Finite difference methods for solving mildly nonlinear elliptic partial differential equations

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

Additional Information:

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

Metadata Record: https://dspace.lboro.ac.uk/2134/10417

Publisher: © Jehad Abdul Hamid El-Nakla

Please cite the published version.
This item was submitted to Loughborough University as a PhD thesis by the author and is made available in the Institutional Repository (https://dspace.lboro.ac.uk/) 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/
FINITE-DIFFERENCE METHODS FOR SOLVING
MILDLY NONLINEAR ELLIPTIC PARTIAL
DIFFERENTIAL EQUATIONS

by

JEHAD ABDUL HAMID EL-NAKLA, B.Sc., M.Sc.

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

Supervisors: Professor D.J. Evans, Ph.D., D.Sc.
Mr. G.N.C. Grant.

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 higher degree.

Jehad A.H. El-Nakla.
To: PALESTINE

where I was born,
I was uprooted
and I will return.
Acknowledgements

I wish to express my gratitude and appreciation to Professor D.J. Evans for supervising me during the course of this research and for his kind assistance and help in solving the problems I faced.

Also I would like to thank Mr. G.N.C. Grant for his guidance at the early stage of this work.

Furthermore, I would particularly like to thank my father for his financial support and also everyone else who helped in this respect.

Finally, I would like to thank Mrs. B. Wright for her professional typing of this thesis.
Abstract

This thesis is concerned with the solution of large systems of linear algebraic equations in which the matrix of coefficients is sparse. Such systems occur in the numerical solution of elliptic partial differential equations by finite-difference methods. By applying some well-known iterative methods, usually used to solve linear PDE systems, the thesis investigates their applicability to solve a set of four mildly nonlinear test problems.

In Chapter 4 we study the basic iterative methods and semi-iterative methods for linear systems. In particular, we derive and apply the GS, SOR, SSOR methods and the SSOR method extrapolated by the Chebyshev acceleration strategy.

In Chapter 5, three ways of accelerating the SOR method are described together with the applications to the test problems. Also the Newton-SOR method and the SOR-Newton method are derived and applied to the same problems.

In Chapter 6, the Alternating Directions Implicit methods are described. Two versions are studied in detail, namely, the Peaceman-Rachford and the Douglas-Rachford methods. They have been applied to the test problems for cycles of 1, 2 and 3 parameters.

In Chapter 7, the conjugate gradients method and the conjugate gradient acceleration procedure are described together with some preconditioning techniques. Also an approximate LU-decomposition algorithm (ALUBOT algorithm) is given and then applied in conjunction with the Picard and Newton methods.

Chapter 8 contains the final conclusions.
# Contents

## CHAPTER 1: INTRODUCTION.

1.1 Classification of Partial Differential Equations.  
1.2 Linear and Nonlinear Problems.  
1.3 Types of Boundary Conditions.  
1.4 Well-posed Problems.  

## CHAPTER 2: BASIC MATHEMATICAL PRELIMINARIES.

2.1 Finite Difference Approximations to Derivatives.  
2.2 Basic Matrix Properties and Concepts.  
2.3 Vectors and Matrix Norms.  
2.4 Eigenvalues and Eigenvectors.  
2.5 Property (A) and Consistently Ordered Matrix.  
2.6 LU Decomposition.  
2.7 The Power Method.  
2.8 Convergence of Sequence of Matrices.  
2.9 Condition Number and Error Estimates.  
2.10 Variational Principles and Conjugate Gradients.

## CHAPTER 3: ALGORITHMS FOR SOLVING UNIVARIATE NONLINEAR EQUATIONS.

3.1 Introduction.  
3.2 A survey of Iterative Methods.  
  3.2.1 Bisection Method.  
  3.2.2 Regula Falsi Method.  
  3.2.3 Secant Method.  
  3.2.4 Newton-Raphson Method.  
3.3 Convergence of Fixed Point Method.  
3.4 Two-Point Newton Method.  
  3.4.1 Basic Two-Point Algorithm.  
  3.4.2 Modified Two-Point Algorithm.  
3.5 Experimental Results.
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.2</td>
<td>The Stationary Case.</td>
<td></td>
</tr>
<tr>
<td>6.2.1</td>
<td>Consistency and Convergence.</td>
<td>186</td>
</tr>
<tr>
<td>6.2.2</td>
<td>Selection of Parameters</td>
<td>187</td>
</tr>
<tr>
<td>6.3</td>
<td>The Commutative Case.</td>
<td>190</td>
</tr>
<tr>
<td>6.4</td>
<td>Good Acceleration Parameters.</td>
<td>195</td>
</tr>
<tr>
<td>6.5</td>
<td>The Peaceman-Rachford (P-R) Method.</td>
<td>199</td>
</tr>
<tr>
<td>6.5.1</td>
<td>Introduction.</td>
<td>201</td>
</tr>
<tr>
<td>6.5.2</td>
<td>Choice of Parameters and Convergence.</td>
<td>204</td>
</tr>
<tr>
<td>6.6</td>
<td>The Douglas-Rachford (D-R) Method.</td>
<td>201</td>
</tr>
<tr>
<td>6.6.1</td>
<td>Introduction.</td>
<td>211</td>
</tr>
<tr>
<td>6.6.2</td>
<td>Choice of Parameters and Convergence.</td>
<td>215</td>
</tr>
<tr>
<td>6.7</td>
<td>Mildly Nonlinear Problems and Applications.</td>
<td></td>
</tr>
<tr>
<td>6.7.1</td>
<td>The Model Problem.</td>
<td>220</td>
</tr>
<tr>
<td>6.7.2</td>
<td>Method of Solution.</td>
<td>222</td>
</tr>
<tr>
<td>6.7.3</td>
<td>Application and Results.</td>
<td>223</td>
</tr>
<tr>
<td>6.7.4</td>
<td>Discussion and Comments.</td>
<td>245</td>
</tr>
</tbody>
</table>

CHAPTER 7: VARIATIONAL METHODS FOR SOLVING NONLINEAR PROBLEMS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1</td>
<td>Conjugate Gradient Method.</td>
<td>248</td>
</tr>
<tr>
<td>7.1.1</td>
<td>Introduction.</td>
<td>248</td>
</tr>
<tr>
<td>7.1.2</td>
<td>Conjugate Gradient Method.</td>
<td>249</td>
</tr>
<tr>
<td>7.1.3</td>
<td>Conjugate Gradient Acceleration.</td>
<td>252</td>
</tr>
<tr>
<td>7.2</td>
<td>Preconditioning the Conjugate Gradient Method.</td>
<td>255</td>
</tr>
<tr>
<td>7.2.1</td>
<td>Introduction.</td>
<td>255</td>
</tr>
<tr>
<td>7.2.2</td>
<td>Preconditioning by Incomplete Factorization.</td>
<td>257</td>
</tr>
<tr>
<td>7.2.3</td>
<td>Preconditioning by SSOR Method.</td>
<td>261</td>
</tr>
<tr>
<td>7.3</td>
<td>Mildly Nonlinear Problems.</td>
<td>262</td>
</tr>
<tr>
<td>7.3.1</td>
<td>Method of Solution.</td>
<td>262</td>
</tr>
<tr>
<td>7.3.2</td>
<td>Experimental Results.</td>
<td>265</td>
</tr>
<tr>
<td>7.4</td>
<td>The LU-Decomposition Algorithm for Two Dimensional P.D.E.'s.</td>
<td>278</td>
</tr>
<tr>
<td>7.4.1</td>
<td>Introduction.</td>
<td>278</td>
</tr>
<tr>
<td>7.4.2</td>
<td>The Approximate LU-Decomposition Algorithm (ALUBOT Algorithm).</td>
<td>282</td>
</tr>
</tbody>
</table>
7.5 Derivation of Newton-ALUBOT Method.  
7.6 Iterative Procedures and Experimental Results.

CHAPTER 8 : CONCLUSIONS AND FINAL COMMENTS.  
REFERENCES :  
APPENDIX : FORTRAN 77 PROGRAMS.
Chapter One

INTRODUCTION

1.1 Classification of Partial Differential Equations
1.2 Linear and Nonlinear Problems
1.3 Types of Boundary Conditions
1.4 Well-posed Problems
1.1 CLASSIFICATION OF PARTIAL DIFFERENTIAL EQUATIONS

Most of the problems in engineering and physics involve the rates of change of unknown quantities (dependent variables) with respect to two or more independent variables. The mathematical formulation of these problems produces a partial differential equation, p.d.e., (or a set of such equations). Such an equation is the general second order p.d.e.

\[ a \frac{\partial^2 u}{\partial x^2} + 2b \frac{\partial^2 u}{\partial x \partial y} + c \frac{\partial^2 u}{\partial y^2} + d \frac{\partial u}{\partial x} + e \frac{\partial u}{\partial y} + gu = f, \quad (1.1.1) \]

where \(a, b, c, d, e, g\) and \(f\) are continuous functions of \(x, y, u, \frac{\partial u}{\partial x}\) and \(\frac{\partial u}{\partial y}\), in some region \(R\), in the \((x, y)\) plane. Equations of the form (1.1.1) are classified according to the quantity \(b^2 - ac\). The equation is said to be

a) elliptic, if \(b^2 - ac < 0\)

b) parabolic, if \(b^2 - ac = 0\) \hspace{1cm} (1.1.2)

c) hyperbolic, if \(b^2 - ac > 0\).

Some well known examples are:

**Elliptic**

Laplace's equation \[ \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \] \hspace{1cm} (1.1.3a)

Poisson's equation \[ \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y) \] \hspace{1cm} (1.1.3b)

**Parabolic**

The diffusion equation \[ \frac{\partial^2 u}{\partial x^2} = \frac{\partial u}{\partial y} \] \hspace{1cm} (1.1.4)

**Hyperbolic**

The wave equation \[ \frac{\partial^2 u}{\partial x^2} = \frac{\partial^2 u}{\partial y^2} \] \hspace{1cm} (1.1.5)
\[ \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \] is referred to as the Laplacian operator and is written as \( \nabla^2 \). Also (1.1.1) is said to be of order \( n \) if the order of the highest derivative involved is \( n \).

1.2 **LINEAR AND NONLINEAR PROBLEMS**

Equation (1.1.1) can be categorized in the sense of linearity or nonlinearity depending on the nature of the coefficients \( a, b, c, d, e, g \) and \( f \). Equation (1.1.1) is said to be:

(i) **Linear** if all the coefficients are constants or functions of one of both independent variables \( x \) and \( y \).

(ii) **Nonlinear** if any of the coefficients \( a, b, \ldots, f \) are functions of the dependent variable \( u \), or its derivatives.

(iii) **Semi-Linear** if the coefficients \( a, b, \ldots, f \) are only functions (not constants) of the independent variable \( x \) and \( y \).

(iv) **Quasi-Linear** if the coefficients \( a, b \) and \( c \) are functions of \( x, y, u, \frac{\partial u}{\partial x} \) and \( \frac{\partial u}{\partial y} \) but not of the second-order derivatives.

(v) **Homogeneous** if \( f = 0 \), otherwise it is called inhomogeneous.

(vi) **Self-Adjoint** if the equation (1.1.1) can be replaced by,

\[
\frac{\partial}{\partial x} \left[ a(x) \frac{\partial u}{\partial x} \right] + \frac{\partial}{\partial y} \left[ c(y) \frac{\partial u}{\partial y} \right] + gu + f = 0 .
\]

Knowing the type of a p.d.e. and whether it is a linear or nonlinear equation will decide on which method will be used to solve the problem.
1.3 TYPES OF BOUNDARY CONDITIONS

The solution of the p.d.e. (1.1.1) must also satisfy the boundary conditions which arise from the problem formulation. There are five main types of such conditions defined on the boundary $C$ of the closed region $R$, these are:

1. **Dirichlet problem**: where the solution $u$ is given at each point on $C$, i.e.
   \[ u|_C = \phi(x, y) . \]

   If $\phi \equiv 0$ the problem is called *Homogeneous Dirichlet* problem.

2. **Neumann problem**: where the values of the normal derivatives are given on $C$, i.e.
   \[ \frac{\partial u}{\partial n}|_C = \psi(x, y) \]

   $\frac{\partial}{\partial n}$ denotes the directional derivative along the outward normal to $C$.

3. **Robin's problem**: where the solution $u$ has to satisfy a combination of $u$ and its derivatives on the boundary $C$, i.e.
   \[ \left[ \alpha(x, y)u + \beta(x, y) \frac{\partial u}{\partial n} \right] = \psi(x, y) , \]

   where $\alpha(x, y) > 0$, $\beta(x, y) > 0$ for $(x, y) \in C$.

4. **Mixed problem**: (or Churchill problem as referred to by Kersten (1969)) where $u$ is given on part of $C$ and $\frac{\partial u}{\partial n}$ is given on the remained of $C$.

5. **Periodic Boundary problem**: in this case we seek the solution such that it satisfies the periodicity conditions, for example,
   \[ u|_x = u|_{x+\ell}, \quad \frac{\partial u}{\partial n}|_x = \frac{\partial u}{\partial n}|_{x+\ell} \]

   $u$ is periodic in the $x$-direction, where $\ell$ is called the period.
1.4 WELL POSED PROBLEMS

In physical applications, the solution of a partial differential equation has to satisfy specified auxiliary data which may appear either as boundary and/or initial value conditions.

We say that the differential equation problem is: well-posed (or properly posed) if there exists one and only one (unique) solution to the problem and this solution depends continuously on the auxiliary data. If the problem is not well-posed we say it is ill-posed.

The continuous dependence criterion is an expression of the stability of the solution, that is a small change (perturbation) in the auxiliary data causes only a small change in the solution. In this case we say the problem is well-conditioned otherwise it is ill-conditioned.

To illustrate the well-posedness condition, we consider the following example devised by Hadamard (see R. Vichnevetsky, 1981). Consider the Laplace equation in two dimensions,

\[
\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 , \quad (1.4.1)
\]

with the given initial value data,

\[
\begin{align*}
\frac{\partial u}{\partial y} (x, 0) &= \frac{1}{k} \sin(kx) \\
u(x, 0) &= 0
\end{align*} \quad (1.4.2)
\]

where \( k > 0 \) and \( \) is a real parameter.

An analytical solution can be determined by separation of variables,

\[
u(x, y) = \frac{1}{k^2} \sin(kx) \sinh(ky) . \quad (1.4.3)\]
As we let \( k \to \infty \), then the initial data (1.4.2) converge uniformly to,

\[
u(x, 0) = 0 , \quad \frac{\partial u}{\partial x}(x, 0) = 0 .
\] (1.4.4)

The solution of (1.4.1) with (1.4.4) is unique and trivial, i.e.,

\[
u(x, y) = 0 ,
\] (1.4.5)

But the analytical solution, in (1.4.3) does not converge to zero for \( y > 0 \) but becomes very large as \( k \to \infty \).

Consequently, approximating the auxiliary data arbitrarily does not guarantee a corresponding approximation for the solution. We conclude that the data of (1.4.2) is ill-posed for the Laplace equation and cannot be associated together with any physical phenomenon, (see W.F. Ames, 1977).
Chapter Two

BASIC MATHEMATICAL PRELIMINARIES

2.1 Finite Difference Approximations to Derivatives
2.2 Basic Matrix Properties and Concepts
2.3 Vectors and Matrix Norms
2.4 Eigenvalues and Eigenvectors
2.5 Property (A) and Consistently Ordered Matrices
2.6 LU Decomposition
2.7 The Power Method
2.8 Convergence of Sequence of Matrices
2.9 Condition Number and Error Estimates
2.10 Variational Principles and Conjugate Gradients
2.1 *Finite Difference Approximation to Derivatives*

Let us assume, without loss of generality, that the p.d.e. problem is to be solved in a connected region R, in the \( x - y \) plane, with boundary C. We cover R by rectangular grid (sometimes called a mesh or net), with grid sizes \( \Delta x, \Delta y \) in the \( x, y \) directions respectively. If we choose any convenient point \((x_0, y_0)\) in the region and the "mesh coordinates" \( i, j \) then,

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

The intersection of these points are referred to as the (unknown) grid points.

Suppose we have \((x_0, y_0) \equiv (0, 0)\) and \( \Delta x = \Delta y = h \), then we have the following illustration.

From Figure (2.1.1), \( P_{ij} \equiv (ih, jh) \).
The following theorems are elementary but fundamental in the derivation of finite difference approximations. The proofs can be found in most calculus books, (W. Kaplan, 1973).

**Theorem 2.1.1 - Taylor's Theorem**

Let $u(x)$ have $n + 1$ continuous derivatives on $a \leq x \leq b$ for some $n > 0$ and $x, x_0 \in [a,b]$. Then the Taylor series expansion of $u(x)$ can be written in the form,

$$u(x) = P_n(x) + R_{n+1}(x)$$ (2.1.1)

$$P_n(x) = u(x_0) + \frac{(x-x_0)}{1!} \frac{du(x_0)}{dx} + \ldots + \frac{(x-x_0)^n}{n!} \frac{d^n u(x_0)}{dx^n}$$ (2.1.2)

$$R_{n+1} = \frac{1}{n!} \int_{x_0}^{x} (x-t)^{n} \frac{d^{n+1} u(t)}{dt^{n+1}} \cdot dt$$

$$= \frac{(x-x_0)^{n+1}}{(n+1)!} \frac{d^{n+1} u(\xi)}{dt^{n+1}}$$ (2.1.3)

for some $\xi$ between $x_0$ and $x$.

**Theorem 2.1.2 - Taylor's Theorem in two dimensions**

Let $(x_0, y_0)$ and $(x_0 + \xi, y_0 + \eta)$ be given points and assume $u(x, y)$ is $(n + 1)$ times continuously differentiable for all $(x, y)$ in some neighbourhood of $L(x_0, y_0; x_0 + \xi, y_0 + \eta)$. Then

$$u(x_0 + \xi, y_0 + \eta) = u(x_0, y_0) + \sum_{j=1}^{n} \frac{1}{j!} \left[ \frac{\partial}{\partial x} + \frac{n}{\partial y} \right]^j u(x, y) \bigg|_{x=x_0, y=y_0}$$

$$+ \frac{1}{(n+1)!} \left[ \frac{\partial}{\partial x} + \frac{n}{\partial y} \right]^{n+1} u(x, y) \bigg|_{x=x_0 + \theta_1 \xi, y=y_0 + \theta_2 \eta}$$

(2.1.4)
for some $0 \leq \theta_1, \theta_2 \leq 1$. The point $(x_0 + \theta_1 \xi, y_0 + \theta_2 \eta)$ is an unknown point on the line $L(x_0, y_0; x_0 + \xi, y_0 + \eta)$. By using the above theorems, we can now write $u_{i+1,j}$ and $u_{i,j+1}$ (using $\Delta x = \Delta y = h$),

$$u_{i+1,j} = u_{i,j} + h \frac{\partial u}{\partial x}_{i,j} + \frac{h^2}{2!} \frac{\partial^2 u}{\partial x^2} i+\theta_1,j \quad (2.1.5)$$

$0 \leq \theta_1 \leq 1$ and

$$u_{i,j+1} = u_{i,j} + h \frac{\partial u}{\partial y}_{i,j} + \frac{h^2}{2!} \frac{\partial^2 u}{\partial y^2} i,j+\theta_2 \quad (2.1.6)$$

$0 \leq \theta_2 \leq 1$.

The value of $\frac{\partial u}{\partial x}$ at the mesh point $i, j$ may be calculated by a number of different finite difference approximations, for example from (2.1.5), we have,

$$\frac{u_{i+1,j} - u_{i,j}}{h} = \frac{\partial u}{\partial x}_{i,j} + h \frac{\partial^2 u}{\partial x^2} x+\theta_1,j \quad (2.1.7)$$

or

$$\frac{\partial u}{\partial x}_{i,j} = \frac{u_{i+1,j} - u_{i,j}}{h} + O(h) , \text{ as } h \to 0 . \quad (2.1.8)$$

This is called forward difference approximation to $\frac{\partial u}{\partial x}_{i,j}$.

Similarly, for the backward difference approximation we have,

$$\frac{u_{i,j-1} - u_{i,j}}{h} = \frac{\partial u}{\partial x}_{i,j} + O(h) , \text{ as } h \to 0 . \quad (2.1.9)$$

If $u_{i+1,j}$ and $u_{i-1,j}$ are expanded about $u_{i,j}$ up to the third power of $h$, by (2.1.5), i.e.,

$$u_{i+1,j} = u_{i,j} + h \frac{\partial u}{\partial x}_{i,j} + \frac{h^2}{2!} \frac{\partial^2 u}{\partial x^2} i+\theta_1,j \pm \frac{h^3}{3!} \frac{\partial^3 u}{\partial x^3} i+\theta_1,j \quad (2.1.10)$$

$$u_{i,j-1} = u_{i,j} + h \frac{\partial u}{\partial y}_{i,j} + \frac{h^2}{2!} \frac{\partial^2 u}{\partial y^2} i,j-\theta_2$$
By subtraction we have,

\[ \frac{\partial^2 u}{\partial x^2}_{i,j} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + O(h^2), \]  

(2.1.11)

which is the central difference approximation.

From (2.1.10) we can approximate the second order derivative as,

\[ \frac{\partial^2 u}{\partial x^2}_{i,j} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + O(h^2). \]  

(2.1.12)

In a similar way, by (2.1.6), we have,

\[ \frac{\partial^2 u}{\partial y^2}_{i,j} = \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2} + O(h^2), \]  

(2.1.13)

and

\[ \frac{\partial^2 u}{\partial y^2}_{i,j} = \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2} + O(h^2). \]  

(2.1.14)

In the next chapters, we will omit the \(O(h^2)\) terms or any of further power which are defined as the local truncation error (L.T.E.).

