF) GO TO 151
Z = Z + F(I-J+M, J-M+1) * S(J)
151 CONTINUE
Z = 0
DO 152 I = N - II
I = M - II
Z = 0
DO 153 J = I, N + P - M - 1
IF (I-J+M, GT, IRF) GO TO 153
Z = Z + F(I-J+M, J-M+1) * S(J)
153 CONTINUE
Z = 0
DO 154 I = N - II
Z = 0
DO 155 J = I, P - 1
IF (I-J+M, GT, IRF) GO TO 231
Z = Z + G(I-J+P, J-P+1) * S(J)
231 IF (I-J+M, GT, IRF) GO TO 55
Z = Z + F(I-J+M, J-M+1) * S(J)
55 CONTINUE
S(I) = S(I) - B(I) * S(I+1) - Z - Z1
DO 56 I = I, M - P - 1
I = M - II
Z = 0
DO 57 J = I, I + M - 1
IF (I-J+M, GT, IRF) GO TO 57
Z = Z + F(I-J+M, J-M+1) * S(J)
57 CONTINUE
Z = 0
DO 91 J = I, I + M - 1
IF (I-J+M, GT, IRF) GO TO 91
Z = Z + G(I-J+P, J-P+1) * S(J)
91 CONTINUE
S(I) = S(I) - B(I) * S(I+1) - Z - Z1
DO 124 I = P, N - P
I=N-11
Z=0
DO 125 J=I*1, I+1, 1
IF (I-J+M GT IRF) GO TO 125
Z=Z+F(I-J+M, J-M+1)*S(J)
125 CONTINUE
Z=0
DO 126 J=I+1, I+ P-1
IF (I-J+P GT IRG) GO TO 126
Z1=Z1+G(I-J+ P, J-M+1)*S(J)
126 CONTINUE
124 S(I)=S(I)-B(I)*S(I)+1-Z-Z1
DO 127 II=N+ P+1, N-M
I=N-II
Z=0
DO 128 J=1, I+1, P-1
IF (I-J*M GT IRF) GO TO 128
Z=Z+F(I-J+M, J-M+1)*S(J)
128 CONTINUE
Z1=0
DO 129 J=I, I+ P-1
IF (I-J+P GT IRG) GO TO 129
Z1=Z1+G(I-J+ P, J-M+1)*S(J)
129 CONTINUE
127 S(I)=S(I)-B(I)*S(I+1)-Z-Z1
DO 58 II=1, N-1
I=M-II
Z=0
DO 59 J=1, M-1
IF (I-J*M GT IRF) GO TO 59
Z=Z+F(I-J+M, J-M+1)*S(J)
59 CONTINUE
Z1=0
DO 93 J=P+1, P+1
IF (I-J+P GT IRG) GO TO 93
Z1=Z1+G(I-J+ P, P+1)*S(J)
93 CONTINUE
58 S(I)=S(I)-B(I)*S(I+1)-Z-Z1
GO TO 66
CASE : H<[(N/2)+1], P=N/2
107 DO 108 II=1, N-P
I=N-II
Z=0
Z1=0
DO 109 J=I, N
IF (I-J+P GT IRG) GO TO 251
Z1=Z1+G(I-J+ P, J-M+1)*S(J)
251 IF (I-J+H GT IRG) GO TO 109
Z=Z+F(I-J+H, J-M+1)*S(J)
109 CONTINUE
108 S(I)=S(I)-B(I)*S(I+1)-Z-Z1
I=P
Z=0
DO 111 J=I, N
IF (I-J+H GT IRG) GO TO 111
Z=Z+F(I-J+H, J-M+1)*S(J)
111 CONTINUE
Z1=0
DO 112 J=P+1, N-1
IF (I-J+P GT IRG) GO TO 112
\[ z_{1} = z_{1} + g(I-J+P,J=1+P) \]

**CONTINUE**

\[ s(I) = s(I) - B(I) \cdot s(I+1) - z = z_{1} \]

**DO** 113 \( I = N - 1 \)

\( z = 0 \)

**DO** 114 \( J = I + 1, N \)

IF \( I-J+M, GT, IRI; \) GO TO 114

\( z = z + f(I-J+M,J=M+1) \cdot s(J) \)

**CONTINUE**

\[ z_{1} = 0 \]

**DO** 115 \( J = I, P+1 \)

IF \( I-J+P, GT, IRI; \) GO TO 115

\( z_{1} = z_{1} + g(I-J+P,J=P+1) \cdot s(J) \)

**CONTINUE**

\[ s(I) = s(I) - B(I) \cdot s(I+1) - z = z_{1} \]

**DO** 116 \( I = M \)

\( z = 0 \)

**DO** 117 \( J = I, P+1 \)

IF \( I-J+P, GT, IRI; \) GO TO 117

\( z_{1} = z_{1} + g(I-J+P,J=P+1) \cdot s(J) \)