2.2 BASIC MATRIX PROPERTIES AND CONCEPTS

The need for Matrix Algebra, when we use finite difference methods, comes from the replacement of the differential system by a matrix system.

In this section we describe some useful notations, definitions and properties.

Notations 2.2.1

- \(A\) square matrix of order \(n\)
- \(a_{ij}\) real number which is the element in the \(i\)th row and the \(j\)th column of the matrix
- \(A^{-1}\) inverse of \(A\)
- \(A^T\) transpose of \(A\)
- \(|A|\) determinant of \(A\) (\(\det(A)\))
identity (unit) matrix of order $n$

the null matrix

spectral radius (largest eigenvalue) of $A$

column vector with element $u_i$, $i = 1, \ldots, n$

row vector with element $u_j$, $j = 1, \ldots, n$

norm of $A$

**Definitions 2.2.1**

The matrix $A$ is said to be:

- **non-singular** if $|A| \neq 0$, otherwise **singular**
- **diagonal** if its only non-zero elements lie on the diagonal
- **identity** if it is diagonal with the non-zero elements being ones only
- **orthogonal** if $A^{-1} = A^T$
- **symmetric** if $A = A^T$, i.e. $a_{ij} = a_{ji}$, $i, j = 1, 2, \ldots, n$
  otherwise **nonsymmetric**
- **tridiagonal** if $a_{ij} = 0$ for $|i - j| > 1$
- **upper triangular** if $a_{ij} = 0$ for $i > j$
- **lower triangular** if $a_{ij} = 0$ for $j > i$
- **sparse** if most of its elements are zero and comparatively large

**Definition 2.2.2**

An $(n \times n)$ matrix $A$ is **diagonally dominant** if,

$$|a_{ii}| \geq \sum_{j=1, j \neq i}^{n} |a_{ij}|,$$

for all $1 \leq i \leq n$. 
and for at least one \( i, \)

\[ |a_{i,i}| > \sum_{j=1}^{n} |a_{ij}|. \]

**Definition 2.2.3**

A permutation matrix \( P = (p_{ij}) \) is a matrix which has elements of zeros and ones only with exactly one non-zero element in each row and each column. For example,

\[
P = \begin{bmatrix}
0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0
\end{bmatrix}
\]

is a permutation matrix of order 4.

For any permutation matrix \( P \) we have,

\[
P^T = P^T P = I \tag{2.2.2}
\]

\[
\overline{P} = P^{-1} \tag{2.2.3}
\]

**Definition 2.2.4**

An \((n \times n)\) matrix \( A \) \((n \geq 2)\) is **reducible** if there exists an \((n \times n)\) permutation matrix \( P \) such that,

\[
PAP^T = \begin{bmatrix}
A_{11} & A_{12} \\
0 & A_{22}
\end{bmatrix} \tag{2.2.4}
\]

where \( A_{11}, A_{12}, \) and \( A_{22} \) are \((r \times r)\), \((r) \times (n - r)\), and \((n - r) \times (n - r)\) matrices respectively and \(1 \leq r \leq n\). If no such matrix exists, then \( A \) is **irreducible**.
Definition 2.2.5

If an \((n \times n)\) matrix \(A\) is real and \(x\) is complex, then \(A\) is positive definite if
\[
(x, Ax) > 0, \quad \text{for all } x \neq 0.
\] (2.2.5)

If the inequality is not a strict one, then \(A\) is called semi-positive (non-negative) definite. In an analogous way, we can define negative definiteness.

Definition 2.2.6

A real matrix \(A = [a_{i,j}], i, j = 1, 2, \ldots, n,\) of order \(n\) is said to be \(L\)-matrix if
\[
a_{ii} > 0, \quad i = 1, 2, \ldots, n, \quad (2.2.6)
\]
and
\[
a_{i,j} \leq 0, \quad i \neq j, \quad \text{for all } i, j = 1, 2, \ldots, n. \quad (2.2.7)
\]
Furthermore, the matrix \(A\) is said to be \(M\)-matrix if (2.2.7) holds and \(A^{-1}\) exists such that \(A^{-1} \geq 0\), (i.e. \(A^{-1}\) has non-negative elements).

2.3 VECTORS AND MATRIX NORMS

The size or the magnitude of a vector or a matrix can be measured as a non-negative scalar. Such a measuring concept is called a norm and is denoted by \(\| \cdot \|\).

Definition 2.3.1

The norm of a vector \(x\), denoted by \(\| x \|\), is a real valued function satisfying the following axioms:

(i) \(\| x \| > 0\) for \(x \neq 0\) and \(\| x \| = 0\) if \(x = 0\).

(ii) \(\| ax \| = |a| \| x \|\), for any scalar \(a\).

(iii) \(\| x + y \| \leq \| x \| + \| y \|\), for all vectors \(x\) and \(y\) (called triangular inequality).
Definition 2.3.2

If $\mathbf{x}^T = (x_1, x_2, ..., x_n)$, the following scalars are defined as the $1, 2$ and $\infty$ norms of the vector $\mathbf{x}$:

$$\| \mathbf{x} \|_1 = \sum_{i=1}^{n} |x_i|$$

$$\| \mathbf{x} \|_2 = \left( \sum_{i=1}^{n} x_i^2 \right)^{1/2}, \text{ (Euclidean norm)}$$

$$\| \mathbf{x} \|_\infty = \max_i |x_i|, \text{ (maximum or uniform norm)}$$

In general $L_p$ norms are given by,

$$\| \mathbf{x} \|_p = \left( \sum_{i=1}^{n} |x_i|^p \right)^{1/p}, \quad 1 \leq p \leq \infty.$$

In a similar fashion, we can define a matrix norm.

Definition 2.3.3

The norm of an $(n \times n)$ matrix $\mathbf{A}$, written as $\| \mathbf{A} \|$ is a scalar satisfying the following axioms:

(i) $\| \mathbf{A} \| > 0$ and $\| \mathbf{A} \| = 0$ if and only if $\mathbf{A} = [0]$

(ii) $\| \alpha \mathbf{A} \| = |\alpha| \| \mathbf{A} \|$, for any scalar $\alpha$

(iii) $\| \mathbf{A} + \mathbf{B} \| \leq \| \mathbf{A} \| + \| \mathbf{B} \|$, for all $\mathbf{A}$ and $\mathbf{B}$

(iv) $\| \mathbf{AB} \| \leq \| \mathbf{A} \| \| \mathbf{B} \|$, for all $\mathbf{A}$ and $\mathbf{B}$

Definition 2.3.4

If $\mathbf{A} = [a_{ij}]$, $i,j = 1, 2, ..., n$, the following scalars are defined as the $1, 2$ and $\infty$ norms of $\mathbf{A}$,

(i) $\| \mathbf{A} \|_1 = \max_j \sum_{i=1}^{n} |a_{ij}|$

$\text{ (maximum absolute column sum) }$
(ii) $\|A\|_2 = \left\{ \rho(ATA) \right\}^{\frac{1}{2}}$ (spectral norm) \hspace{1cm} (2.3.6)

where $\rho(ATA)$ is the spectral radius of $ATA$, if $A$ is symmetric then $\|A\|_2 = \rho(A)$.

(iii) $\|A\|_\infty = \max_{i} \sum_{j=1}^{n} |a_{ij}|$, maximum absolute row sum. \hspace{1cm} (2.3.7)

Another norm is the Euclidean norm and is defined as,

$$\|A\|_E = \left\{\sum_{i,j} (a_{ij})^2\right\}^{\frac{1}{2}}. \hspace{1cm} (2.3.8)$$

**Definition 2.3.5**

A matrix norm is said to be *compatible* with a vector norm, if

$$\|Ax\| \leq \|A\| \cdot \|x\|. \hspace{1cm} (2.3.9)$$

**Definition 2.3.6**

A matrix norm is said to be *subordinate* to a vector norm if it can be constructed in the following form,

$$\|A\| = \max_{x \neq 0} \frac{\|Ax\|}{\|x\|}. \hspace{1cm} (2.3.10)$$

or equivalently,

$$\|A\| = \max_{\|x\|=1} \|Ax\|. \hspace{1cm} (2.3.11)$$

(for more details see S.D. Conte, 1981).
2.4 EIGENVALUES AND EIGENVECTORS

Definition 2.4.1

Let A be a square matrix of order n and \( \mathbf{x} \) is a non-zero vector such that

\[
A\mathbf{x} = \lambda \mathbf{x},
\]

where \( \lambda \) is a constant value, then \( \lambda \) is said to be an eigenvalue of A and \( \mathbf{x} \) is its corresponding eigenvector.

To find \( \lambda \) in (2.4.1), we rewrite it as

\[
(A - \lambda I)\mathbf{x} = \mathbf{0}.
\]

The eigenvector \( \mathbf{x} \neq 0 \) exists if and only if the matrix \( (A - \lambda I) \) is singular, i.e.

\[
\det(A - \lambda I) = 0.
\]

Equation (2.4.2) is called the characteristic equation of A, and the values of \( \lambda \)'s are the roots of the characteristic polynomial \( |A - \lambda I| = 0 \).

Definition 2.4.2

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

\[
\rho(A) = \max_{i} |\lambda_i|
\]

is the spectral radius of A.

Definition 2.4.3

Two matrices A and B are similar if there is a non-singular matrix P, such that

\[
B = P^{-1}AP.
\]

Then \( P^{-1}AP \) is a similarity transformation of A. Further more, if P is symmetric then \( P^{-1}AP \) is symmetric too.
Theorem 2.4.1

If two matrices $A$ and $B$ satisfy (2.4.3) then $A$ and $B$ have the same eigenvalues. If $x$ is any eigenvector of $A$ then $B$ has the corresponding eigenvector $y = p^{-1}x$.

Proof

(See Mitchell and Griffiths, 1980, p. 5).

Theorem 2.4.2

If an $(n \times n)$ matrix $A$ has distinct eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ then the corresponding eigenvectors $x_1, x_2, \ldots, x_n$ are linearly independent.

Corollary 2.4.1

If an $(n \times n)$ matrix $A$ has a distinct eigenvalue $\lambda_1, \lambda_2, \ldots, \lambda_n$, then $A$ is similar to the diagonal matrix $\text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$.

(For the proofs of Theorem 2.4.2 and Corollary 2.4.1, see R. Goult, 1974).

Theorem 2.4.3

The matrices $A$ and $A^T$ always have the same eigenvalues.

Proof

For any square matrix $A$, we have $|A| = |A^T|$. Hence,

$$|A - \lambda I| = |(A - \lambda I)^T| = |A^T - \lambda I|$$

that is $A$ and $A^T$ have the same characteristic equation, and therefore have the same eigenvalues.
Theorem 2.4.4: Gerschgorin's first theorem

The largest of the moduli of the eigenvalues of the square matrix A cannot exceed the largest sum of the moduli of the elements along any row or any column of A.

Theorem 2.4.5: Gerschgorin's circle theorem

Let $P_s$ be the sum of the moduli of the elements along the $s$th row excluding the diagonal elements $a_{ss}$. Then each eigenvalue of $A$ lies inside or on the boundary of at least one of the circles $|\lambda - a_{ss}| = P_s$.

(Proofs of Theorems 2.4.4 and 2.4.5 can be found in G. Smith, 1978, pp. 87-89).

From Gerschgorin's theorems we obtain,

$$\rho(A) \leq \min \left\{ \max_i \left( \sum_j a_{ij} \right), \max_j \left( \sum_i a_{ij} \right) \right\}$$

and more generally it can be proved that,

$$\rho(A) \leq \|A\|. \quad (2.4.4)$$

The following theorems (for proofs, see R. Goult, 1974), constitute the main properties of the eigenvalues of symmetric matrices.

(i) The eigenvalues of a real symmetric matrix are all real.

(ii) Orthogonality property: if $x_1$ and $x_2$ are eigenvectors of a real symmetric matrix $A$, corresponding to distinct eigenvalues $\lambda_1$ and $\lambda_2$, then $x_1^T x_2 = x_2^T x_1 = 0$.

(iii) If $A$ is a symmetric, diagonally dominant matrix with positive diagonal elements, then it is positive definite.

(iv) A real symmetric matrix is positive definite if its quadratic form is positive for all $x \neq 0$, i.e., $x^T Ax > 0$. 
(v) Let $\lambda$ be an eigenvalue of $A$ with eigenvector $x$. Then,
(a) $\alpha \lambda$ is an eigenvalue of $\alpha A$ with eigenvector $x$
(b) $(\lambda - \mu)$ is an eigenvalue of $(A - \mu I)$ with eigenvector $x$
(c) If $A$ is non-singular, then $\lambda \neq 0$ and $\lambda^{-1}$ is an eigenvalue of $A^{-1}$ with eigenvector $x$.

2.5 PROPERTY A AND CONSISTENTLY ORDERED MATRICES

Definition 2.5.1

An $(n \times n)$ matrix $A = [a_{ij}]$ is said to have Property A (Young, 1971) if there exist two disjoint subsets $S_1$ and $S_2$ of $W = \{1, 2, \ldots, n\}$ such that $S_1 + S_2 = W$ and such that if $i \neq j$ and if either $a_{ij} \neq 0$ or $a_{ji} \neq 0$, then $i \in S_1$ and $j \in S_2$ or else $i \in S_2$ and $j \in S_1$.

If $S_1$ or $S_2$ is empty then $A$ is diagonal. If neither $S_1$ nor $S_2$ is empty then there exists a permutation matrix $P$ such that $PAP^T = T$ (tridiagonal representation of $A$) has the form,

$$PAP^T = \begin{bmatrix} D_1 & F \\ E & D_2 \end{bmatrix},$$

(2.5.1)

where $D_1$ and $D_2$ are square diagonal matrices and $F$ and $E$ are rectangular matrices.

Ordinarily, if $A$ has property A then there are many suitable matrices $P$, each giving rise to a different tridiagonal representation of $A$. Furthermore, if $A$ has property A then $PAP^T$ has property A.

Definition 2.5.2

Given a matrix $A = [a_{ij}]$, the integers $i$ and $j$ are associated with respect to $A$ if $a_{ij} \neq 0$ or $a_{ij} \neq 0$. 
Definition 2.5.3

The matrix $A$ of order $n$ is **consistently ordered** if for some $t$ there exists disjoint subsets $S_1, S_2, \ldots, S_t$ of $W = \{1, 2, \ldots, n\}$ such that $\bigcup_{k=1}^t S_k = W$ and such that if $i$ and $j$ are associated then $j \in S_{k+1}$ if $j > i$ and $j \in S_{k-1}$ if $j < i$, where $S_k$ is the subset containing $i$.

Definition 2.5.4

An $(n \times n)$ real matrix $A$ is **weakly cyclic of index $p$, $p \geq 2$** (or weakly $p$-cyclic) if there exists a permutation matrix $P$ of order $n$ such that $PA^T P^T$ has the form

$$
PA^T P^T = \begin{bmatrix}
0 & 0 & 0 & \ldots & 0 & A_{1,1} \\
A_{2,1} & 0 & 0 & 0 & 0 \\
0 & A_{3,2} & 0 & 0 & 0 \\
\vdots & \ddots & \ddots & \ddots & \ddots \\
0 & 0 & 0 & \ldots & A_{p-1,p} & 0
\end{bmatrix}, \quad p \geq 2,
$$

where $A_{i,j}$ is an $n_i \times n_j$ submatrix and $\sum_{i=1}^p n_i = n$.

Definition 2.5.5

A matrix is **two-cyclic** if by a suitable permutation of its rows and corresponding columns, it can be written in the form,

$$
\begin{bmatrix}
I_1 & P \\
G & I_2
\end{bmatrix}
$$

where $I_1$ and $I_2$ are square unit matrices, and $F$ and $G$ are arbitrary rectangular matrices. Furthermore a matrix is weakly two-cyclic if it can be written in the form,
where $O_1$ and $O_2$ are square null matrices.

**Theorem 2.5.1**

If $A$ is a matrix with a tridiagonal representation (2.5.1), then $A$ is consistently ordered.

**Proof**

(See Young, 1971).

**Definition 2.5.6**

If a matrix $(I - L - U)$ is two-cyclic, then it is consistently ordered if all the eigenvalues of the matrix

$$\alpha L + \frac{1}{\alpha} U, \ (\alpha \neq 0),$$

are independent of $\alpha$.

### 2.6 LU Decomposition

When the finite difference approximation to the p.d.e. is applied, a matrix of coefficients, namely $A$ is obtained. One of the solution methods which can be used, for example, is the elimination method. A modification to this method is the so-called LU decomposition method by which the product of two matrices $L$ and $U$, where $L$ is a lower triangular and $U$ is an upper triangular matrix with 1's on its diagonal. We get the rules for such LU decomposition from the relationship that $LU = A$. For illustration, we use the case of a $4 \times 4$ matrix:
multiplying the rows of \(L\) by the first column of \(U\), we obtain \(l_{11} = a_{11},\)
\(l_{21} = a_{21},\) \(l_{31} = a_{31},\) \(l_{41} = a_{41};\) we now multiply the first row of \(L\) by
the columns of \(U\) to obtain \(u_{1i}, i = 2, 3, 4:\)
\[
l_{11}u_{12} = a_{12}, \quad l_{11}u_{13} = a_{13}, \quad l_{11}u_{14} = a_{14}
\]
or
\[
u_{12} = \frac{a_{12}}{l_{11}}, \quad u_{13} = \frac{a_{13}}{l_{11}}, \quad u_{14} = \frac{a_{14}}{l_{11}}
\]
Thus, the first column of \(L\) and the first row of \(U\) have been obtained.

We now proceed in a similar fashion to perform the same strategy on the
following rows. We notice that we alternate between obtaining a column
of \(L\) and a row of \(U\).

The general formula for deriving elements of \(L\) and \(U\) corresponding
to the \((n \times n)\) matrix can be written as,
\[
l_{ij} = a_{ij} - \sum_{k=1}^{j-1} l_{ik}u_{kj}, \quad j < i, \quad i = 1, 2, \ldots, n,
\]
and for \(j = 1\) it reduces to \(l_{i1} = a_{i1}\)
\[
u_{ij} = \frac{a_{ij} - \sum_{k=1}^{i-1} l_{ik}u_{kj}}{l_{ii}}, \quad i < j, \quad j = 2, 2, \ldots, n
\]
and for \(i = 1,\) it reduces to,
The method is a popular direct method because of the saving of the storage space since there is no need to store the zeros in either of the L or U forms. Also, the 1's on the diagonal of U can be omitted.

To solve the linear system $LUx = b$, we follow the following two step procedure:

(i) Step 1: the forward substitution

We have, $LUx = b$

and by introducing the auxiliary vector $y$ such that $Ux = y$

we have, $Ly = b$.

We obtain the elements of $y$ from,

$$y_1 = \frac{b_1}{\ell_{11}}$$

$$b_i = \ell_{i1} y_1 + \sum_{k=1}^{i-1} \ell_{ik} y_k$$

and

$$y_i = \frac{b_i - \sum_{k=1}^{i-1} \ell_{ik} y_k}{\ell_{ii}}$$ , $i = 2, 3, \ldots, n$

(ii) Step 2: the backward substitution

We solve $Ux = y$, where we obtain the elements of $x$, the required solution by the equation,

$$x_n = y_n$$

$$x_i = y_i - \sum_{k=i+1}^{n} u_{ik} x_k$$ , $i = n - 1, n - 2, \ldots, 2, 1$. 
We summarize the LU decomposition method by the following steps:

(i) Compute $L$ and $U$ such that $A = LU$.
(ii) Solve $Ly = b$.
(iii) Solve $Ux = y$.

2.7 THE POWER METHOD

In the following chapters, we will require the largest and the smallest eigenvalues of the iteration matrix to determine the convergence rate of an iterative method. The power method, in its basic form, yields the eigenvalues of the largest modulus together with its corresponding eigenvector.

Theory of the Power Method

Let us assume that $A$ is a real $(n \times n)$ matrix, with $n$ real distinct eigenvalues $\lambda_i$, $i = 1, 2, \ldots, n$. $A$ thus has a full set of linearly independent eigenvectors $x_i$, $i = 1, 2, \ldots, n$, such that their linear combination can be written as,

$$y^{(0)} = \sum_{i=1}^{n} \alpha_i x_i,$$

(2.7.1)

where $y^{(0)}$ is an arbitrary vector and $\alpha_i$, $i = 1, 2, \ldots, n$ are constant, not all zero. We also assume that the eigenvalues are numbered so that,

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \ldots \geq |\lambda_n|$$

(2.7.2)

Now, we form the sequence,
\[
\gamma^{(1)} = A\gamma^{(0)} \\
\gamma^{(2)} = A\gamma^{(1)} = A^2\gamma^{(0)} \\
\vdots \\
\gamma^{(k)} = A\gamma^{(k-1)} = A^k\gamma^{(0)}
\]  
\tag{2.7.3}

Also, for any \(i, 1 < i < n\), we have,

\[
A\mathbf{x}_i = \lambda_i\mathbf{x}_i \\
A^2\mathbf{x}_i = \lambda_i A\mathbf{x}_i = \lambda_i^2\mathbf{x}_i \\
\vdots \\
A^k\mathbf{x}_i = \lambda_i^{k-1} A\mathbf{x}_i = \lambda_i^k\mathbf{x}_i
\]  
\tag{2.7.4}

Now, by (2.7.1), (2.7.3) and (2.7.4) we have,

\[
\gamma^{(k)} = A^k\gamma^{(0)} = \sum_{i=1}^{n} \alpha_i A^k\mathbf{x}_i = \sum_{i=1}^{n} \alpha_i \lambda_i^k\mathbf{x}_i ,
\]  
\tag{2.7.5}

or,

\[
\gamma^{(k)} = \lambda_1^k \left\{ \alpha_1\mathbf{x}_1 + \sum_{i=2}^{n} \alpha_i \left( \frac{\lambda_i}{\lambda_1} \right)^k \mathbf{x}_i \right\} .
\]  
\tag{2.7.6}

Since \(\left| \frac{\lambda_1}{\lambda_i} \right| < 1\), for \(i = 2, \ldots, n\), then as \(k \to \infty\) the terms \(\left\{ \frac{\lambda_i}{\lambda_1} \right\}^k \to 0\), i.e.,

\[
\gamma^{(k)} \approx \lambda_1^k \alpha_1\mathbf{x}_1 .
\]  
\tag{2.7.7}

In order to estimate \(\lambda_1\), we obtain from (2.7.7), by replacing \(k\) by \(k + 1\),

\[
\gamma^{(k+1)} \approx \lambda_1^{k+1} \alpha_1\mathbf{x}_1 ,
\]

or

\[
\gamma^{(k+1)} \approx \lambda_1\gamma^{(k)} .
\]

This result suggests a number of ways to approximate \(\lambda_1\). For example

\[
(\gamma^{(k)})^T (\gamma^{(k+1)}) = \lambda_1 (\gamma^{(k)})^T \gamma^k
\]
and thus
\[ \lambda_1 = \frac{\gamma^{(k)}_T (k+1)}{\gamma^{(k)}_T (k)} \]

The disadvantage is that if \(|\lambda_1|\) is large then after a few iterations the elements of \(\gamma^{(k)}\) will be very large, and overflow may soon occur. To avoid that we normalize \(\gamma^{(k)}\) at each iteration.

**The Algorithm of the Method**

The Power method proceeds for \(k = 1, 2, 3, \ldots\)

1. Let \(\gamma^{(k)} = Ax^{(k-1)}\)
2. Set \(\beta_k = \gamma^{(k)}_T x^{(k-1)}\)
3. Let \(n_k = (\gamma^{(k)}_T \gamma^{(k)} / \beta_k)^{1/2}\)
4. Set \(x^{(k)} = \gamma^{(k)} / n_k\)

and return to step 1 until the stopping criterion is achieved.

Note that \(\beta_k\) is the approximation to \(\lambda_1\) and that each \(x^{(k)}\) is an approximate eigenvector, as \(k \to \infty\).

**Remarks**

1. Convergence will be slow for matrices having two large eigenvalues of nearly equal modulus.
2. There is no need to compute \(A^k\), nor to know the \(a_i\)'s and the \(x_i\)'s in (2.7.1).
3. If \(A\) does not have \(n\) linearly independent eigenvectors, the Power method still converges provided \(A\) has a single dominant eigenvalue \(\lambda_1\).
4. To find the approximate eigenvalues other than the dominant one we can use the concept of the inverse Power method.

[Johnson and Reiss, 1982.]
2.8 CONVERGENCE OF SEQUENCES OF MATRICES

A sequence of matrices $A^{(1)}, A^{(2)}, A^{(3)}, \ldots, A^{(k)}$ is convergent to a matrix $A$ (say) if,

$$\lim_{k \to \infty} \| A - A^{(k)} \| = 0,$$

where $\| A \|$ is any norm of the matrix $A$.

**Definition 2.8.1**

The matrix $A$ converges to zero if the sequence of matrices $A, A^2, A^3, \ldots$ converges to the null matrix $0$.

**Theorem 2.8.1**

For the matrix $A$,

$$\lim_{k \to \infty} A^k = 0, \text{ if } \| A \| < 1.$$  \hspace{1cm} (2.8.1)

**Proof**

$$\| A^k \| = \| A \| A^{k-1} \| \leq \| A \| \cdot \| A \|^{k-1} \|
\leq \| A \|^2 \| A \|^{k-2} \|
\leq \| A \|^k$$

and so the result follows.

**Theorem 2.8.2**

If $A$ is an $m \times m$ matrix, then $A$ is convergent if and only if $\rho(A) < 1$, where $\rho(A)$ is the spectral radius of $A$.

**Proof**

Let us assume that all the eigenvectors of $A$ are linearly independent, then they can be assembled as a non-singular matrix $Q$, such that,
Q^{-1} A Q = D = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_m), \quad (2.8.2)

where $\lambda_i$, $i = 1, 2, \ldots, m$ are the eigenvalues of $A$.

Then

$$A = QDQ^{-1}, \quad (2.8.3)$$

whence,

$$A^n = QDQ^{-1} QDQ^{-1} \ldots QDQ^{-1} = QD^n Q^{-1}. \quad (2.8.4)$$

Now,

$$D^n = \text{diag}(\lambda_1^n, \lambda_2^n, \ldots, \lambda_m^n). \quad (2.8.5)$$

Since $\rho(A) < 1$, then $D^n \to 0$ and whence, $A^n \to 0$ as $n \to \infty$ which is the result. (See Varga (1962), p. 13, for a different proof).

2.9 CONDITION NUMBER AND ERROR ESTIMATES

Suppose we have the linear system

$$Ax = b. \quad (2.9.1)$$

Let $x'$ be an approximation to the exact solution $x$ and the residual vector,

$$r = b - Ax', \quad (2.9.2)$$

is such that $\|r\|$ is small, then $\|x - x'\|$ would be small too.

Theorem 2.9.1

If $x'$ is an approximation to the solution $x$ of $Ax = b$ and $A$ is a non-singular matrix. Then for any compatible matrix and vector norms,

$$\|x - x'\| \leq \|r\| \|A^{-1}\| \quad (2.9.3)$$

and

$$\frac{1}{\|A\| \|A^{-1}\| \|b\|} \leq \frac{\|x - x'\|}{\|x\|} \leq \|A\| \|A^{-1}\| \frac{\|r\|}{\|b\|} \quad (2.9.4)$$
Proof

Since \( r = b - Ax' = A(x - x') \) and is non-singular, \( x - x' = A^{-1}r \), then by (2.3.9),

\[
\| x - x' \| = \| A^{-1}r \| < \| A^{-1} \| \| r \| . \tag{2.9.5}
\]

Moreover, since \( b = Ax \), \( \| b \| < \| A \| \| x \| \); so

\[
\frac{\| x - x' \|}{\| x \|} \leq \frac{\| A \| A^{-1} \|}{\| x \|} \| r \|, \tag{2.9.6}
\]

which establishes the right-hand side of (2.9.4). Now,

\[
\| r \| = \| A(x - x') \| < \| A \| \| x - x' \|
\]

and since \( \| A \| > 0 \), then,

\[
\| x - x' \| > \frac{\| r \|}{\| A \|}. \tag{2.9.7}
\]

Also, since \( x = A^{-1}b \), so \( \| x \| < \| A^{-1} \| \| b \| \), or

\[
\frac{1}{\| x \|} > \frac{1}{\| A^{-1} \| \| b \|}. \tag{2.9.8}
\]

Now, by combining (2.9.7) and (2.9.8), we have,

\[
\frac{\| x - x' \|}{\| x \|} > \frac{1}{\| A \| \| A^{-1} \| \| b \|} \| r \| \tag{2.9.9}
\]

establishing the left-hand side of (2.9.4).

From (2.9.4) we notice that the relative error \( \| x - x' \|/\| x \| \) is bounded by the product of \( \| A \| \| A^{-1} \| \) with the relative residual for this approximation \( (x') \), \( \| r \|/\| b \| \).
Definition 2.9.1

The condition number $k(A)$ of the non-singular matrix $A$ relative to a norm $\| \cdot \|$ is defined to be,

$$k(A) = \| A \| \| A^{-1} \|,$$  \hspace{1cm} (2.9.10)

with this notation, (2.9.3) and (2.9.4) become,

$$\| x - x' \| \leq k(A) \frac{\| r \|}{\| A \|}$$  \hspace{1cm} (2.9.11)

and,

$$\frac{1}{k(A)} \frac{\| \xi \|}{\| b \|} \leq \frac{\| x - x' \|}{\| x \|} \leq k(A) \frac{\| \xi \|}{\| b \|}.$$  \hspace{1cm} (2.9.12)

Since for any non-singular matrix $A$,

$$1 = \| I \| = \| A A^{-1} \| \leq \| A \| \| A^{-1} \| = k(A).$$

The matrix $A$ is called a well-conditioned matrix if $k(A)$ is close to one, and ill-conditioned if $k(A)$ is significantly greater than one. The entries of $A$ and $b$ might be changed slightly or perturbed by an amount $\delta A$ and $\delta b$ resulting in the linear system,

$$(A + \delta A)x = b + \delta b.$$  \hspace{1cm} (2.9.13)

The following theorem relates the perturbations of linear system to the condition number of the matrix

Theorem 2.9.2

Suppose $A$ is non-singular and,

$$\| \delta A \| < \frac{1}{\| A^{-1} \|},$$
then the solution $x'$ to $(A + \delta A)x' = b + \delta b$ approximates the solution $x$ of $Ax = b$ with error estimate,

$$\frac{\|x - x'\|}{\|x\|} \leq \frac{k(A)}{1 - k(A)\|\delta A\|/\|A\|\|b\|/\|A\|}$$

The inequality (2.9.14) shows that if the matrix $A$ is well-conditioned, i.e., $k(A)$ is not too large, then the perturbations in $A$ and $b$ produce correspondingly small changes in the solution $x$. But if $A$ is ill-conditioned, then a perturbation in $A$ and $b$ produce large changes in $x$. (See R.L. Burden, et al., 1978).

2.10 VARIATIONAL PRINCIPLES AND CONJUGATE GRADIENTS

A. VARIATIONAL PRINCIPLES

The solutions of many boundary value problems of Mathematical Physics minimize appropriate functionals. In particular, every configuration of static equilibrium in classical (Lagrangian) mechanics minimizes a suitable energy function. The approximate solution of the boundary value problem is then the approximate solution of the corresponding variational problem.

The origin of variational methods came from the formulation of the so-called Dirichlet principle: consider the Dirichlet problem for the Laplace equation,

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \text{ in } R$$

and $u = g$ on $C$,

where $R$ is any bounded 2-dimensional region and $C$ is the continuous boundary.
Let $M$ be the set of functions which are continuous and have continuous first partial derivatives in $\mathbb{R}^+ \times \mathbb{R}$ and satisfy the condition (2.10.2). The Dirichlet principle states that the function which minimizes the so-called Dirichlet integral,

$$I(u) = \iint_R \left[ \left( \frac{\partial u}{\partial x} \right)^2 + \left( \frac{\partial u}{\partial y} \right)^2 \right] \, dx \, dy,$$

(2.10.3)

among all the functions from $M$, is harmonic in $\mathbb{R}$ and thus constitutes the solution of the problem (2.10.1), (2.10.2) (see Rektorys, 1975).

**Definition 2.10.1**

A functional is an operator that converts functions into scalars. The typical form of a functional is,

$$J(u) = \int_a^b \phi(u(x), \frac{\partial u(x)}{\partial x}, x) \, dx,$$

(2.10.4)

where $u(x)$ here is an independent variable ($x \in [a, b]$) and $J$ is the dependent variable (a scalar).

**Example 2.10.1**

Consider the path joining two points $A$ and $B$ with the coordinates $(a, \alpha)$ and $(b, \beta)$ respectively, (Figure 2.10.1).
The length of the minimum path connecting the points A and B is given by:

\[
\text{length} = \int_a^b \left[1 + (u')^2\right]^{\frac{1}{2}} \, dx,
\]  

(2.10.5)

where \( u' = \frac{du(x)}{dx} \).

Now, consider the functional,

\[
J(u) = \int_0^\lambda \phi(u, u', x) \, dx,
\]  

(2.10.6)

where \( u = u(x) \) and \( 0 < x < \lambda \).

We define the difference,

\[
\delta J = J(u + \delta u) - J(u)
\]  

(2.10.7)

corresponding to a small variation \( \delta u \) in the argument \( u \), (Figure 2.10.2),

where \( \delta u \) is a continuous function with a continuous derivative which vanishes at the ends of the range, i.e. \( \delta u(0) = \delta u(\lambda) = 0 \). \( \delta J \) is called the variation of \( J \). Now by using Taylor's Theorem to expand \( J(u + \delta u) \), we can have the following final form of \( \delta J \),

\[
\delta J = \int_0^\lambda \frac{\partial \phi}{\partial u} \delta u \, dx + \frac{1}{2} \int_0^\lambda \frac{\partial^2 \phi}{\partial u^2} (\delta u)^2 \, dx + \cdots
\]
\[ \delta J = \int_0^L \left[ \frac{\partial \phi}{\partial u} - \frac{d}{dx} \left( \frac{\partial \phi}{\partial u'} \right) \right] \delta u \, dx \]  

(2.10.8)

**Definition 2.10.2**

If \( J(u) \) (as in (2.10.4)) reaches a maximum or minimum, then \( u(x) \) is called an **extremal function** or **extremal curve** of the functional \( J(u) \).

A necessary condition for \( u(x) \) to be extremal is that \( \delta J = 0 \) for any small, permissible displacement \( \delta u(x) \). This involves,

\[ \frac{\partial \phi}{\partial u} - \frac{d}{dx} \left( \frac{\partial \phi}{\partial u'} \right) = 0 , \]  

everywhere in \( x \in [0, L] \).

Equation (2.10.9) is called the Euler-Lagrange equation for the given extremal problem.

**B. CONJUGATE GRADIENTS**

The conjugate gradient method is a conjugate direction method, using conjugate successive gradients obtained as the method progresses. The general conjugate direction method has the form,

\[ u^{(k+1)} = u^{(k)} + \alpha_k p^{(k)} , \quad k \geq 0 , \]  

(2.10.10)

where \( p^{(k)} \) is a conjugate direction vector and \( \alpha_k \) is a scalar. (See later).

**Definition 2.10.3**

Given a symmetric matrix \( A \), two vectors \( p_1 \) and \( p_2 \) are said to be \( A \)-**orthogonal**, or **conjugate** with respect to \( A \) if \( (p_1, Ap_2) = 0 \). (The phrase 'with respect to \( A \)' will be omitted where there is no ambiguity).
Theorem 2.10.1

If the vectors \( \overline{P}(j) \) are mutually conjugate, then they are linearly independent.

Proof

Suppose that

\[
\sum_j \alpha_j \overline{P}(j) = 0
\]

for some scalar \( j \). If \( k \) is any one of the values of \( j \), then,

\[
\left[ \overline{P}(k), A \sum_j \alpha_j \overline{P}(j) \right] = 0,
\]

i.e.,

\[
\alpha_k \left[ \overline{P}(k), A \overline{P}(k) \right] = 0.
\]

Now \( A \) is positive definite and \( \overline{P}(k) \neq 0 \), hence \( \alpha_k = 0 \) and the proof follows.

Theorem 2.10.2

Let \( \left\{ \overline{P}(j) \right\} \), \( j = 0, 1, \ldots, n - 1 \), be a set of non-zero conjugate vectors. For any \( u(0) \in \mathbb{R}^n \) the sequence \( u(k) \) generated according to

\[
u(k+1) = u(k) + \alpha_k \overline{P}(k), \quad k \geq 0,
\]

with,

\[
\alpha_k = \frac{\left( \overline{r}(k), \overline{P}(k) \right)}{\left( \overline{P}(k), A \overline{P}(k) \right)}
\]

and

\[
\overline{r}(k) = b - Au(k),
\]

converges to the unique solution \( u^* \) of \( Au = b \) after \( n \) steps, i.e., \( u^{(n)} = u^* \).

Proof


More details and applications are presented in Chapter 7.
Chapter Three

ALGORITHMS FOR SOLVING UNIVARIATE NONLINEAR EQUATIONS

3.1 Introduction

3.2 A Survey of Iterative Methods
   3.2.1 Bisection Method
   3.2.2 Regula Falsi Method
   3.2.3 Secant Method
   3.2.4 Newton-Raphson Method

3.3 Convergence of Fixed Point Methods

3.4 Two-Point Newton Method
   3.4.1 Basic Two Point Newton Algorithm
   3.4.2 Modified Two Point Algorithm

3.5 Experimental Results
3.1 INTRODUCTION

Consider the differentiable univariate nonlinear equation,

\[ f(x) = 0 \]  \hspace{1cm} (3.1.1)

we want to find the roots of the above equation (i.e. zeros of the function). This reduces to the problem of finding a point \( x^* \) in a bounded interval \([a, b]\), such that

\[ f(x^*) = 0, \]

where \( f(a)f(b) < 0 \) and \( |a - b| < \varepsilon \) \hspace{1cm} (3.1.2)

\( \varepsilon \) being some small tolerance. If the function is continuous, then at least one root must lie in that interval. If the function is discontinuous, but bounded, then instead of a root there might be a step discontinuity which crosses zero.

There are many effective iterative methods to find the roots of an equation. Each of them have differing qualities (in terms of number of iterations and computational work per iteration) depending on the behaviour of the equation under investigation. We will be dealing with the Newton-Raphson and the secant methods. We will also introduce a '2-point' Newton method which requires two initial values to start the process.

In this chapter we construct an algorithm which combines the three methods mentioned above, where at each iteration we choose the method which gives the best approximation to the root by applying particular tests and without increasing the computational work (in terms of number of function evaluations).
3.2 A SURVEY OF ITERATIVE METHODS

Consider the function $f(x)$, differentiable and defined in the interval $[a, b]$, such that $f(a)$ and $f(b)$ of opposite sign. $x^*$ is the root of $f(x)$ such that $a < x^* < b$ and let $\epsilon$ be a small tolerance.

To find the root of $f(x)$, there are two classes of methods namely:

1. Bracketing methods,
2. Fixed-point methods.

The bracketing methods always need an interval contains (brackets) a root and at each iteration, two points are used to obtain a new approximation of the root and then the interval must be updated to a new one which again brackets the root. Such methods are the bisection and the regula falsi methods. These methods always converge, but slowly, when the function is continuous on $[a, b]$. The fixed point methods use an auxiliary function $x = g(x)$ such that $x^* = g(x^*)$ whenever $f(x^*) = 0$ where $a \leq x^* \leq b$. The construction of $g(x)$ is not unique. These methods such as the Newton's and the Secant methods, usually converge faster than the bracketing methods.

3.2.1 The Bisection Method

This method is very effective although the idea is simple. Over the interval $[a, b]$, the function passes through zero because it has opposite signs at $a$ and $b$. Evaluate the function at the interval's midpoint (i.e. $\frac{a+b}{2}$ ) and consider its sign. We then replace whichever limit ($a$ or $b$) has the same sign, and repeat the process. After each iteration the interval containing the zero decreases by a factor $\frac{1}{2}$. If after $n$ iterations the zero will be
within an interval of length $\varepsilon_n$, then after the next iteration
it will be bracketed within an interval of size

$$\varepsilon_{n+1} = \frac{\varepsilon_n}{2}.$$  \hfill (3.2.1)

Thus we can know in advance the number of iterations required to
achieve convergence, which is

$$n = \log_2 \frac{\varepsilon_0}{\varepsilon},$$  \hfill (3.2.2)

where $\varepsilon_0$ is the initial bracketing interval's size, and $\varepsilon$ is the
required tolerance. If the interval contains one or more zeros,
bisection method will find one of them. If the interval contains
no zeros but a singularity, it may converge on the singularity.

**Bisection Algorithm**

Given a function $f(x)$ defined on the interval $[a, b]$ and
such that $f(a)f(b) < 0$.

For $n = 0, 1, \ldots$, do

set $m = \frac{a_n + b_n}{2}$.

If $f(m) \leq \varepsilon$, then $x^* = m$ and stop.

If $f(a_n)f(m) < 0$ set $a_{n+1} = a_n, b_{n+1} = m$.

Otherwise, set $a_{n+1} = m, b_{n+1} = b_n$.

3.2.2 **Regula Falsi (False Position) Method**

In this method, we take the approximation as the point of
intersection of the straight line through the points $(a, f(a))$ and
$(b, f(b))$ and the $x$-axis as illustrated in figure (3.2.1), which
can be expressed by the formula,

\[ m = \frac{f(b)a - f(a)b}{f(b) - f(a)} \]  \hspace{1cm} (3.2.3) \]

where the interval \([a, b]\) brackets the root. The method will be more efficient if the function \(f(x)\) is approximately linear in the local region of interest, i.e. \([a, b]\).

![Figure (3.2.1), Regula Falsi Method](image)

The regula falsi method retains the prior point for which the function value has opposite sign from that one at current estimate, so that the two points (the new interval) continue to bracket the root.

**Regula Falsi Algorithm**

For \(n = 0, 1, 2, \ldots\), do

Compute \( m = \frac{f(b)_{n}a_{n} - f(a)_{n}b_{n}}{f(b)_{n} - f(a)_{n}} \)

If \( |f(m)| \leq \epsilon \), then \( x^* = m \) and stop

If \( f(a)_{n}f(m) < 0 \), set \( a_{n+1} = a_{n}, \; b_{n+1} = m \)

Otherwise, set \( a_{n+1} = m, \; b_{n+1} = b_{n} \).
3.2.3 **Secant Method**

This method is a modification to the regula falsi method, where it drops the bracketing process of the root. It retains the most recent prior estimates (see Figure 3.2.2).

\[ f(x) \]

Figure (3.2.2), Secant method

**Secant Algorithm**

Given a function \( f(x) \) and two points \( x^{(-1)}, x^{(0)} \).