**CONTINUE**

\[ s(I) = s(I) - B(I) \cdot s(I+1) - z = z_{1} \]

**DO** 118 \( I = N, P+1 \)

\( z = 0 \)

**DO** 119 \( J = I, P+1 \)

IF \( I-J+P, GT, IRI; \) GO TO 119

\( z = z + f(I-J+P,J=P+1) \cdot s(J) \)

**CONTINUE**

\[ s(I) = s(I) - B(I) \cdot s(I+1) - z = z_{1} \]

**GO TO** 66

**IF** \( (N=M/2*2*2), EQ, 0, AND, I, EQ, N/2) \) GO TO 822

**CASE:** \( M < [N/2+1] \), \( P \) DOES NOT EXIST

**DO** 381 \( I = 1, M-1 \)

\( I = N-I \)

\( z = 0 \)

**DO** 382 \( J = I+1, N \)

IF \( I-J+M, GT, IRI; \) GO TO 382

\( z = z + f(I-J+M,J=M+1) \cdot s(J) \)

**CONTINUE**

\[ s(I) = s(I) - B(I) \cdot s(I+1) - z \]

**DO** 383 \( I = 1, N-2*M+1 \)

\( I = N-M+1-I \)

\( z = 0 \)

**DO** 384 \( J = I+1, I+I-I \)

IF \( I-J+M, GT, IRI; \) GO TO 384

\( z = z + f(I-J+M,J=M+1) \cdot s(J) \)

**CONTINUE**

\[ s(I) = s(I) - B(I) \cdot s(I+1) - z \]
SUBROUTINE RESID3D (X, IND, S)
C THIS SUBROUTINE CALCULATES THE CORRESPONDING TO THE SET W*X=S
C RESIDUALS WHERE THE COEFFICIENT MATRIX IS OF ORDER N;
C SYMMETRIC, POSITIVE DEFINITE, SEVEN DIAGONAL OF BANDWIDTH M
C AND P (4M<P<2N/2) AND A, B, C, H, S ARE VECTORS CONTAINING
C RESPECTIVELY THE DIAGONAL, CO DIAGONAL, M-TH DIAGONAL,
C P-TH DIAGONAL AND CONSTANT ELEMENTS. THE RESIDUAL IS
C OVERWRITTEN BY VECTOR S. IF INDEX IND=1 THEN THE SUBROUTINE
C CALCULATES THE PRODUCTS OF THE GIVEN SPARSE MATRIX W BY THE
C INPUT VECTOR X.
D DIMENSION A(N), B(N), C(N), H(N), S(N), X(N)
D INTEGER P
C Ijong BLOCK1/N7P, H
C CALL GEVEC(N, P, M, IND; A, B, C, H, S)
C S(I)=S(I)-((A(I)*X(I)+B(I)*X(2)+C(I)*X(M)+H(I)*X(P))
D DO 5 I=2; H+1
C S(I)=S(I)-(D(I-1)+X(I-1)+A(I)+X(I)+B(I)*X(I+1)+
C C(I)*X(I+M+1)+H(I)*X(I+P-1))
C 5 CONTINUE
D DO 13 I=H, P-1
C S(I)=S(I)+(C(I+1)+X(I+H)+B(I)+X(I+1)+A(I)*X(I))
C 13 CONTINUE
C RETURN
C END
*B(I)*X(I+1)+C(I)*X(I+H-1)+H(I)*X(I+P-1))

13 CONTINUE
DO 14 I=P,N-P+1
S(I)=S(I)+(H(I-P+1)*X(I-P+1)+C(I-M+1)*X(I-M+1)*B(I-1)*
*X(I-1)+A(I)*X(I)+B(I)*X(I+1)+C(I)*X(I+H-1)+H(I)*X(I+P-1))

14 CONTINUE
DO 15 I=N-P+2,N-H+1
S(I)=S(I)+(H(I-P+1)*X(I-P+1)+C(I-M+1)*X(I-M+1)*B(I-1)*
*X(I-1)+A(I)*X(I)+B(I)*X(I+1)+C(I)*X(I+H-1))

15 CONTINUE
DO 16 I=N-H+2,N-1
S(I)=S(I)-(H(I-P+1)*X(I-P+1)+C(I-M+1)*X(I-M+1)*B(I-1)*
*X(I-1)+A(I)*X(I)+B(I)*X(I+1))

16 CONTINUE
S(N)=S(N)-(H(N-P+1)*X(N-P+1)+C(N-M+1)*X(N-M+1)*B(N-1)*
*X(N-1)+A(N)*X(N))
IF (IND.EQ.1) GO TO 20
RETURN

20 DO 12 I=1,N
12 S(I)=-S(I)
RETURN
END