For \( k = 0,1,2,\ldots \), do

\[
x^{(k+1)} = \frac{x^{(k)}x^{(k-1)} - f(x^{(k-1)})x^{(k)}}{f(x^{(k)}) - f(x^{(k-1)})}.
\]  

(3.2.4)

If \( |f(x^{(k+1)})| \leq \epsilon \), \( x^* = x^{(k+1)} \) and stop.

3.2.4 **Newton-Raphson Method**

The basic Newton-Raphson method, also called Newton method, for finding a zero of \( f(x) \) arises from the first order Taylor expansion of \( f(x) \) in the neighbourhood of a point, i.e.
\[ f(x+h) = f(x) + h \frac{f'(x)}{2} + h^2 \frac{f''(x)}{2} + \theta h \]  

(3.2.5)

where \( 0 < \theta < 1 \) and \( h \) is small value. If \( x+h \) is an approximation to the zero and \( h \) is small enough, then \( f(x+h) \approx 0 \) implies

\[ h = -\frac{f(x)}{f'(x)} \]  

(3.2.6)

By adding the subscripts \( k \) and \( k+1 \) to \( x \) and setting \( h = x^{(k+1)} - x^{(k)} \), then (3.2.6) becomes

\[ x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})} \]  

(3.2.7)

which is the Newton iterative method. Geometrically, \( x_{n+1} \) is constructed from the intersection of the tangent to \( f(x) \) at the point \( x^{(k)} \) with the x-axis (see Figure (3.2.3)). If the initial guess is near a local extremum, so that the first derivative \( f'(x) \) nearly vanishes, the Newton method can send its solution far away from the actual zero.

Variations of this method have the form

\[ x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{S^{(k)}} \]  

(3.2.8)

where \( S^{(k)} \) is an approximation of \( f'(x^{(k)}) \).

By setting

\[ S^{(k)} = \frac{f(x^{(k)}) - f(x^{(k-1)})}{x^{(k)} - x^{(k-1)}} \]

equation (3.2.8) reduces to the secant method.
Newton-Raphson Algorithm

Given a function \( f(x) \) and initial point \( x^{(0)} \).

For \( k = 0,1,2,\ldots \), do

Compute \( x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})} \).

If \( |f(x^{(k+1)})| \leq \varepsilon \), then \( x^* = x^{(k+1)} \) and stop.

3.3 CONVERGENCE OF FIXED-POINT ITERATION

Consider the function \( f(x) \) which is continuous and differentiable on the interval \( I = [a,b] \) where there exists a fixed point \( x^* \in I \) such that \( f(x^*) = 0 \). \( g(x) \) is the fixed-point function derived from \( f(x) \). It is also continuous and differentiable on \( I \) and the function \( g(x) \) maps \( I \) into itself.

Theorem 3.3.1

Let \( g(x) \) be an iteration function which is continuous and differentiable on \( I = [a,b] \) and there exists a positive constant
M < 1 such that

\[ |g'(x)| \leq M, \quad \text{for all } x \in I. \]

Then \( g(x) \) has one fixed point \( x^* \) in \( I \), and with starting point \( x(0) \in I \), the sequence \( x(1), x(2), \ldots \) generated by \( g(x) \) converges to \( x^* \).

**Proof:**

Let the error in the \( k \)th iteration be

\[ e(k) = x^* - x(k), \quad k = 0, 1, 2, \ldots \quad (3.3.1) \]

Since, \( x^* = g(x^*) \) and \( x(k) = g(x(k-1)) \), then we have,

\[ e(k) = g(x^*) - g(x(k-1)) \]

\[ = g'(\eta(k)) e(k-1), \quad (3.3.2) \]

for some \( \eta(k) \) between \( x^* \) and \( x(k-1) \), obtained by the mean-value theorem for derivatives, i.e.

\[ \frac{[g(x^*) - g(x(k-1))]}{(x^* - x(k-1))} = g'(\eta(k)). \]

Now by the assumption that there exists a positive constant \( M < 1 \) such that

\[ |g'(x)| \leq M, \quad \text{for all } x \in I. \quad (3.3.3) \]

Then from (3.3.2), we have

\[ |e(k)| \leq M|e(k-1)|. \]

It follows using (3.3.1) that

\[ |e(k)| \leq M^k |e(0)| \quad (3.3.4) \]

\[ \lim_{k \to \infty} |e(k)| \leq \lim_{k \to \infty} M^k |e(0)|, \]
and since $0 < M < 1$ then $\lim_{k \to \infty} M^k = 0$, so

$$\lim_{k \to \infty} |e^{(k)}| = 0,$$

(3.3.5)

regardless of the initial error $e^{(0)}$.

The proof follows.

Now, from (3.3.2), we have

$$e^{(k+1)} = g'(x^*) e^{(k)},$$

(3.3.6)

The smaller $|g'(x^*)|$, the more rapidly $e^{(k)} \to 0$ as $n \to \infty$, therefore when $g'(x^*) = 0$ the iteration is most rapid.

Now, if $g(x)$ is twice differentiable, then by Taylor's expansion, we get,

$$e^{(k+1)} = x^* - x^{(k+1)} = g(x^*) - g(x^{(k)})$$

$$= - g'(x^*)(x^{(k)} - x^*) - \frac{1}{2} g''(\xi)(x^{(k)} - x^*)^2,$$

for some $\xi \in (x^{(k)}, x^*)$,

$$e^{(k+1)} = g'(x^*) e^{(k)} - \frac{1}{2} g''(\xi) [e^{(k)}]^2.$$  

(3.3.7)

If $g'(x^*) = 0$ and $g''(x^*)$ is definite, then

$$e^{(k+1)} \approx - \frac{1}{2} g''(x^*) [e^{(k)}]^2,$$

for large $k$.  

(3.3.8)

Clearly, $e^{(k+1)}$ is (roughly) a quadratic function of $e^{(k)}$, so we say that the method converges quadratically.

The Newton's method with $g(x) = x - f(x)/f'(x)$ converges quadratically where its error equation can be written as (see Conte, 1981)

$$e^{(k+1)} = - \frac{1}{2} \frac{f''(x^{(k)})}{f'(x^{(k)})} [e^{(k)}]^2,$$

(3.3.9)
for some \( \eta^{(k)} \in (x^*, x^{(k)}) \).

For the secant method, it can be shown (see Conte, 1981) that

\[
\eta^{(k+1)} \approx \frac{1}{2} \frac{f''(x^*)}{f'(x^*)} \eta^{(k)} \eta^{(k-1)}.
\]

We notice that here the order of convergence is between 1 and 2.

Now, we can define the order of convergence in a fashionable way:

**Definition 3.3.1**

Let \( x^{(0)}, x^{(1)}, \ldots \) be a sequence which converges to a number \( x^* \), and set \( \eta^{(k)} = x^* - x^{(k)} \). Suppose there exist a number \( p \) and a constant \( C \neq 0 \) such that

\[
\lim_{k \to \infty} \frac{|\eta^{(k+1)}|}{|\eta^{(k)}|^p} = C,
\]

then we call \( p \) the order of convergence of the sequence and \( C \) the asymptotic error constant. The convergence of iteration is said to be linear if \( p = 1 \) and quadratic if \( p = 2 \). If \( 1 < p < 2 \), then it is superlinear.

### 3.4 TWO-POINT NEWTON METHOD

A method is now proposed in which the locally quadratic convergence of Newton Raphson and the more robust global behaviour of the False Position method are retained by using function and derivative values at two points which bracket the desired root.

Suppose we have \( x_L, x_R \in I = [a, b] \) where \( f(x) \) is continuous and differentiable and such that \( f(x_L)f(x_R) < 0 \). The Taylor expansions of \( f(x) \) about \( x_L \) and \( x_R \) are respectively
\[ f(x) = f(x_r) + (x-x_r)f'(x_r) + O(x-x_r)^2 \]  
\( f(x) = f(x_r) + (x-x_r)f'(x_r) + O(x-x_r)^2 \)  

The corresponding Newton points arising from \( f(x) = 0 \) to the first order are:

\[ x_{\ell \ell} = x_{\ell} - \frac{f_{\ell}}{f'_{\ell}} \]  
\( x_{r r} = x_{r} - \frac{f_{r}}{f'_{r}} \)  

where \( f_{\ell} = f(x_{\ell}) \), \( f_{r} = f(x_{r}) \) and \( f'_{\ell} \) similarly for \( f'_{\ell} \) and \( f'_{r} \).

Alternatively, if (3.4.1) and (3.4.2) are equated to first order, we get

\[ x(f'_{r} - f'_{\ell}) = f_{\ell} - x_{\ell}f'_{\ell} - f_{r} + x_{r}f'_{r} \]

the solution of which represents a third Newton point \( x_n \) say

\[ x_n = \frac{x_{\ell}f'_{r} - x_{\ell}f'_{\ell} - f_{r} + f_{\ell}}{f'_{r} - f'_{\ell}} \]  

Using (3.4.3) and (3.4.4) an alternative expression is derived, i.e.

\[ x_n = \frac{x_{\ell} f'_{r} - x_{\ell} f'_{\ell}}{f'_{r} - f'_{\ell}} \]

This construction forms the basis of the proposed two-point iteration and is illustrated geometrically in figure (3.4.1).
To see whether the point $x_n$ generated by the 2-point Newton method lies in the domain $(x_L, x_r)$ or outside it, (see figures 3.4.2 and 3.4.3), we have the following:

**LEMMA: 3.4.1**

If $f(x)$ and $f'(x)$ are defined at distinct points $x_L$ and $x_r$ ($x_L < x_r$) and $S_{lr} = \frac{f_r - f_L}{x_r - x_L}$, such that

$$ (f'_L - S_{lr})(f'_r - S_{lr}) < 0, \quad (3.4.7) $$

then $x_L < x_n < x_r$,

where $x_n$ is given by (3.4.5) or (3.4.6).

**Proof:**

Using $S_{lr} = \frac{f_r - f_L}{x_r - x_L}$ in (3.4.7) we get

$$ \left( f'_L - \frac{f_r - f_L}{x_r - x_L} \right) \left( f'_r - \frac{f_r - f_L}{x_r - x_L} \right) < 0. \quad (3.4.8) $$

Now consider the first term on the l.h.s.
Similarly for the second term of the l.h.s. of (3.4.8), we have

\[
\frac{f'_r - f'_\ell}{x_r - x_\ell} = \frac{x_n f'_r - x_n^r f' - f_r + f_\ell}{x_r - x_\ell}, \quad \text{from (3.4.5)}
\]

\[
\frac{(x_n - x_r)(f'_r - f'_\ell)}{x_r - x_\ell}. \quad (3.4.9)
\]

Now, by substituting the equalities (3.4.9) and (3.4.10) in the inequality (3.4.8), we obtain

\[
\frac{(x_n - x_r)(f'_r - f'_\ell)}{(x_n - x_\ell)(f'_r - f'_\ell)} < 0
\]

or

\[
(x_n - x_r)(x_n - x_\ell) \left[ \frac{(f'_r - f'_\ell)}{(x_r - x_\ell)} \right]^2 < 0
\]

or

\[
(x_n - x_r)(x_n - x_\ell) < 0, \quad (3.4.11)
\]

since

\[
\left[ \frac{(f'_r - f'_\ell)}{(x_r - x_\ell)} \right]^2 > 0.
\]

The inequality (3.4.11) holds if and only if \(x_\ell < x_n < x_r\). We will refer to the inequality (3.4.7) as the convexity condition throughout this chapter.
Figure (3.4.2) illustrates the case, $x_n \notin (x_s, x_r)$ and $x_t < x_n < x_r$.

Figure (3.4.3) illustrates the case, $x_n \in (x_s, x_r)$ i.e. $x_s < x_n < x_r$. 
3.4.1 Basic Two-Point Newton Algorithm

This algorithm is based on the idea that if the condition in Lemma 3.4.1 is satisfied we consider $x_n$ (2-point Newton point) as the new approximation otherwise we take $x_s$ (Secant point).

Thus, we propose the algorithm with convergence test on $f(x)$ using some small tolerance $\varepsilon$.

1. Input $x_L, x_R$ such that $f_L f_R < 0$

2. If $(f'_L - S_{x_L})(f'_R - S_{x_R}) < 0$ then
   
   \[ x = x_n, \text{ using (3.4.6)}, \]

   else
   
   \[ x = x_s, \text{ using (3.2.4)}, \]

3. If $|f_x| < \varepsilon$ then stop (convergence in $f$)

4. Update using the sign test:
   
   if $f_R f_x < 0$ then $x_L = x$

   else
   
   \[ x_R = x \]

5. Repeat (2), (3) and (4) until convergence.

This algorithm was implemented as a FORTRAN 77 routine (see Flowchart in figure (3.4.4)) and it was noticed that it converges very slowly. Also the Newton points $x_L$ and $x_R$ (obtained by equation (3.2.7)) at some stages are much better approximations than $x_n$. 
Given $f, f', S_{x r}$

$x_{x}, x_{r}$

s.t. $f_{x}^{2} f_{x} < 0$

$n = 0$

Compute $f_{x}, f_{r}, f_{x}', f_{r}'$

and $S_{x r}$

if $f_{x}' = 0$

or $f_{x}' = 0$

or $f_{r}' = 0$

or $f_{r}' = 0$

or $(f_{x}' - S_{x r})(f_{r}' - S_{r r}) < 0$

$x = x_{n}$

$x = x_{s}$

STOP

$n = n + 1$

if $|f_{x}| < \varepsilon$

$x_{x} = x$

if $f_{x} f_{x} < 0$

$x_{r} = x$

Figure (3.4.4), Basic Two-Point Newton Algorithm
3.4.2 Modified Two-Point Newton Algorithm

In this section, we develop an algorithm (four versions) to find a zero of a function using the secant method, Newton-Raphson and the two-point Newton methods.

**Version 1**

In this version of the algorithm we modify step (2) in the basic algorithm by adding a new condition which allows us to take \(x_{r+1}\) or \(x_{r}\) when \(x_{n}\) is not better than both of them. Also we will use a higher (second) order approximation for the ordinary Newton value \((x_{r+1}\) or \(x_{r}\)) when considered as the next approximation. We will use no more function evaluations than those which are already available. This will allow us to have a better approximation of the zero.

Consider the Taylor series expansion of \(f(x)\) at \(x_{r}\) and ignore terms of degree 3 and above,

\[
f(x) \approx f_{r} + (x-x_{r})f'_{r} + \frac{(x-x_{r})^{2}}{2!} f''_{r} ,
\]

(3.4.12)

where, \(f'_{r} = f'(x_{r})\) and \(f''_{r} = f''(x_{r})\).

If \(x\) is a good approximation to the root, then \(f(x) \approx 0\), it follows from (3.4.12) that

\[
x = - \frac{f_{r}}{f'_{r}} + x_{r} - \frac{(x-x_{r})^{2}}{2! f'_{r}} f''_{r} .
\]

(3.4.13)

Now, consider the approximation to \(f''_{r}\)

\[
f''_{r} = \frac{f'_{r} - f'_{r}}{x_{r} - x_{r}}
\]

(3.4.14)
and for \((x - x_i)\) in the r.h.s. of \((3.4.13)\) as

\[
(x - x_i^*) = -\frac{f_{i'}}{f'_{i'}}.
\]  

(3.4.15)

So, equation \((3.4.13)\) becomes,

\[
x_{i2} = x_i + \frac{f_i^2(f_{r}' - f'_{r})}{2(f'_{r})^2(x_{r} - x_i^*)}.
\]  

(3.4.16)

Similarly, for \(x_{r2}\), we get

\[
x_{r2} = x_r - \frac{f^2_{r}(f_{r}' - f'_{r})}{2(f'_{r})^3(x_{r} - x_r^*)}.
\]  

(3.4.17)

The second order approximations of \(x_{i2}\) and \(x_{r2}\) given here may be used to improve the new approximation.

Now, step \((2)\) in the basic algorithm becomes:

\(2\) If \((f_{i}' - S_{ki})(f_{r}' - S_{kr}) < 0\) then

(i) Compute \(x_{i2}, x_{r2}\) and \(x_s\) using \((3.4.3), (3.4.4)\) and \((3.2.4)\) respectively.

(ii) If \((x_{i} < x_{i2} < x_{r} < x_r)\) then

if \(x_s < x_{i2}\) then \(x = x_{i2}\) (using \((3.4.16)\))

(A) \[\text{if } x < x_s \Rightarrow x = x_s\]

else if \(x_s > x_{r2}\) then \(x = x_{r2}\) (using \((3.4.17)\))

if \(x > x_s \Rightarrow x = x_s\)

end if

else
(B) \[ x = x_n \]
    end if

else

(C) \[ x = x_s \]
    end if

**Version 2**

In this version we try to improve the calculated value of \( x_n \)
by modifying one of the points \( x_L \) or \( x_R \) which are used to
calculate \( x_n \).

We effectively replace any value \( (x_L \) or \( x_R \)) which lies
outside the interval \( (x_L, x_R) \) in the following manner.

If \[ x_L < x < x_L \Rightarrow x = x_L \]
else if \[ x_R < x_r < x_L \Rightarrow x = x_L \]
else if \[ x_r > x \Rightarrow x = x_r \]
else if \[ x_r < x_L \Rightarrow x = x_r \]

Now \( x_n \) is calculated according to the conditional tests above.

In the previous version we considered \( x_n \) as the new approximation
when it was calculated but here we add two choices before deciding
on the new approximation. These cases are:

if \[ x_L < x_L \leq x_s \leq x_n \Rightarrow x = x_s \]
else if \[ x_n \leq x_s \leq x_r < x_n \Rightarrow x = x_s \]
else \[ x = x_n \].
Version 3

When the convexity condition (3.4.7) did not apply, we used to consider \( x_s \) as the new approximation. In this version we add a few cases to give more choices of \( x \) before deciding on \( x_s \) (based on the higher order approximation described in section (3.5.1)) and this should result in an improved approximation. \( \varepsilon \) is a given small number. Consequently, we will have

\[
\text{If } (f'_x - S_{x \| r})(f'_r - S_{x \| r}) \geq 0 \text{ then }
\]

(A) \( \text{If } |f'_x| < \frac{1}{3} |f'_r| \text{ then } \)

\[
\text{if } \frac{f'_r}{S_{x \| r}} > 1 \text{ then } x = x_{x \| r} - \frac{f'_r (S_{x \| r} - f'_x)}{2(f'_r)^3} (x_r - x_x)
\]

else if \( 1 \geq \frac{f'_r}{S_{x \| r}} > \varepsilon \text{ then } x = x_x - \frac{2f'_x}{S_{x \| r} + f'_r}
\]

else if \( \frac{f'_r}{S_{x \| r}} \leq \varepsilon \text{ then } x = x_x - \frac{2f'_x}{S_{x \| r}}
\]

end if

(B) \( \text{ELSE if } |f'_r| < \frac{1}{3} |f'_x| \text{ then } \)

\[
\text{if } \frac{f'_x}{S_{x \| r}} > 1 \text{ then } x = x_{x \| r} - \frac{f'_x (f'_r - S_{x \| r})}{2(f'_x)^3} (x_r - x_x)
\]

else if \( 1 \geq \frac{f'_x}{S_{x \| r}} > \varepsilon \text{ then } x = x_r - \frac{2f'_r}{S_{x \| r} + f'_x}
\]

else if \( \frac{f'_x}{S_{x \| r}} \leq \varepsilon \text{ then } x = x_r - \frac{2f'_r}{S_{x \| r}}
\]

end if

(C) \( \text{ELSE } x = x_s \).
It is observable that convergence of some stages is slow and monotonic especially for the test functions (4) and (5). This has led us to apply Aitken's $\delta^2$ method on any three successive approximations in the same direction (towards $x_\downarrow$ or $x_\uparrow$). This did not improve the convergence in any significant way. The Bisection method was tried based on the idea of Aitken's $\delta^2$ that is to take the average of $x_\downarrow$ and $x_\uparrow$ (one of them would be the third of three successive values in the same direction) as the new approximation. Further we applied this technique to 4 and 5 successive values and we found the 4 successive values method to be the best from the number of iterations point of view.

3.5 EXPERIMENTAL RESULTS

We applied the algorithm with its various variations to the following test functions,

(1) $f(x) = e^{-x} - \sin x$ \hspace{1cm} (3.5.1)
which has zeros at 0.588533, 3.096364, $n\pi$
For $n = 2,3,\ldots,\infty$.

(2) $f(x) = 1/(x-2) + x^2$ \hspace{1cm} (3.5.2)
having three zeros at -0.618034, 1.0 and 1.618034.

(3) $f(x) = x^4 - x - 10$ \hspace{1cm} (3.5.3)
which has two zeros at -1.697473 and 1.855585.

(4) $f(x) = x^5 - 5.95x^4 + 11.2x^3 - 6x^2 + 0.1x - 2.6016$ \hspace{1cm} (3.5.4)
which has a zero at 2.262520 and zeros in (2.02, 2.027).
The stopping criterion used in the algorithm is

\[ |f(x)| \leq \varepsilon, \text{ where } \varepsilon = 10^{-6}. \quad (3.5.5) \]

We did not include a convergence test on every two successive iterates, i.e. \( |x^{(k+1)} - x^{(k)}| \leq \varepsilon \), for \( k = 0, 1, \ldots \), to avoid stopping the computation in the case of a very slow convergence (where the solution has not been obtained yet) as we noticed in some experiments.

Tables (3.5.1) to (3.5.4) contain the results of applying the basic algorithm and the four variations to the four test functions described earlier. Each table contains different domains on which the algorithm and its variations are applied, together with the roots available in each domain. Also the number of iterations corresponding to each version together with the root obtained.

Generally, for our test functions, there is no considerable difference between the results of Version 3 and that of Version 4. For function (4), i.e. \( f(x) = x^5 - 2.6016 \), we notice from table (3.5.4) that there are more than one root in the interval (2.02, 2.027). Such a function will beat most of the methods.
<table>
<thead>
<tr>
<th>Domain</th>
<th>(0,2)</th>
<th>(1,6)</th>
<th>(5,7)</th>
<th>(0,7)</th>
<th>(2,10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root</td>
<td>0.588533</td>
<td>3.096364</td>
<td></td>
<td>0.588533</td>
<td>0.588533</td>
</tr>
<tr>
<td>Basic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Algorithm</td>
<td>12</td>
<td>15</td>
<td>15</td>
<td>16</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>0.588533</td>
<td>3.096364</td>
<td>6.285049</td>
<td>6.285049</td>
<td>9.424697</td>
</tr>
<tr>
<td>Version 1</td>
<td>6</td>
<td>13</td>
<td>3</td>
<td>16</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>0.588533</td>
<td>3.096364</td>
<td>6.285049</td>
<td>6.285049</td>
<td>9.424697</td>
</tr>
<tr>
<td>Version 2</td>
<td>3</td>
<td>6</td>
<td>3</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>0.588533</td>
<td>3.096364</td>
<td>6.285049</td>
<td>6.285049</td>
<td>9.424697</td>
</tr>
<tr>
<td>Version 3</td>
<td>3</td>
<td>6</td>
<td>3</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>0.588533</td>
<td>3.096364</td>
<td>6.285049</td>
<td>6.285049</td>
<td>9.424697</td>
</tr>
<tr>
<td>Version 4</td>
<td>3</td>
<td>6</td>
<td>3</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>0.588533</td>
<td>3.096364</td>
<td>6.285049</td>
<td>6.285049</td>
<td>9.424697</td>
</tr>
</tbody>
</table>

Table (3.5.1), \( f(x) = e^{-x} - \sin x \)
<table>
<thead>
<tr>
<th>Domain</th>
<th>(-1,0)</th>
<th>(.5,1.3)</th>
<th>(1.2,1.8)</th>
<th>(.5,2.1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root</td>
<td>-.618034</td>
<td>1.0</td>
<td>1.618034</td>
<td>-.618034</td>
</tr>
<tr>
<td>Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Basic</td>
<td>11</td>
<td>10</td>
<td>15</td>
<td>162</td>
</tr>
<tr>
<td>Algorithm</td>
<td>-.618034</td>
<td>1.0</td>
<td>1.618034</td>
<td>1.0</td>
</tr>
<tr>
<td>Version 1</td>
<td>3</td>
<td>10</td>
<td>6</td>
<td>162</td>
</tr>
<tr>
<td>Version 2</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>162</td>
</tr>
<tr>
<td>Version 3</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>84</td>
</tr>
<tr>
<td>Version 4</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>9</td>
</tr>
</tbody>
</table>

Table (3.5.2), \( f(x) = \frac{1}{x-2} + x^2 \)
<table>
<thead>
<tr>
<th>Domain</th>
<th>(-3, -1)</th>
<th>(-3, 1.5)</th>
<th>(1, 2)</th>
<th>(1.4, 1.96)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root Method</td>
<td>-1.697472</td>
<td>-1.697472</td>
<td></td>
<td>1.855585</td>
</tr>
<tr>
<td>Basic Algorithm</td>
<td>16</td>
<td>18</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>-1.697472</td>
<td>-1.697472</td>
<td>1.855585</td>
<td>1.855585</td>
</tr>
<tr>
<td>Version 1</td>
<td>6</td>
<td>6</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>-1.697472</td>
<td>-1.697472</td>
<td>1.855585</td>
<td>1.855585</td>
</tr>
<tr>
<td>Version 2</td>
<td>5</td>
<td>6</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>-1.697472</td>
<td>-1.697472</td>
<td>1.855585</td>
<td>1.855585</td>
</tr>
<tr>
<td>Version 3</td>
<td>5</td>
<td>6</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>-1.697472</td>
<td>-1.697472</td>
<td>1.855585</td>
<td>1.855585</td>
</tr>
<tr>
<td>Version 4</td>
<td>5</td>
<td>6</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>-1.697472</td>
<td>-1.697472</td>
<td>1.855585</td>
<td>1.855585</td>
</tr>
</tbody>
</table>

Table (3.5.3), \( f(x) = x^4 - x - 10 \)
<table>
<thead>
<tr>
<th>Domain</th>
<th>(-1,5)</th>
<th>(0,3)</th>
<th>(2.01,2.28)</th>
<th>(2.1,2.4)</th>
<th>(2,2.5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root</td>
<td>All roots</td>
<td>All roots</td>
<td>All roots</td>
<td>2.262522</td>
<td>All roots</td>
</tr>
<tr>
<td>Method</td>
<td>Basic Algorithm</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>35</td>
<td>16</td>
<td>9</td>
<td>12</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>2.026680</td>
<td>2.262523</td>
<td>2.026702</td>
<td>2.262519</td>
<td>2.026671</td>
</tr>
<tr>
<td>Version 1</td>
<td>57</td>
<td>11</td>
<td>5</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>2.262522</td>
<td>2.262521</td>
<td>2.026703</td>
<td>2.262519</td>
<td>2.026636</td>
</tr>
<tr>
<td>Version 2</td>
<td>56</td>
<td>32</td>
<td>4</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>2.262525</td>
<td>2.262524</td>
<td>2.026661</td>
<td>2.262521</td>
<td>2.026636</td>
</tr>
<tr>
<td>Version 3</td>
<td>46</td>
<td>21</td>
<td>4</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>2.262522</td>
<td>2.262523</td>
<td>2.026659</td>
<td>2.262521</td>
<td>2.026636</td>
</tr>
<tr>
<td>Version 4</td>
<td>22</td>
<td>11</td>
<td>4</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>2.262519</td>
<td>2.262521</td>
<td>2.026659</td>
<td>2.262521</td>
<td>2.262520</td>
</tr>
</tbody>
</table>

Table (3.5.4), $f(x) = x^5 - \ldots - 2.6016$
Chapter Four

ITERATIVE METHODS FOR SPARSE SYSTEMS OF LINEAR EQUATIONS

4.1 Introduction
4.2 The Model Problem: Two Dimensional Case
4.3 Basic Iterative Methods
4.4 Convergence of Basic Iterative Methods
4.5 Rate of Convergence
4.6 Theoretical Determination of the Optimal Relaxation Parameter $\omega$
4.7 Semi-Iterative Methods and Chebyshev Polynomials
4.8 Chebyshev Acceleration
4.9 Mildly NonLinear Elliptic Equations
   4.9.1 Formulation and Method of Solution
   4.9.2 Application and Results
4.10 Computational Complexity
4.1 INTRODUCTION

Consider the non-singular linear system of equations

\[ Au = b \]  \hspace{1cm} (4.1.1)

To solve the system iteratively we need a sequence of vectors \( \mathbf{u}^{(k)} \),
\( k = 0, 1, 2, \ldots \) such that,

\[ \mathbf{u}^{(k)} = A^{-1}b \text{ as } k \to \infty. \]

Here \( \mathbf{u}^{(k)} \) is a function of \( A, b, u^{(k-1)}, \ldots, u^{(k-r)} \), where \( r \) is called the degree of the iteration. To save computer storage \( r \) is usually taken to be small, i.e. equal to 1, which is then a first degree method, whence we can write,

\[ \mathbf{u}^{(k)} = F_k(A, b, u^{(k-1)}). \]  \hspace{1cm} (4.1.2)

The iterative method is said to be stationary if for some integer \( i > 0 \), \( F_k \) is independent of \( k \) for all \( k \geq i \). Otherwise, it is non-stationary.

Also it is linear if for each \( k \), \( F_k \) is a linear function of \( u^{(k-1)} \). Otherwise it is nonlinear.

The most general linear stationary iterative method of first degree has the form,

\[ \mathbf{u}^{(k+1)} = G_k \mathbf{u}^{(k)} + \mathbf{r}, \]  \hspace{1cm} (4.1.3)

where \( G \) is called the iteration matrix which depends on \( A \) and \( b \), and \( \mathbf{r} \) is a column vector.

The exact solution \( \mathbf{u} = A^{-1}b \) can be expressed according to (4.1.3) as,

\[ A^{-1}b = GA^{-1}b + \mathbf{r}, \]  \hspace{1cm} (4.1.4)

hence,

\[ \mathbf{r} = (I-G)A^{-1}b. \]  \hspace{1cm} (4.1.5)

This is called the consistency condition.
If the consistency condition applies (Young, 1971), then for some k,

$$u^{(k+1)} = Gu^{(k)} + r = Gu + r = u,$$  \hspace{1cm}, \hspace{1cm} (4.1.6)

i.e. as soon as the solution is obtained, the iterative process does not make any further modification of successive iterates (see Young, 1971, p.65).

4.2 THE MODEL PROBLEM: TWO DIMENSIONAL CASE

Now we describe the use of the finite difference approximation to partial derivatives (Section 2.1) to approximate the two dimensional Dirichlet problem involving the Laplace equation defined on a unit region. Thus, we consider the following boundary value problem

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0, \hspace{1cm} (x,y) \in R \hspace{1cm} (4.2.1)$$

$$u(x,y) = g(x,y), \hspace{1cm} (x,y) \in C . \hspace{1cm} (4.2.2)$$

The problem is to obtain an approximation to the solution $u(x,y)$ which satisfies the equation (4.2.1) and the boundary condition (4.2.2). The region $R$ under consideration is covered by a rectangular net with mesh spacing $h$ in the $x$ and $y$ directions and mesh points $(x_i,y_j)$, where $x_i = ih$, and $y_j = jh$, $i,j = 0,1,\ldots,n$.

By substituting the finite difference approximations,

$$\frac{\partial^2 u}{\partial x^2} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} \quad \text{and} \quad \frac{\partial^2 u}{\partial y^2} = \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2}.$$

in equation (4.2.1) we obtain the following five-point formula,

$$-u_{i-1,j} + u_{i+1,j} + 4u_{i,j} - u_{i,j-1} - u_{i,j+1} = 0 , \hspace{1cm} (4.2.3)$$

which can be represented by the computational molecule shown in Figure (4.2.1),
which when applied at each mesh point according to the following internal row-wise ordering (Figure 4.2.2), where $h = \frac{1}{m}$ and $n = m - 1$.

yields the linear system,

$$Au = b \quad (4.2.4)$$

The coefficient matrix $A$ of the system (4.2.4) is a real, square, symmetric matrix and can be written in the following block tridiagonal form

$$A = \begin{bmatrix} A_1 & -I \\ -I & A_2 & -I \\ & -I & \ddots & \ddots \\ & & -I & A_n \end{bmatrix} \quad (4.2.5)$$
where $I$ is the unit matrix of order $n$ and $A_i$ ($1 \leq i \leq n$) is also of order $n$ and given by,

$$A_i = \begin{bmatrix}
4 & -1 & & \\
-1 & 4 & -1 & \\
& -1 & & \\
& & & -1
\end{bmatrix}, \quad 1 \leq i \leq n. \quad (4.2.6)$$

### 4.3 BASIC ITERATIVE METHODS

The $(n\times n)$ non-singular matrix $A$ of the system (4.1.1) may be expressed as,

$$A = Q - C, \quad (4.3.1)$$

where $Q$ and $C$ are $(n\times n)$ matrices, and $Q$ is non-singular. Equation (4.1.1) can be written as,

$$Qu = Cu + b.$$

By adding the subscripts $(k+1)$ on the left and $(k)$ on the right, we obtain the following iterative method,

$$Qu^{(k+1)} = Cu^{(k)} + b. \quad (4.3.2)$$

Equation (4.3.1) represents a splitting of the matrix $A$. It is clear that different splittings of the matrix $A$ result in different iterative methods. All the basic iterative methods may assume the partition,

$$A = D - L - U,$$
where $D$ is a diagonal matrix and $L$ and $U$ are strictly lower triangular and upper triangular matrices usually chosen to the given components of the matrix $A$.

Now, for the linear system (4.1.1), let $a_{ij}$, $i = 1, \ldots, n$ be the elements of $A$ and $b_i$, $i = 1, \ldots, n$ those of $b$. We assume that the diagonal elements are non-zero, i.e., $a_{ii} \neq 0$, $i = 1, 2, \ldots, n$.

The $i$th equation of (4.1.1) is written as,

$$a_{i1}u_1 + a_{i2}u_2 + \ldots + a_{ii}u_i + \ldots + a_{in}u_n = b_i.$$

Let,

$$u^{(k)} = (u_1^{(k)}, u_2^{(k)}, \ldots, u_n^{(k)})^T,$$

with $k = 0, 1, 2, \ldots$, (4.3.4)

be the vector of the approximate solution obtained at the end of the $k$th iteration with $u^{(0)}$ some initial estimate of the solution.

We can now define the following iterative methods:

1. **The Jacobi Method**

   Each iteration of this method can be carried out using the form,

   $$u_i^{(k+1)} = \left[ b_i - \sum_{j=1}^{n} a_{ij}u_j^{(k)} \right] / a_{ii}, \quad i = 1, 2, \ldots, n \quad (4.3.5)$$

   as derived from equation (4.3.3). The procedure (4.3.5) is repeated for $k = 0, 1, 2, \ldots$ until convergence is achieved. To write the Jacobi method in matrix form we let $Q = D$ and $C = L + U$, so equation (4.3.2) becomes,

   $$Du^{(k+1)} = (L+U)u^{(k)} + b.$$

   (4.3.6)

   We require $D$ to be nonsingular, i.e. $D^{-1}$ exists. Then from (4.3.6) we obtain,

   $$u^{(k+1)} = D^{-1}(L+U)u^{(k)} + D^{-1}b,$$

   (4.3.7)
\[ B = D^{-1}(L+U) \] is called the Jacobi iteration matrix.

It is clear that all the elements of \( u^{(k)} \) must be saved while calculating the elements of \( u^{(k+1)} \), i.e. no overwriting of the element of the vector \( u^{(k)} \) in the \( k+1 \)th iteration.

This method is also called the method of Simultaneous Displacement.

Related to the Jacobi method is the Jacobi overrelaxation method which uses a real parameter \( \omega \) such that,

\[
u^{(k+1)}_i = u^{(k)}_i + \omega(\hat{u}^*_i - u^{(k)}_i),\]

where \( \hat{u}^*_i \) is obtained by the Jacobi method (4.3.5) and \( \omega > 1 \).

Accordingly the method (4.3.5) becomes,

\[
u^{(k+1)}_i = \omega \left[ b_i - \sum_{j=1}^{n} a_{ij} u^{(k)}_j \right]/a_{ii} + (1-\omega)u^{(k)}_i \]

or equivalently, in a matrix form,

\[
u^{k+1} = \left[ \omega D^{-1}(L+U) + (1-\omega)I \right] u^{(k)} + \omega D^{-1} b.\]

If \( \omega = 1 \), we have the Jacobi method.

2. The Gauss-Seidel Method (GS)

The Jacobi method can be modified by inserting the most recent values \( u^{(k+1)}_i \) in the remaining equations as they become available, so equation (4.3.5) may be replaced by,

\[
u^{(k+1)}_i = \left[ b_i - \sum_{j=1}^{i-1} a_{ij} u^{(k+1)}_j - \sum_{j=i+1}^{n} a_{ij} u^{(k)}_j \right]/a_{ii}, \quad i = 1, 2, \ldots, n \]

(4.3.8)

To write the G.S. method (4.3.8) in matrix form we set \( Q = D - L \) and \( C = U \), equation (4.3.2) becomes
\[(D-L)u^{(k+1)} = Uu^{(k)} + b, \quad (4.3.9)\]

or
\[Du^{(k+1)} = Lu^{(k+1)} + Uu^{(k)} + b. \quad (4.3.10)\]

The G.S. method in explicit form is defined by,
\[u^{(k+1)} = (D-L)^{-1} Uu^{(k)} + (D-L)^{-1} b, \quad (4.3.11)\]

where \(L = (D-L)^{-1}U\) is the G.S. iteration matrix.

The method is also called the *Successive Displacement* method.

It can be seen that the method does not require the simultaneous storage of the values of \(u^{(k+1)}\) and \(u^{(k)}\) and therefore efficiently uses the computer's memory.

3. The Successive Overrelaxation Method (SOR)

The convergence of the G.S. method can be accelerated by introducing a relaxation parameter \(\omega\). The method is then called the *Successive Overrelaxation* method.

Rather than taking the value of \(u_i^{(k+1)}\), obtained by (4.3.8), as the solution of the given system, we set,
\[u_i^{(k+1)} = u_i^{(k)} + \omega \left( u_i^{(k+1)} - u_i^{(k)} \right), \quad (4.3.12)\]

where \(u_i\) is obtained by the G.S. method (4.3.8) and \(\omega\) is an overrelaxation parameter such that \(1 < \omega \leq 2\).

If we substitute \(u_i^{(k+1)}\) (from (4.3.8)) into equation (4.3.12), we obtain, after some rearranging,
\[u_i^{(k+1)} = (1-\omega)u_i^{(k)} + \omega \sum_{j=1}^{i-1} a_{ij} u_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} u_j^{(k)} + \frac{1}{a_{ii}} \left( b_i - \sum_{j=1}^{i-1} a_{ij} u_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} u_j^{(k)} \right) \]

\[i = 1, 2, \ldots, n, \quad (4.3.13)\]
which can be expressed in matrix form, with the use of (4.3.10), as,

\[ u^{(k+1)} = u^{(k)} - \omega D^{-1} \left[ L u^{(k+1)} + D u^{(k)} + b \right] \]  

(4.3.14)

or

\[ (D - \omega L) u^{(k+1)} = \omega U + (1 - \omega) D u^{(k)} + \omega b. \]  

(4.3.15)

By the assumption that D is a non-singular matrix, thus \((D - \omega L)^{-1}\) exists. Equation (4.3.15) becomes,

\[ u^{(k+1)} = (D - \omega L)^{-1} \left[ \omega U + (1 - \omega) D u^{(k)} + (D - \omega L)^{-1} \omega b \right]. \]  

(4.3.16)

We now let,

\[ L_w = (D - \omega L)^{-1} \left[ \omega U + (1 - \omega) D \right], \]  

(4.3.17)

where \(L_w\) denotes the SOR iteration matrix. Furthermore, equation (4.3.16) can be written in the form,

\[ u^{(k+1)} = u^{(k)} - \omega (D - \omega L)^{-1} r, \]  

(4.3.18)

where \(r = (A u^{(k)} - b)\) is the residual vector.

For \(\omega = 1\), the method becomes the G.S. method, and for \(\omega < 1\), we have the Successive Underrelaxation method.


Each iteration of the SSOR method consists of two distinct passes (sweeps) through the components of the solution (unknown) vector \(u\).

In the first half of the iteration, the SOR method is used with a relaxation parameter \(\omega\) in a forward sweep through \(u_1^{(k)}, u_2^{(k)}, \ldots, u_n^{(k)}\) and will generate new values, we call \(u_1^{(k+1)}, u_2^{(k+1)}, \ldots, u_n^{(k+1)}\).

In the second half of the iteration, the SOR method is used with the same relaxation parameter used in the first half of the iteration,
in a backward (reverse) sweep through \( u^{(k+1)}_{n}, u^{(k+1)}_{n-1}, \ldots, u^{(k+1)}_{1} \) and will generate the final values of the iteration which we will call \( u^{(k+1)} \).

The two sweeps may be written as,

\[
\begin{align*}
    u^{(k+1)} &= L_{\omega} u^{(k)} + k_{1}, \quad (4.3.19) \\
    u^{(k+1)} &= L_{\omega}^{b} u^{(k+1)} + k_{2}, \quad (4.3.20)
\end{align*}
\]

where \( L_{\omega} \) is given in (4.3.17) and

\[
L_{\omega}^{b} = (D-\omega U)^{-1} \left( (1-\omega)D + \omega L \right),
\]

and \( k_{1} \) and \( k_{2} \) are constant vectors.

By eliminating the intermediate approximation \( u^{(k+1)} \) from (4.3.20), we obtain

\[
\begin{align*}
    u^{(k+1)} &= G_{\text{SSOR}} u^{(k)} + k, \quad (4.3.22)
\end{align*}
\]

where \( G_{\text{SSOR}} \) is the SSOR method iteration matrix which is given by,

\[
G_{\text{SSOR}} = (D-\omega U)^{-1} \left( (1-\omega)D + \omega L \right) (D-\omega L)^{-1} \left( (1-\omega)D + \omega U \right),
\]

and

\[
k = (D-\omega U)^{-1} \left( (1-\omega)D + \omega L \right) (D-\omega L)^{-1} \omega b + (D-\omega U)^{-1} \omega b.
\]

Another variation of the SSOR method is the unsymmetric SSOR (USSOR) which may be described in a similar way as the SSOR method (see Lapidus, 1982), except that the relaxation parameter \( \omega \) has two different values, i.e. \( \omega_{1} \) and \( \omega_{2} \), one of them is used in the first sweep and the other value in the second sweep.

We summarize the iterative methods described earlier in the following table according to the general form,

\[
\begin{align*}
    u^{(k+1)} &= G u^{(k)} + r,
\end{align*}
\]
METHOD | THE ITERATION MATRIX G | THE VECTOR r  
--- | --- | ---  
Jacobi | $D^{-1}(L+U)$ | $D^{-1}b$  
JOR | $\omega D^{-1}(L+U)+(1-\omega)I$ | $\omega D^{-1}b$  
G.S. | $(D-L)^{-1}U$ | $(D-L)^{-1}b$  
SOR | $(D-\omega L)^{-1}[\omega U+(1-\omega)D]$ | $(D-\omega L)^{-1}\omega b$  
SSOR | $(D-\omega U)^{-1}[((1-\omega_2)D+\omega_2 U)*]$ | $(D-\omega_2 U)^{-1}[(1-\omega_2)D+\omega_2 U]*$  
USSOR | $(D-\omega_1 U)^{-1}[(1-\omega_1)D+\omega_1 U]$ | $(D-\omega_1 U)^{-1}\omega_1 b+(D+\omega_1 U)^{-1}\omega_2 b$  

TABLE 4.3.1

where G, r, D, L, U and b are as defined before.

4.4 CONVERGENCE OF BASIC ITERATIVE METHODS

We observe that each of the iterative methods described previously, can be written in the form,

$$u^{(k+1)} = Gu^{(k)} + r \quad (4.4.1)$$

where G is the iteration matrix and r is a constant vector.

Definition 4.4.1

The iterative method (4.4.1) converges if,

$$\lim_{k \to \infty} u^{(k)} = u \quad (4.4.2)$$

for any initial vector $u^{(0)}$ and where u is the exact solution of the system,

$$Au = b \quad (4.4.3)$$
Theorem 4.4.1

The iterative method (4.4.1) converges if and only if $\rho(G) < 1$.

Proof:

It is required to prove that,

$$\lim_{k \to \infty} u^{(k+1)} = \lim_{k \to \infty} u^{(k)} = u = A^{-1}b,$$  \hspace{1cm} (4.4.4)

where $u$ is the exact solution which must satisfy,

$$u = Gu + r.$$  \hspace{1cm} (4.4.5)

If we subtract (4.4.5) from (4.4.1), we obtain

$$u^{(k+1)} - u = G(u^{(k)} - u).$$  \hspace{1cm} (4.4.6)

By denoting $e^{(k+1)} = u^{(k+1)} - u$, equation (4.4.6) becomes,

$$e^{(k+1)} = Ge^{(k)} = G^2 e^{(k-1)}$$
$$\vdots$$
$$= G^k e^{(0)}.$$  \hspace{1cm} (4.4.7)

For convergence, we require,

$$\lim_{k \to \infty} e^{(k+1)} = 0.$$  \hspace{1cm} (4.4.8)

Since in the r.h.s. of equation (4.4.7), $e^{(0)}$ is arbitrary, then we require,

$$\lim_{k \to \infty} G^k = 0,$$  \hspace{1cm} (4.4.8)

and by Theorem (2.8.2), the proof follows. (See Burden, 1978, for another proof).

The following theorems deal with the convergence of the basic iterative methods described above.
Theorem 4.4.2

If $A$ is diagonally dominant, then $A$ is non-singular and the sequence $\{u^{(k)}\}$, $k = 1, 2, \ldots$ defined by the Jacobi method (4.3.4) converges for any initial guess $u^{(0)}$.

Theorem 4.4.3

If $A$ is symmetric and positive-definite, then the sequence $\{u^{(k)}\}$, $k = 1, 2, \ldots$, defined by the Gauss-Seidel method (4.3.7) converges for any initial guess $u^{(0)}$.

For the proofs of the preceding two theorems (see, Johnson and Riess, 1982).

Theorem 4.4.4 (Stein-Rosenberg)

For the matrix $A$ in the system (4.4.3), if $a_{ij} \leq 0$, for each $i \neq j$ and $a_{ii} > 0$ for each $i = 1, 2, \ldots, n$ then one and only one of the following statements holds:

(a) $0 < \rho(L) < \rho(B) < 1$,
(b) $1 < \rho(B) < \rho(L)$,
(c) $\rho(B) = \rho(L) = 0$,
(d) $\rho(B) = \rho(L) = 1$.

We can see from Theorem (4.4.4) that the Jacobi method and the GS method are either both divergent or both convergent and in the latter case the GS method converges faster than the Jacobi method. This result can be expressed by the following corollary:

Corollary:

If the non-negative Jacobi iteration matrix $B = L + U$ is such that $0 < \rho(B) \leq 1$, then

$$R(L) > R(B),$$

where $R$ is the rate of convergence defined in (4.5.8).
Theorem 4.4.5

If $A$ is symmetric and $a_{ii} > 0$, $i = 1, 2, \ldots, n$, then the SOR method (4.3.10) converges iff $A$ is positive definite and $0 < \omega < 2$.

(See Young, 1971, or Varga, 1962, for proof.)

Theorem 4.4.6

For $A$, if $a_{ii} \neq 0$, $i = 1, 2, \ldots, n$, then

$$\rho(L_\omega) > |\omega - 1|.$$ (4.4.11)

Moreover, if $\rho(L_\omega) < 1$ then,

$$0 < \omega < 2.$$ (4.4.12)

(See Varga, 1962, for proof.)

4.5 RATE OF CONVERGENCE

The efficiency of an iterative method depends on the computational work required per iteration and on the number of iterations required for convergence. To estimate the number of iterations, we require the norm of the error vector $e^{(n)}$ to be reduced to some prespecified factor $\varepsilon$.

From (4.4.7), we have,

$$\| e^{(k)} \| \leq \| G^k \| \| e^{(0)} \|.$$ (4.5.1)

Then, if $e^{(0)} \neq 0$, we have,

$$\frac{\| e^{(k)} \|}{\| e^{(0)} \|} \leq \| G^k \|.$$ (4.5.2)

Since it is assumed that the method (4.4.1) is convergent, we require,

$$\| e^{(k)} \| \leq \varepsilon \| e^{(0)} \|,$$ (4.5.3)
and we choose \( k \) sufficiently large such that

\[
\|G^k\| < \epsilon
\]  \hspace{1cm} (4.5.4)

and if \( k \) is large enough so that \( \|G^k\| < 1 \), it follows that (4.5.4) is equivalent to

\[
k > - \log \epsilon \left( - \frac{1}{k} \log \|G^k\| \right).
\]  \hspace{1cm} (4.5.5)

This is the lower bound for the number of iterations required for the iterative method (4.4.1) to converge to the specified tolerance.

**Definition 4.5.1**

Let

\[
R_k(G) = - \frac{1}{k} \log \|G^k\|.
\]  \hspace{1cm} (4.5.6)

We call \( R_k(G) \) the *average rate of convergence* of the iterative method (4.4.1) for \( k \) iterations. It can be proved (Young (1971), Varga (1962)) that,

\[
\rho(G) = \lim_{k \to \infty} \left( \|G^k\| \right)^{1/k}.
\]  \hspace{1cm} (4.5.7)

Hence we have the following definition.

**Definition 4.5.2**

We define the *asymptotic average rate of convergence* (Young, 1954) or simply the *rate of convergence* by

\[
R(G) = \lim_{k \to \infty} R_k(G) = - \log \rho(G).
\]  \hspace{1cm} (4.5.8)

Thus we can say that the larger the rate of convergence is, the faster the iterative method converges.
4.6 DETERMINATION OF THE OPTIMAL PARAMETER $\omega$ AND COMPARISON OF RATES OF CONVERGENCE

It is important to determine an optimal or a suitable value of the relaxation parameter $\omega$ of the S.O.R. method, which will minimise the spectral radius and hence maximise the rate of convergence.

Before we determine the theoretical optimum of $\omega$, we will establish a relationship between the eigenvalues of the Jacobi and SOR iteration matrices, $B$ and $L_\omega$ respectively.

Suppose we have the linear system of equations,

$$Au = b', \quad (4.6.1)$$

using $A = D - L' - U' = D(I-L-U)$. Equation (4.6.1) may be written in the form,

$$(I-L-U)u = b, \quad (4.6.2)$$

where $I$ is the identity matrix, $L$ and $U$ are as described previously, and $b = D^{-1}b'$. This is called a scaling operation.

Then the SOR iteration matrix has the form,

$$(I-\omega L)^{-1} [\omega U + (1-\omega)I], \quad (4.6.3)$$

and the Jacobi iteration matrix is,

$$(L+U).$$

To find the eigenvalues $\lambda$'s of the SOR iteration matrix, we have to solve the characteristic equation,

$$|(I-\omega L)^{-1}[\omega U + (1-\omega)I] - \lambda I| = 0$$

or

$$|(I-\omega L)^{-1}[I + \omega(U-I)] - \lambda I| = 0$$

or

$$|I + \omega(U-I) - \lambda(I-\omega L)| = 0$$

$$|\omega^{-1}I + U - I - \lambda\omega^{-1}I + \lambda L| = 0$$
\[ |(\lambda L + U) - \frac{\lambda + \omega - 1}{\omega} I| = 0 \]
\[ |\lambda^{\frac{1}{2}}(\lambda^{\frac{1}{2}} L + \lambda^{-\frac{1}{2}} U) - \frac{\lambda + \omega - 1}{\lambda^{\frac{1}{2}} \omega} I| = 0 \]

or
\[ |(\lambda^{\frac{1}{2}} L + \lambda^{-\frac{1}{2}} U) - \frac{\lambda + \omega - 1}{\lambda^{\frac{1}{2}} \omega} I| = 0 \] (4.6.4)

Now, if \( I-L-U \) is a two cyclic matrix and is consistently ordered, then the eigenvalues of \( \lambda^{\frac{1}{2}} L + \lambda^{-\frac{1}{2}} U \) are independent of \( \lambda \) and \( \lambda^{\frac{1}{2}} L + \lambda^{-\frac{1}{2}} U \) can be replaced by \( L+U \), i.e. equation (4.6.4) thus becomes,

\[ |(L+U) - \frac{\lambda + \omega - 1}{\lambda^{\frac{1}{2}} \omega} I| = 0. \] (4.6.5)

It is clear that the quotient \( \left[ \frac{\lambda + \omega - 1}{\lambda^{\frac{1}{2}} \omega} \right] \) represents the eigenvalue of the Jacobi iteration matrix \( L+U \) and we call it \( \mu \), thus, i.e.,

\[ \mu = \frac{\lambda + \omega - 1}{\lambda^{\frac{1}{2}} \omega}. \] (4.6.6)

Now to find a formula for the optimal relaxation parameter \( \omega \), we rewrite equation (4.6.6) in the form

\[ \mu \lambda^{\frac{1}{2}} = \frac{\lambda + \omega - 1}{\omega} = \frac{1}{\omega} \lambda + \left( \frac{\omega - 1}{\omega} \right). \] (4.6.7)

We define two functions of \( \lambda \) such that

\[ f_1(\lambda) = \frac{1}{\omega} \lambda + \frac{\omega - 1}{\omega} \]

and

\[ f_2(\lambda) = \mu \lambda^{\frac{1}{2}} \]

which are represented graphically in Fig. (4.6.1).
As shown in Fig. (4.6.1) $f_1(\lambda)$ is a straight line passing through the point $(1,1)$ and its slope $1/\omega$ decreases as $\omega$ increases, $f_2(\lambda)$ is a parabola. The largest abscissa of the two points of intersection of $f_1(\lambda)$ and $f_2(\lambda)$ decreases with increasing $\omega$ (decreasing the slope of $f_1(\lambda)$) until the line is a tangent to the parabola at the point C.

The two points of intersection A and B are given by,

$$\lambda^2 + 2\lambda \left[ (\omega - 1) - \frac{1}{2} \mu \omega^2 \right] + (\omega - 1)^2 = 0,$$  \hspace{1cm} (4.6.8)

which is quadratic in $\lambda$, so,

$$\lambda = \frac{1}{2} \mu \omega^2 - (\omega - 1) \pm \mu \omega \left[ \frac{1}{4} \mu \omega^2 - (\omega - 1)^2 \right]^{\frac{1}{2}}.$$  \hspace{1cm} (4.6.9)

At the tangency point C, $\lambda$ has two equal roots, so, equation (4.6.9) reduces to,

$$\frac{1}{4} \mu \omega^2 - (\omega - 1) = 0$$

or
\[ \omega = \frac{1 \pm (1-\mu^2)^{1/2}}{\mu^2} \]

\[ = \frac{2}{1 \pm (1-\mu^2)^{1/2}} . \quad (4.6.10) \]

With the negative sign, \( 0 \leq \mu \leq 1 \), gives the range \( \omega \geq 2 \), and with the positive sign, \( 0 \leq \mu \leq 1 \) gives the range \( 1 \leq \omega \leq 2 \). Now the range of \( \omega \) must include \( \omega = 1 \), so we choose the positive sign, to give the formula, i.e.,

\[ \omega_b = \omega_{\text{optimal}} = \frac{2}{1 + \sqrt{1-\mu^2}} , \quad (4.6.11) \]

where, \( \mu \) denotes the largest eigenvalue of \( B \). The value of \( \omega \) which minimises \( \rho(L_\omega) \), so that,

\[ \rho(L_{\omega_b}) < \rho(L_\omega) , \quad (4.6.12) \]

is given by (4.6.11) and the minimum value of the spectral radius \( \rho(L_\omega) \) can also be shown, (Mitchel and Griffiths, 1980), as,

\[ \rho(L_{\omega_b}) = \omega_b - 1 = \frac{1 - \sqrt{1-\mu^2}}{1 + \sqrt{1-\mu^2}} . \quad (4.6.13) \]

For \( \omega = 1 \), equation (4.6.6) gives,

\[ \rho(L) = \left[ \rho(B) \right]^2 = (\mu)^2 , \quad (4.6.14) \]

where \( L \) is the G.S. iteration matrix.

For Poisson's equation over a rectangle of sides \( ph \) and \( qh \) with Dirichlet boundary conditions, it is proved (see Lapidus, 1982, Smith, 1978) that \( \rho(B) \) for the five point difference approximation using a square mesh of side \( h \), is

\[ \rho(B) = \frac{1}{2} \left[ \cos \frac{\pi}{p} + \cos \frac{\pi}{q} \right] . \quad (4.6.15) \]
Suppose the problem is defined in the unit square as follows,

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

$$u(x,y) = g(x,y), \quad (x,y) \in C$$

(4.6.16)

where $D$ is the interior domain and $C$ is the boundary such that $D \cup C$ is the unit square,

$$D \cup C = \{(x,y) : 0 \leq x \leq 1, \ 0 \leq y \leq 1\} \ .$$

Also suppose we have equal mesh sizes, i.e. $h = \frac{1}{p} = \frac{1}{q}$, then equation (4.6.15) becomes,

$$\rho(B) = \cos \pi h .$$

(4.6.17)

And hence equation (4.6.11) and (4.6.13) become,

$$\omega_b = \frac{2}{1 + \sin \pi h}$$

(4.6.18)

and

$$\rho(L_{\omega_b}) = \omega_b - 1 = \frac{1 - \sin \pi h}{1 + \sin \pi h} .$$

(4.6.19)

In this case, if we choose $h = \frac{1}{20}$ then $\rho(B) = 0.98769$. In Figure (4.6.1) we investigate the behaviour of $\rho(L_{\omega})$ as a function of $\omega$. As $\omega$ increases from 0 to 1, $\rho(L_{\omega})$ decreases very slowly from $1 - \omega^2 = 0.9755$. As $\omega$ increases further $\rho(L_{\omega})$ decreases slightly more rapidly until $\omega$ gets close to $\omega_b = 1.7295$, at which point $\rho(L_{\omega}) = \omega_b - 1 = 0.7295$. As $\omega$ increases further, $\rho(L_{\omega})$ increases linearly to $\rho(L_{\omega}) = 1$ when $\omega = 2$. 
NOW, for the SSOR method (see Colgan, 1984) we have the following relations,

$$\omega_b = \frac{2}{1 + \sqrt{2(1-\mu)}}$$  \hspace{1cm} (4.6.20)

and

$$\rho(G_{SSOR}) \leq \frac{1 - \sqrt{(1-\mu)/2}}{1 + \sqrt{(1-\mu)/2}}$$  \hspace{1cm} (4.6.21)

By using (4.6.17) we obtain,

$$\omega_b = \frac{2}{1 + 2\sin \frac{\pi h}{2}}$$  \hspace{1cm} (4.6.22)

and

$$\rho(G_{SSOR}) \leq \frac{1 - \sin \pi h/2}{1 + \sin \pi h/2}$$  \hspace{1cm} (4.6.23)

We can see that the SOR method converges about twice as quickly as the SSOR method.
4.7 SEMI-ITERATIVE METHODS AND CHEBYSHEV POLYNOMIALS

A semi-iterative method (SI method) is a nonstationary method associated with a linear stationary iterative method of first degree.

The construction of SI methods is analogous to certain summability methods for accelerating the convergence of sequences of real numbers (see Young, 1971). Consider the linear system,

\[ Au = b , \quad (4.7.1) \]

which results from using a finite difference approximation to an elliptic partial differential equation and where the matrix A is nonsingular. Given the sequence \( u^{(0)}, u^{(1)}, u^{(2)}, \ldots \) obtained by the completely consistent linear stationary iterative method,

\[ u^{(k+1)} = Gu^{(k)} + r \quad (4.7.2) \]

where,

\[ r = (I-G)A^{-1}b . \]

By the method of summability, we can often develop another sequence \( v^{(1)}, v^{(2)}, v^{(3)}, \ldots \) so that either the new sequence converges when the old one does not or else the new one converges faster than the old one if the old one converges.

We now define the vector sequence determined by the linear combination,

\[ v^{(k)} = \sum_{\ell=0}^{k} a_{k, \ell} u^{(\ell)} , \quad k = 0,1,2,\ldots \quad (4.7.3) \]

where the coefficients \( a_{k, \ell} \) are real with the restriction that,

\[ \sum_{\ell=0}^{k} a_{k, \ell} = 1 , \quad k = 0,1,2,\ldots \quad (4.7.4) \]

also

\[ v^{(0)} = u^{(0)} . \]

The combined procedure of (4.7.2) and (4.7.3) is said to be a
semi-iterative method (Varga, 1962) or a polynomial acceleration method applied to the basic method (4.7.2).

To analyze the convergence of (4.7.3), we define the error vectors of the $k$th iterate as,

$$e_{(k)} = u_{(k)} - u,$$  \hspace{1cm} (4.7.5)

and

$$e_{(k)} = v_{(k)} - u,$$ \hspace{1cm} (4.7.6)

where $u$ is the exact solution of (4.7.1). From equation (4.7.3), equation (4.7.6) becomes,

$$e_{(k)} = \sum_{l=0}^{k} a_{k,l} e_{(l)} - u$$

or

$$e_{(k)} = \sum_{l=0}^{k} a_{k,l} e_{(l)} .$$ \hspace{1cm} (4.7.7)

since the method (4.7.2) is consistent, then,

$$e_{(l)} = G^l e_{(0)} ,$$

hence

$$e_{(k)} = \sum_{l=0}^{k} a_{k,l} G^l e_{(0)} .$$ \hspace{1cm} (4.7.8)

Now, we define the polynomial,

$$P_k(G) = \sum_{l=0}^{k} a_{k,l} G^l .$$ \hspace{1cm} (4.7.9)

Hence, (4.7.8) can be written as,

$$e_{(k)} = P_k(G) e_{(0)} = P_k(G) e_{(0)} .$$ \hspace{1cm} (4.7.10)
Let us suppose that the real eigenvalues $\mu$ of $G$ of the iterative method (4.7.2) lie in the range,

$$m \leq \mu \leq M < 1,$$ 

(4.7.11)

where $m$ and $M$ are real numbers, then from equation (4.7.10), we have,

$$\| \epsilon^{(k)} \| \leq \| P_k(G) \| \| \epsilon^{(0)} \| ,$$

(4.7.12)

and since (Varga, 1962, p. 135):

$$\| P_k(G) \| = \rho (P_k(G)) = \max_{m \leq \mu \leq M} | P_k(\mu) |$$

we now need to minimise

$$\max_{m \leq \mu \leq M} | P_k(\mu) |$$

(4.7.13)

where $P_k(1) = 1$, for all $k$.

Let us introduce a new variable $w = w(\mu)$ defined by,

$$w(\mu) = \frac{2\mu - (m+M)}{M-m}$$

(4.7.14)

which involves that $w(m) = -1$ and $w(M) = 1$, hence $-1 \leq w \leq 1$.

Also if we let,

$$z = w(1) = \frac{2 - (m+M)}{M-m}$$

(4.7.15)

then

$$z > 1.$$ 

(4.7.16)

We now define the polynomial $Q_k(w)$ by,

$$Q_k(w) = P_k \left[ \frac{(M-m)w + (M+m)}{2} \right]$$

(4.7.17)

$$= P_k(\mu) \quad \text{(From (4.7.14))}.$$ 

Therefore,

$$\max_{m \leq \mu \leq M} | P_k(\mu) | = \max_{-1 \leq w \leq 1} | Q(w) |.$$ 

(4.7.18)
The problem now is to find a polynomial \( Q_k(w) \) of degree \( k \) or less such that \( Q_k(z) = P_k(1) = 1 \) and \( \max_{-1 \leq w \leq 1} |Q_k(w)| \) is minimized.

Now we define the Chebyshev polynomials as,

\[
\begin{align*}
T_0(x) &= 1, \quad T_1(x) = x \\
T_{k+1}(x) &= 2xT_k(x) - T_{k-1}(x), \quad k \geq 1
\end{align*}
\]

(4.7.19)

The following theorem gives some fundamental properties of the Chebyshev polynomials that we shall use.

**Theorem 4.7.1**

Let \( k \) be a fixed integer and let \( d \) be any fixed real number such that \( z > 1 \). If we let,

\[
H_k(w) = \frac{T_k(w)}{T_k(d)}
\]

(4.7.20)

where \( T_k(w) \) and \( T_k(d) \) are the Chebyshev polynomials (4.7.19), then

\[
H_k(d) = 1
\]

(4.7.21)

and

\[
\max_{-1 \leq w \leq 1} |H_k(w)| = \frac{1}{T_k(d)}
\]

(4.7.22)

Moreover, if \( Q(w) \) is any polynomial of degree \( k \) or less such that \( Q(d) = 1 \) and

\[
\max_{-1 \leq w \leq 1} |Q(w)| < \max_{-1 \leq w \leq 1} |H_k(w)|,
\]

(4.7.23)

then

\[
Q(w) = H_k(w).
\]

**Proof:** See Flanders and Shortley, (1950).

By Theorem (4.7.1) we can define,

\[
Q_k(w) = \frac{T_k(w)}{T_k(z)}.
\]

(4.7.24)
Moreover,
\[ \max_{-1 \leq \omega \leq 1} |Q_k(\omega)| = \frac{1}{T_k(z)} = \frac{1}{T_k \left( \frac{2-(m+M)}{M-m} \right)}. \] (4.7.25)

Therefore, we have,
\[ P_k(\mu) = Q_k \begin{bmatrix} 2\mu - (m+M) \\ \frac{M-m}{M-m} \end{bmatrix} \]
\[ = \frac{T_k \left[ \frac{2\mu - (M+m)}{M-m} \right]}{T_k \left[ \frac{2-(m+M)}{M-m} \right]}. \] (4.7.26)

Since \( z > 1 \) and \( T_k(z) > 1 \), then the method is convergent even if the basic iterative method (4.7.2) does not converge. Now for the semi-iterative method (4.7.3), it will be desirable to develop a relation between the vectors \( \mathbf{v}(k) \), \( k = 0, 1, 2, ... \) such that \( \mathbf{v}(k) \) is directly computed without the predetermination of \( \mathbf{u}(k) \). Here we give the final form of a second degree iteration method for \( \mu > 1 \),
\[ \mathbf{v}(k+1) = 2 \begin{bmatrix} \frac{2}{M-m} & \left( \frac{m+M}{M-m} \right) I \end{bmatrix} \frac{T_k(z)}{T_{k+1}(z)} \mathbf{v}(k) - \frac{T_{k-1}(z)}{T_{k+1}(z)} \mathbf{v}(k-1) + \frac{4}{M-m} \cdot \frac{T_k(z)}{T_{k+1}(z)} \mathbf{v}(k) \] (4.7.27)

4.8 CHEBYSHEV ACCELERATION

Suppose we want to solve the linear system
\[ A\mathbf{u} = b, \] (4.8.1)
by using a basic iterative method of the form,
\[ \mathbf{u}(k+1) = G\mathbf{u}(k) + \mathbf{r}, \quad k = 0, 1, \ldots \] (4.8.2)

Let \( m = m(G) \) and \( M = M(G) \) be, respectively, the smallest and the largest eigenvalues of the iteration matrix \( G \). We assume also that
the matrix $G$ is symmetrizable in that it is similar to a symmetric matrix $W$.

We now consider the computational aspects of polynomial acceleration using $P_k(u)$ as defined in (4.7.26), and (4.7.19) when applied to the basic method (4.8.2). It can be shown (Hageman and Young, 1981) that the polynomials $P_k(x)$ satisfy the recurrence relation,

$$
P_0(x) = 1
$$

$$
P_1(x) = \gamma x - \gamma + 1,
$$

$$
P_{k+1}(x) = \rho_{k+1}(\gamma x + 1 - \gamma)P_k(x) + (1 - \rho_{k+1})P_{k-1}(x)
$$

where $\gamma = 2(2 - M - m)$

$$
\rho_{k+1} = 2w(1)T_k(w(1))/T_{k+1}(w(1)).
$$

It follows from Theorem (4.7.1) that the iterates for the polynomial procedure based on $P_k(x)$ may be obtained by (see Hageman and Young, 1981),

$$
u^{(k+1)} = \rho_{k+1}\left\{\gamma \left\{G_u^{(k)} + r\right\}\right\} + (1 - \gamma)u^k + (1 - \rho_{k+1})u^{k-1}.
$$

The procedure (4.8.6) is known as the Chebyshev acceleration method.

If we use the exact values of $m$ and $M$ then we have the optimal Chebyshev acceleration method. The parameters $\rho_{k+1}$ of (4.8.5) can be written, using the Chebyshev polynomial (4.7.19) in the computational form:

$$
\rho_1 = 1,
\rho_2 = \left[1 - \frac{\sigma^2}{2}\right]^{-1}
$$

$$
\rho_{k+1} = \left[1 - \frac{\sigma^2}{4\rho_k}\right]^{-1}, \quad k \geq 2.
$$
where
\[ \sigma = 1/\omega(1) = \frac{M-m}{2-M-m}. \] (4.8.8)

It can be shown (Varga, 1962) that,
\[ \lim_{k \to \infty} \rho_k = \rho_\infty = \frac{2}{1 + \sqrt{1-\sigma^2}}. \] (4.8.9)

The Chebyshev acceleration procedure applied to a basic iterative method consists of the recurrence relation (4.8.6) and the parameters (4.8.4), (4.8.8) and (4.8.7). One disadvantage of the method is that the storage of two iteration vectors is needed at each stage. A first degree form can be derived, but the acceleration parameters then become large which might make the iteration unstable because of rounding errors. In equation (4.8.6), the term \((G \bar{u}(k) + r)\) is actually the right hand side of equation (4.8.2) which is the basic iterative method, so this leaves the method (4.8.6) suitable for use in conjunction with any basic iterative method, i.e. changing the iteration matrix \(G\) in (4.8.6).

In Section 4.9 we will apply the Chebyshev acceleration procedure to the SSOR method where we can use the initial guess vector as \(\bar{u}(k-1)\) and the iteration vector obtained from the first half iteration as \(\bar{u}(k)\) then the one from the second half (complete SSOR) for the term \(G \bar{u}(k) + r\). When we obtain the new iterate \(\bar{u}(k+1)\) we take it as \(\bar{u}(k-1)\) in the next iteration and so on until convergence is achieved.
4.9 MILDLY NONLINEAR ELLIPTIC EQUATIONS

4.9.1 Formulation and Method of Solution

We recall that the general second order p.d.e. (1.1.1) can be written in the form,
\[ a \frac{\partial^2 u}{\partial x^2} + 2b \frac{\partial^2 u}{\partial x \partial y} + c \frac{\partial^2 u}{\partial y^2} = f(x,y,u), \tag{4.9.1} \]
where \( a, b \) and \( c \) are functions of \( \tilde{x} \) and \( \tilde{y} \), such that at each point of a finite connected two dimensional region \( R \),
\[ a^2 + b^2 + c^2 \neq 0, \tag{4.9.2} \]
then we say that equation (4.9.1) is mildly nonlinear. Also, we know that equation (4.9.1) is elliptic when, \( b^2 - ac < 0 \), it can be transformed to,
\[ a \frac{\partial^2 u}{\partial x^2} + \beta \frac{\partial^2 u}{\partial y^2} = f(x,y,u), \quad \alpha, \beta > 0, \tag{4.9.3} \]
(see W.F. Ames, 1965).

To solve equation (4.9.3) numerically we approximate the partial derivatives using the finite difference approximation procedure (see Sections 2.1 and 4.2). If \( \alpha = \beta = 1 \) and with mesh sizes equal (i.e. \( h = k \)), then equation (4.9.3) can be written in the form,
\[ u_{i+1,j} + u_{i-1,j} - 4u_{i,j} + u_{i,j+1} + u_{i,j-1} = h^2 f_{ij}, \tag{4.9.4} \]
where
\[ f_{ij} = f(x_i,y_j,u_{ij}), \quad i,j = 1,2,\ldots,n. \]
Equation (4.9.4) can be represented by the nonlinear system,
\[ Au = f(u). \tag{4.9.5} \]
To solve the system (4.9.5) we apply the inner-outer iteration
strategy where the outer iteration is Picard and the inner iteration is carried out by an iterative method of the form:

\[ u^{(k+1)} = G u^{(k)} + r. \]  \hspace{1cm} (4.9.6)

In the flowchart (Figure 4.9.1), \( u^{(i,j)} \) denotes the iterate at the \( i \)th inner iteration of the \( j \)th outer iteration, \( u^{(*,j)} \) denotes the iterate at the completion of the \( j \)th outer iteration. \( r^{(j)} \) denotes the constant vector of the inner iterations for the \( j \)th outer iteration. With reference to equation (4.1.5) we define the function \( g \) as,

\[ g(u) = (I-G)A^{-1} f(u). \]  \hspace{1cm} (4.9.7)

Here, \( r^{(j)} = g(u^{(*,j-1)}) \) which in practice is not calculated explicitly as in (4.9.7) nor in the simple explicit form indicated in box (3) of the flowchart, but it is computed implicitly, by the point iterative method derived directly from the five point finite difference equation (4.9.4) as explained later.

Two convergence criteria are used to terminate the inner and the outer iteration.

Usually, the mixed convergence test between two consecutive iterations is used, i.e.,

\[ \frac{|u_{i,j}^{(k+1)} - u_{i,j}^{(k)}|}{1 + |u_{i,j}^{(k)}|} < \varepsilon_\rho, \quad \rho = 1, 2 \]

for every \( i,j = 1,2,\ldots,n \).

We choose the convergence tolerance \( \varepsilon_1 \) for the inner iteration less accurate than \( \varepsilon_2 \) for the outer iteration, since the solution obtained by the inner solution is not necessarily the right one.
1. Initialize $u^{(*,j)}$
   $j \leftarrow 0$

2. $j \leftarrow j + 1$
   $i \leftarrow 0$
   $u^{(0,j)} \leftarrow u^{(*,j-1)}$
   $\Sigma^{(j)} \leftarrow g(u^{(*,j-1)})$

3. $i \leftarrow i + 1$
   $u^{(i,j)} \leftarrow u^{(i-1,j)} + \Sigma^{(j)}$

4. $|u^{(i,j)} - u^{(i-1,j)}| < \epsilon_1$
   NO

5. $|u^{(*,j)} - u^{(*,j-1)}| < \epsilon_2$
   YES
   NO

6. solution obtained
   STOP

FIGURE 4.9.1
An alternative approach for the inner iteration is to use a fixed number of inner iterations (say N) for each outer iteration. For our test problems, of Laplace type, N=3 would be a good choice.

4.9.2 Applications and Results

We restrict our experimental work to the two dimensional mildly nonlinear elliptic equations. They occur in a large variety of applied problems, for example the equation $\nabla^2 u = u^2$ is important in the study of subsonic flows and molecular interactions and $\nabla^2 u = e^u$ is important in radiation problems and certain aspects of electrohydrodynamics. We will apply the basic iterative methods described earlier in this chapter to the following problems, which are defined on a unit square region $\Omega$ with boundary $C$,

1. $\nabla^2 u = u^2$  \hspace{1cm} (4.9.8)
   with $u = 10$ on the boundary $C$

2. $\nabla^2 u = u^4$  \hspace{1cm} (4.9.9)
   with $u = 0$ on $C$

3. $\nabla^2 u = e^u$  \hspace{1cm} (4.9.10)
   with $u = 0$ on $C$

4. $\nabla^2 u = -\lambda^2 \sin u$, $\lambda = 1$,  \hspace{1cm} (4.9.11)
   with $u = 0$ on $C$.

We use the five point difference equation (4.9.4) with $b = f(u_{ij})$ as the r.h.s. unit to obtain the point-iterative methods for GS, SOR, SSOR and SSOR with Chebyshev acceleration methods which are implemented in FORTRAN 77 programs to solve the above problems.

The theoretical optimum relaxation parameter $\omega_b$ can be
calculated using the formulas,

$$\omega_b = \frac{2}{1 + \sin \pi h} \quad \text{for SOR method,}$$

or

$$\omega_b = \frac{2}{1 + 2\sin \frac{\pi h}{2}} \quad \text{for the SSOR method.}$$

The convergence criteria used in the numerical experiments is the average convergence test of any two successive iterates, i.e. the iteration (the inner iteration or the outer iteration) is continued until the following condition is satisfied:

$$\frac{|u^{(k+1)} - u^{(k)}|}{1 + |u^{(k)}|} < \varepsilon_\ell, \quad \ell = 1, 2 \quad (4.9.12)$$

where $\varepsilon_1 = 10^{-3}$ and $\varepsilon_2 = 5 \times 10^{-6}$ correspond to the inner iteration and the outer iteration respectively, $u^{(k)}$ denote the solution vector at the $k^{th}$ iterate, either of the inner loop or of the outer loop.

The experimental optimal value of the overrelaxation parameter $\omega_b$ is determined by solving the problem for a range of values of the overrelaxation parameter $\omega$ (obtained by the formula) and choosing the value which gives the lowest number of iterations. Each of the following tables shows the results for one of the four test problems under investigation. Also given are the graphs which correspond to these tables.

The curves are plotted between the total number of inner iterations versus $h^{-1}$ (where $h$ is the mesh size) in logarithmic scale. We notice that the slope of the regression line for the SOR method is nearly 1 except for problem 1 where it is less. For the
The GS method the slope is greater than 1 except for the first problem. The SSOR and the SSOR with Chebyshev acceleration have slopes less than 1.

Remark: For difficult problems like the first test problem, i.e. \( \varphi^2 u = u^2 \)

with \( u = 10 \) on the boundary, the GS method can be more efficient (in comparison with the SOR method) if the initial guess is close to the solution, i.e. a good guess is 7 which produces results near to the corresponding results given by the SOR, unlike when the initial guess is 1 and this is apparent in figure (4.9.1) where the slope of the regression line for the GS method is less than 1 unlike the cases of the other problems.
Table (4.9.1a) : $v^2 u = u^2$ with initial guess 7.

<p>| $1/h$ | GS method | | | SOR method | |
|---|---|---|---|---|---|---|---|---|
|   | Spectral Radius | Number of inner iterations | Number of outer iterations | Spectral Radius | $u_b$ | Number of inner iterations | Number of outer iterations |
| 10 | .90 | 290 | 37 | .52 | 1.53 | 1.52 | 181 | 30 |
| 20 | .98 | 521 | 110 | .70 | 1.73 | 1.7 | 284 | 37 |
| 30 | .989 | 560 | 119 | .78 | 1.81 | 1.78 | 362 | 47 |
| 40 | .994 | 751 | 360 | .81 | 1.85 | 1.81 | 425 | 56 |
| 50 | .996 | 986 | 615 | .722 | 1.88 | 1.722 | 514 | 73 |
| 60 | .997 | 706 | 307 | .87 | 1.90 | 1.87 | 518 | 72 |
| 80 | .998 | 1835 | 1379 | .87 | 1.92 | 1.87 | 582 | 94 |</p>
<table>
<thead>
<tr>
<th>$h^{-1}$</th>
<th>( \omega_b )</th>
<th>Number of iterations</th>
<th>( \omega_b )</th>
<th>Number of iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Theory</td>
<td>Experiment</td>
<td>Inner</td>
<td>Outer</td>
</tr>
<tr>
<td>10</td>
<td>1.52</td>
<td>1.58</td>
<td>127</td>
<td>18</td>
</tr>
<tr>
<td>20</td>
<td>1.73</td>
<td>1.75</td>
<td>181</td>
<td>19</td>
</tr>
<tr>
<td>30</td>
<td>1.81</td>
<td>1.82</td>
<td>231</td>
<td>29</td>
</tr>
<tr>
<td>40</td>
<td>1.85</td>
<td>1.88</td>
<td>268</td>
<td>29</td>
</tr>
<tr>
<td>50</td>
<td>1.88</td>
<td>1.89</td>
<td>296</td>
<td>34</td>
</tr>
<tr>
<td>60</td>
<td>1.90</td>
<td>1.91</td>
<td>331</td>
<td>48</td>
</tr>
<tr>
<td>80</td>
<td>1.92</td>
<td>1.94</td>
<td>373</td>
<td>49</td>
</tr>
</tbody>
</table>

Table (4.9.1b): \( \nabla^2 u = u^2 \) with initial guess 7.
<table>
<thead>
<tr>
<th>Method</th>
<th>GS Method</th>
<th>SOR Method</th>
<th>Number of Spectral Radius</th>
<th>Number of Inner Iterations</th>
<th>Number of Outer Iterations</th>
<th>Number of Spectral Radius</th>
<th>$\omega_b$</th>
<th>Number of Inner Iterations</th>
<th>Number of Outer Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$\omega_b$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Theory</td>
<td>Experim.</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>.9</td>
<td>59</td>
<td>54</td>
<td></td>
<td></td>
<td>.53</td>
<td>1.53</td>
<td>1.53</td>
<td>20</td>
</tr>
<tr>
<td>20</td>
<td>.98</td>
<td>178</td>
<td>174</td>
<td>.71</td>
<td>1.73</td>
<td>1.71</td>
<td>40</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>.989</td>
<td>327</td>
<td>323</td>
<td>.79</td>
<td>1.81</td>
<td>1.79</td>
<td>60</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>.994</td>
<td>487</td>
<td>483</td>
<td>.83</td>
<td>1.85</td>
<td>1.83</td>
<td>80</td>
<td>61</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>.996</td>
<td>647</td>
<td>643</td>
<td>.86</td>
<td>1.88</td>
<td>1.86</td>
<td>100</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>.997</td>
<td>798</td>
<td>794</td>
<td>.88</td>
<td>1.90</td>
<td>1.88</td>
<td>120</td>
<td>90</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>.998</td>
<td>1041</td>
<td>1037</td>
<td>.90</td>
<td>1.92</td>
<td>1.9</td>
<td>155</td>
<td>118</td>
<td></td>
</tr>
</tbody>
</table>

Table (4.9.2a): $\nabla^2 u = u^4$ with initial guess 0.01.
<table>
<thead>
<tr>
<th>Method</th>
<th>$\omega_b$</th>
<th>Number of iterations</th>
<th>$\omega_b$</th>
<th>Number of iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Theory</td>
<td>Experiment</td>
<td>Inner</td>
<td>Outer</td>
</tr>
<tr>
<td>$h^{-1}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1.52</td>
<td>1.55</td>
<td>17</td>
<td>13</td>
</tr>
<tr>
<td>20</td>
<td>1.73</td>
<td>1.8</td>
<td>29</td>
<td>25</td>
</tr>
<tr>
<td>30</td>
<td>1.81</td>
<td>1.87</td>
<td>41</td>
<td>37</td>
</tr>
<tr>
<td>40</td>
<td>1.85</td>
<td>1.89</td>
<td>52</td>
<td>48</td>
</tr>
<tr>
<td>50</td>
<td>1.88</td>
<td>1.92</td>
<td>62</td>
<td>59</td>
</tr>
<tr>
<td>60</td>
<td>1.90</td>
<td>1.94</td>
<td>71</td>
<td>68</td>
</tr>
<tr>
<td>80</td>
<td>1.92</td>
<td>1.95</td>
<td>89</td>
<td>86</td>
</tr>
</tbody>
</table>

Table (4.9.2b): $\nabla^2 u = u^4$ with initial guess 0.01.
<table>
<thead>
<tr>
<th>( h^{-1} )</th>
<th>GS method</th>
<th>SOR method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Spectral Radius</td>
<td>Number of inner iterations</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>.9</td>
<td>67</td>
</tr>
<tr>
<td>20</td>
<td>.98</td>
<td>223</td>
</tr>
<tr>
<td>30</td>
<td>.989</td>
<td>432</td>
</tr>
<tr>
<td>40</td>
<td>.994</td>
<td>679</td>
</tr>
<tr>
<td>50</td>
<td>.996</td>
<td>953</td>
</tr>
<tr>
<td>60</td>
<td>.997</td>
<td>1244</td>
</tr>
<tr>
<td>80</td>
<td>.998</td>
<td>1244</td>
</tr>
</tbody>
</table>

Table (4.9.3a): \( \nabla^2 u = e^u \) with initial guess -0.01.
<table>
<thead>
<tr>
<th>$h^{-1}$</th>
<th>Method</th>
<th>( \omega_b )</th>
<th># of iterations</th>
<th>Method</th>
<th>( \omega_b )</th>
<th># of iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Theory</td>
<td>Experiment</td>
<td>Inner</td>
<td>Outer</td>
<td>Theory</td>
</tr>
<tr>
<td>10</td>
<td>SSOR method</td>
<td>1.52</td>
<td>1.68</td>
<td>17</td>
<td>10</td>
<td>1.52</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>1.73</td>
<td>1.72</td>
<td>31</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30</td>
<td>1.81</td>
<td>1.84</td>
<td>49</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td></td>
<td>40</td>
<td>1.85</td>
<td>1.88</td>
<td>64</td>
<td>46</td>
</tr>
<tr>
<td></td>
<td></td>
<td>50</td>
<td>1.88</td>
<td>1.90</td>
<td>79</td>
<td>59</td>
</tr>
<tr>
<td></td>
<td></td>
<td>60</td>
<td>1.90</td>
<td>1.92</td>
<td>92</td>
<td>71</td>
</tr>
<tr>
<td></td>
<td></td>
<td>80</td>
<td>1.92</td>
<td>1.94</td>
<td>119</td>
<td>97</td>
</tr>
</tbody>
</table>

Table (4.9.3b): \( \nabla^2 u = e^u \) with initial guess -0.01.
<table>
<thead>
<tr>
<th>$h^{-1}$</th>
<th>GS method</th>
<th>SOR method</th>
<th>SOR method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Spectral Radius</td>
<td>Number of inner iterations</td>
<td>Number of outer iterations</td>
</tr>
<tr>
<td>10</td>
<td>.9</td>
<td>62</td>
<td>57</td>
</tr>
<tr>
<td>20</td>
<td>.98</td>
<td>186</td>
<td>182</td>
</tr>
<tr>
<td>30</td>
<td>.989</td>
<td>339</td>
<td>335</td>
</tr>
<tr>
<td>40</td>
<td>.994</td>
<td>504</td>
<td>500</td>
</tr>
<tr>
<td>50</td>
<td>.996</td>
<td>668</td>
<td>664</td>
</tr>
<tr>
<td>60</td>
<td>.997</td>
<td>821</td>
<td>817</td>
</tr>
<tr>
<td>80</td>
<td>.998</td>
<td>1062</td>
<td>1058</td>
</tr>
</tbody>
</table>

Table (4.9.4a): $\nabla^2 u = -\lambda^2 \sin u$, $\lambda = 1$, with initial guess $0.01$. 
| $h^{-1}$ | SSOR method | | SSOR with Cheby. Acc. | |
|---|---|---|---|---|---|
| | $\omega_b$ | Number of iterations | | $\omega_b$ | Number of iterations | |
| | Theory | Experiment | Inner | Outer | Theory | Experiment | Inner | Outer | |
| 10 | 1.52 | 1.6 | 18 | 15 | 1.52 | 1.55 | 11 | 8 | |
| 20 | 1.73 | 1.8 | 31 | 27 | 1.72 | 1.75 | 16 | 11 | |
| 30 | 1.81 | 1.86 | 43 | 39 | 1.81 | 1.85 | 17 | 11 | |
| 40 | 1.85 | 1.9 | 54 | 50 | 1.85 | 1.93 | 21 | 15 | |
| 50 | 1.88 | 1.92 | 64 | 61 | 1.88 | 1.95 | 24 | 17 | |
| 60 | 1.90 | 1.94 | 74 | 71 | 1.90 | 1.96 | 29 | 21 | |
| 80 | 1.92 | 1.95 | 93 | 90 | 1.92 | 1.97 | 33 | 24 | |

Table (4.9.4b): $\nabla^2 u = -\lambda^2 \sin u$, $\lambda = 1$ with initial guess 0.01.
Problem 1

Legend
- GS
- SOR
- SSOR
- SSOR & Cheby.

Figure (4.9.1): $\nabla^2 u = u^2$

Problem 2

Legend
- GS
- SOR
- SSOR
- SSOR & Cheby.

Figure (4.9.2): $\nabla^2 u = u^4$
Problem 3

Legend

- GS
- SOR
- SSOR
- SSOR & Cheby.

Figure (4.9.3): $\nabla^2 u = e^u$

Problem 4

Legend

- GS
- SOR
- SSOR
- SSOR & Cheby.

Figure (4.9.4): $\nabla^2 u = -\lambda^2 \sin(u), \lambda=1$
4.10 COMPUTATIONAL COMPLEXITY

In the previous section a comparison was made between the performance of the different methods when applied to the test problems in terms of the number of iterations required for convergence to a predetermined degree of accuracy. In this section we shall give a comparison between the methods by estimating the amount of computational work involved in terms of the arithmetic operations (i.e. multiplications and additions) which is performed to obtain the required solution. We will see that the methods with the smaller number of iterations are not necessarily the ones which require the least amount of computational work to solve the problem. In each case we assume that there are $n^2$ internal mesh points.

The G.S. Method

The solution of the test problems using the G.S method can be derived from (4.9.4) as,

$$u_{ij}^{(k+1)} = u_{ij}^{(k)} - \frac{h^2 b_{ij}}{2}$$

(4.10.1)

where $h^2 b_{ij} = h^2 f(i^{(m)}, j^{(m)}, u_{ij}^{(m)})$ are the right hand side (r.h.s.) elements of the system (4.9.5) at the $m$th outer iteration. Hence, the number of operations required to carry out the above iteration is,

$$n^2 \text{ multiplications} + 4n^2 \text{ additions} + 1 \text{ r.h.s. unit}$$

(4.10.2)

where the r.h.s. unit includes the operations to compute $h^2 b$ in the outer iteration loop.
The SOR Method

The solution using this method is given by,

\[ u_{ij}^{(k+1)} = (1-\omega)u_{ij}^{(k)} + u_{i+1,j}^{(k)} + u_{i-1,j}^{(k)} + u_{i,j+1}^{(k)} + u_{i,j-1}^{(k)} - h^2 b_{i,j} \times 0.25 \]

from which the computational complexity can be derived as,

2\( n^2 \) multiplications + 5\( n^2 \) additions + 1 r.h.s. unit \hspace{1cm} (4.10.3)

The SSOR Method

The computational complexity for the SSOR method can be derived directly from that of the SOR method since it involves two SOR sweeps applied in different directions, i.e.,

4\( n^2 \) multiplications + 10\( n^2 \) additions + 2 r.h.s. unit \hspace{1cm} (4.10.5)

The SSOR Method with Chebyshev Acceleration

The Chebyshev acceleration process can be applied to any basic iterative method \( u^{(k+1)} = Gu^{(k)} + r \) which can then have the form,

\[ u^{(k+1)} = \omega_{k+1} \left\{ \gamma(Gu^{(k)} + r) + (1-\gamma)u^{(k)} \right\} + (1-\omega_{k+1})u^{(k-1)} \]

(4.10.6)

where the sequence of the acceleration parameters \( \omega_k \) is defined by,

\[ \omega_1 = 1, \quad \omega_2 = \frac{1}{1 - \sigma^2 / 2}, \quad \omega_{k+1} = \frac{1}{1 - \sigma^2 / 4 \omega_k}, \quad \text{for } k \geq 2 \]

(4.10.7)

and \( \gamma, \sigma \) are constants.

We are concerned here with the SSOR method, hence \( G \) in (4.10.6) would be replaced by the SSOR iteration matrix \( G_{SSOR} \) as defined in.
(4.3.23). To implement the term \( G_{SSOR}\hat{u}^{(k)} + r \), which is the SSOR iteration, we need the computational complexity of the SSOR method, (4.10.5).

The overall computational complexity for a complete SSOR and Chebyshev iteration given by (4.10.6) and (4.10.7) can be derived as,

\[
10n^2 \text{ multiplications} + 13n^2 \text{ additions} + 1 \text{ r.h.s. unit}. \tag{4.10.8}
\]

For all the methods described above, the computational complexity of the r.h.s. unit can be estimated by multiplying the number of the outer definitions by the number of operations (multiplications and additions) in the right hand side (r.h.s. unit).

The following tables contain the computational complexity of the methods applied to the test problems where \( M \) and \( A \) denote the multiplication and addition operations respectively. We notice from the tables that for problem 1 the GS method is almost as good as the SOR method at some values of \( h^{-1} \), both methods are better (less computational work) than the SSOR method which is itself better than the SSOR with Chebyshev acceleration.
<table>
<thead>
<tr>
<th>$h^{-1}$</th>
<th>GS method</th>
<th>SOR method</th>
<th>SSOR method</th>
<th>SSOR with Cheby.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$M$</td>
<td>$A$</td>
<td>$M$</td>
<td>$A$</td>
</tr>
<tr>
<td>10</td>
<td>$290n^2$</td>
<td>$1160n^2$</td>
<td>$362n^2$</td>
<td>$905n^2$</td>
</tr>
<tr>
<td></td>
<td>$1010n^2$</td>
<td>$1313n^2$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>$521n^2$</td>
<td>$2084n^2$</td>
<td>$568n^2$</td>
<td>$1420n^2$</td>
</tr>
<tr>
<td></td>
<td>$1290n^2$</td>
<td>$1677n^2$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>$560n^2$</td>
<td>$2240n^2$</td>
<td>$724n^2$</td>
<td>$1810n^2$</td>
</tr>
<tr>
<td></td>
<td>$1440n^2$</td>
<td>$1872n^2$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>$751n^2$</td>
<td>$3004n^2$</td>
<td>$850n^2$</td>
<td>$2125n^2$</td>
</tr>
<tr>
<td></td>
<td>$1680n^2$</td>
<td>$2184n^2$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>$986n^2$</td>
<td>$3944n^2$</td>
<td>$1028n^2$</td>
<td>$2570n^2$</td>
</tr>
<tr>
<td></td>
<td>$1790n^2$</td>
<td>$2327n^2$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>$706n^2$</td>
<td>$2824n^2$</td>
<td>$1036n^2$</td>
<td>$2590n^2$</td>
</tr>
<tr>
<td></td>
<td>$1990n^2$</td>
<td>$2587n^2$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>$1835n^2$</td>
<td>$7340n^2$</td>
<td>$1164n^2$</td>
<td>$2910n^2$</td>
</tr>
<tr>
<td></td>
<td>$2340n^2$</td>
<td>$3042n^2$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Computational Complexity for $\nabla^2 u = u^2$.

Table (4.10.1)
<table>
<thead>
<tr>
<th>( h^{-1} )</th>
<th>GS method</th>
<th>SOR method</th>
<th>SSOR method</th>
<th>SSOR with Cheby.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M</td>
<td>A</td>
<td>M</td>
<td>A</td>
</tr>
<tr>
<td>10</td>
<td>59 ( n^2 )</td>
<td>236 ( n^2 )</td>
<td>40 ( n^2 )</td>
<td>100 ( n^2 )</td>
</tr>
<tr>
<td>20</td>
<td>178 ( n^2 )</td>
<td>712 ( n^2 )</td>
<td>80 ( n^2 )</td>
<td>200 ( n^2 )</td>
</tr>
<tr>
<td>30</td>
<td>327 ( n^2 )</td>
<td>1308 ( n^2 )</td>
<td>120 ( n^2 )</td>
<td>300 ( n^2 )</td>
</tr>
<tr>
<td>40</td>
<td>487 ( n^2 )</td>
<td>1948 ( n^2 )</td>
<td>160 ( n^2 )</td>
<td>400 ( n^2 )</td>
</tr>
<tr>
<td>50</td>
<td>647 ( n^2 )</td>
<td>2588 ( n^2 )</td>
<td>200 ( n^2 )</td>
<td>500 ( n^2 )</td>
</tr>
<tr>
<td>60</td>
<td>798 ( n^2 )</td>
<td>3192 ( n^2 )</td>
<td>240 ( n^2 )</td>
<td>600 ( n^2 )</td>
</tr>
<tr>
<td>80</td>
<td>1041 ( n^2 )</td>
<td>4164 ( n^2 )</td>
<td>310 ( n^2 )</td>
<td>775 ( n^2 )</td>
</tr>
</tbody>
</table>

Computational Complexity for \( v^2u = u^4 \)

Table (4.10.2)
<table>
<thead>
<tr>
<th>$h^{-1}$</th>
<th>GS method</th>
<th>SOR method</th>
<th>SSOR method</th>
<th>SSOR with Cheby.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$M$</td>
<td>$A$</td>
<td>$M$</td>
<td>$A$</td>
</tr>
<tr>
<td>10</td>
<td>$67n^2$</td>
<td>$268n^2$</td>
<td>$50n^2$</td>
<td>$125n^2$</td>
</tr>
<tr>
<td>20</td>
<td>$223n^2$</td>
<td>$892n^2$</td>
<td>$86n^2$</td>
<td>$215n^2$</td>
</tr>
<tr>
<td>30</td>
<td>$432n^2$</td>
<td>$1728n^2$</td>
<td>$120n^2$</td>
<td>$300n^2$</td>
</tr>
<tr>
<td>40</td>
<td>$679n^2$</td>
<td>$2716n^2$</td>
<td>$160n^2$</td>
<td>$400n^2$</td>
</tr>
<tr>
<td>50</td>
<td>$953n^2$</td>
<td>$3812n^2$</td>
<td>$202n^2$</td>
<td>$505n^2$</td>
</tr>
<tr>
<td>60</td>
<td>$1244n^2$</td>
<td>$4976n^2$</td>
<td>$242n^2$</td>
<td>$605n^2$</td>
</tr>
<tr>
<td>80</td>
<td>No. Conv.</td>
<td>$320n^2$</td>
<td>$800n^2$</td>
<td>$476n^2$</td>
</tr>
</tbody>
</table>

Computational Complexity for $\nabla^2 u = e^u$.

Table (4.10.3)
<table>
<thead>
<tr>
<th>h^{-1}</th>
<th>GS method</th>
<th>SOR method</th>
<th>SSOR method</th>
<th>SSOR with Cheby.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M</td>
<td>A</td>
<td>M</td>
<td>A</td>
</tr>
<tr>
<td>10</td>
<td>62 n^2</td>
<td>248 n^2</td>
<td>40 n^2</td>
<td>100 n^2</td>
</tr>
<tr>
<td>20</td>
<td>186 n^2</td>
<td>744 n^2</td>
<td>82 n^2</td>
<td>205 n^2</td>
</tr>
<tr>
<td>30</td>
<td>339 n^2</td>
<td>1356 n^2</td>
<td>122 n^2</td>
<td>305 n^2</td>
</tr>
<tr>
<td>40</td>
<td>504 n^2</td>
<td>2016 n^2</td>
<td>162 n^2</td>
<td>405 n^2</td>
</tr>
<tr>
<td>50</td>
<td>668 n^2</td>
<td>2672 n^2</td>
<td>200 n^2</td>
<td>500 n^2</td>
</tr>
<tr>
<td>60</td>
<td>821 n^2</td>
<td>3284 n^2</td>
<td>240 n^2</td>
<td>600 n^2</td>
</tr>
<tr>
<td>80</td>
<td>1062 n^2</td>
<td>4248 n^2</td>
<td>320 n^2</td>
<td>800 n^2</td>
</tr>
</tbody>
</table>

Computational Complexity for \( V^2 u = -\lambda^2 \sin u, \lambda = 1. \)

Table (4.10.4)
Chapter Five

EXTRAPOLATED ITERATIVE METHODS FOR NONLINEAR PROBLEMS

5.1 Introduction

5.2 Extrapolated Linear First Order Iterative Methods

5.3 Theorems on Convergence for Linear Equations.

5.4 Accelerating the SOR Method
   5.4.1 Extrapolated SOR Method
   5.4.2 Accelerated Overrelaxation Method
   5.4.3 Two-Parametric Method

5.5 Mildly Nonlinear Problems and Application

5.6 Newton - SOR Method for Nonlinear Equations

5.7 Experimental Results
5.1 **INTRODUCTION**

Consider the system of linear equations

$$Au = b$$ \hspace{1cm} (5.1.1)

which results from applying finite difference methods to approximate a given elliptic p.d.e., where $A$ is non-singular $(n \times n)$ real matrix, $u$ unknown vector $(n \times 1)$ and $b$ a constant vector $(n \times 1)$.

The aim of this chapter is to extrapolate the basic iterative methods described in the previous chapter essentially the SOR method. The idea of extrapolating an iterative procedure to accelerate the convergence was first introduced by L.F. Richardson in 1910. Many researchers have been developing Richardson's idea in various ways, in particular the preconditioning of the coefficient matrix $A$.

We now describe a preconditioning approach (Evans and Missirlis, 1983, also Missirlis, 1985) which will be the base for the construction of extrapolated linear first order iterative methods. The approach essentially needs a non-singular matrix $R$ of order $n$ which is a good approximate of $A$ to transform the original system (5.1.1) into the preconditioned one of the form:

$$R^{-1}Au = R^{-1}b$$ \hspace{1cm} (5.1.2)

or

$$Hu = k$$

where

$$H = R^{-1}A$$ and $$k = R^{-1}b$$.

We refer to $R$ as the preconditioning matrix. It is required to have the following properties:

(a) $R$ is "easily" invertible.

(b) The condition number of the matrix $R^{-1}A$ is smaller than that of $A$, i.e.
For any vectors \( \mathbf{s} \) and \( \mathbf{t} \) it is "convenient" to solve the system \( \mathbf{R}\mathbf{s} = \mathbf{t} \).

The approach consists of the following steps:

(i) Generate the sequence \( \{u^{(k)}\} \), \( k = 0,1,2,\ldots \) defined by the extrapolated method

\[
\begin{align*}
  u^{(k+1)} &= u^{(k)} + \tau(k \mathbf{H} u^{(k)}) & (5.1.3) \\
  \text{or} &
  u^{(k+1)} &= G_{\tau} u^{(k)} + \mathbf{r}_{\tau} & (5.1.4)
\end{align*}
\]

where \( G_{\tau} = I - \tau\mathbf{H} \), \( \mathbf{r}_{\tau} = \tau\mathbf{k} \), and \( u^{(0)} \) is arbitrary vector. Also \( \tau \neq 0 \) is a real extrapolation parameter.

(ii) Determine the optimal value of \( \tau \) which makes the sequence \( \{u^{(k)}\}, \ k = 0,1,2,\ldots \) converge as fast as possible to the solution of the system (5.1.1).

For the impractical case where \( \mathbf{R} = \mathbf{A} \), it follows that:

\[
\begin{align*}
  u^{(k+1)} &= (1-\tau)u^{(k)} + \tau\mathbf{A}^{-1}\mathbf{b} \\
  \text{and} &
  (5.1.6)
\end{align*}
\]

Also, for the choice \( \tau = 1 \), (5.1.6) gives the solution immediately and (5.1.4) reduces to the basic iterative method

\[
\begin{align*}
  u^{(k+1)} &= G u^{(k)} + \mathbf{r} & (5.1.7) \\
\end{align*}
\]

where \( G = I - \mathbf{H} \) and \( \mathbf{r} = \mathbf{k} \). (5.1.8)

For the extrapolated method (5.1.4) to converge to the unique solution of (5.1.1), we must have,
\[ \rho(G_\tau) < 1 \] (5.1.9)

where \( \rho(G_\tau) \) is the spectral radius of \( G_\tau \). The problem now is to choose \( \tau \) (optimal) such that \( \rho(G_\tau) \) is minimum and \( \rho(G_\tau) < \rho(G) \) to produce a higher rate of convergence.

Assume \( \mu_i, \ i = 1,2,\ldots,n \) are the real eigenvalues of \( H \) such that

\[ \mu_i \in [a,b], \ i = 1,2,\ldots,n \] (5.1.10)

where \( a \) and \( b \) are the lower and upper bounds of the eigenvalues of \( H \). It follows from (5.1.5) that the eigenvalues \( \lambda_i, \ i = 1,2,\ldots,n \) of \( G_\tau \) and \( \mu_i \) satisfy the relationship

\[ \lambda_i = 1 - \tau \mu_i, \ i = 1,2,\ldots,n \] (5.1.11)

The spectral radius of \( G_\tau \) is less than unity if and only if \(|\lambda_i| < 1\). It follows now that the necessary and sufficient conditions for convergence are either

\[ a > 0 \text{ and } 0 < \tau < \frac{2}{b} \] (5.1.12a)

or

\[ b < 0 \text{ and } \frac{2}{a} < \tau < 0 \] (5.1.12b)

Equation (5.1.12) indicates that the method (5.1.4) converges if and only if all the eigenvalues \( \mu_i \) of \( H \) are all positive or negative (i.e. \( 0 < a < b \) or \( a < b < 0 \)) and \( \tau \) satisfies (5.1.12a) or (5.1.12b) respectively.

It is clear that the extrapolated method (5.1.4) can converge while the basic method (5.1.7) may diverge, in other words it is not required that \( \rho(G) < 1 \).
Here we present the extrapolated versions of the conventional first order iterative methods. We use the usual splitting for \( A \), i.e.

\[
A = D - L - U
\]  

(5.2.1)

where \( D \) is a diagonal matrix with non-zero elements and \( L \) and \( U \) are the strictly lower and upper triangular matrices respectively.

The system (5.1.1) may be written in component form as:

\[
\sum_{j=1}^{n} a_{i,j} u_j = b_i, \quad i = 1, 2, \ldots, n
\]  

(5.2.2)

where \( A = (a_{i,j}), i,j = 1, 2, \ldots, n \), \( u = (u_i), i = 1, 2, \ldots, n \)

and \( b = (b_i), i = 1, 2, \ldots, n \).

### 1 - Extrapolated Jacobi Method (EJ)

For the choice \( R = D \), equation (5.1.4) yields

\[
u^{(k+1)} = B \cdot u^{(k)} + \tau c
\]  

(5.2.3)

where

\[
B = I - TD^{-1}A \quad \text{and} \quad c = D^{-1}b
\]  

(5.2.4)

Equivalently, equation (5.2.3) can be written in the following form,

\[
u^{(k+1)} = (1-\tau)I + \tau B \cdot u^{(k)} + \tau c
\]  

(5.2.5)

where,

\[
B = L' + U', \quad L' = D^{-1}L \quad \text{and} \quad U' = D^{-1}U
\]  

(5.2.6)

Equation (5.2.3) or equation (5.2.5) is called the Extrapolated Jacobi method or sometimes the Jacobi overrelaxation method, (see (4.3.7b)).

Clearly, for \( \tau = 1 \), the EJ method reduces to the Jacobi
method. In component form equation (5.2.5) will have the form,

\[ u^{(k+1)}_i = (1-\tau)u^{(k)}_i + \tau \left[ b_i - \sum_{j=1, j \neq i}^{n} \frac{a_{i,j}u^{(k)}_j}{a_{i,i}} \right] \]

for \( i = 1, 2, \ldots, n \).

2 - Extrapolated Gauss-Seidel Method (EGS)

By setting \( R = D - L \), which is a better approximation to \( A \) than \( R = D \) as in the EJ method, then equation (5.1.4) yields

\[ u^{(k+1)} = L^{-1}u^{(k)} + \tau (D-L)^{-1}b \]

where \( L^{-1} = I - \tau (D-L)^{-1}A \).

In terms of \( L \) and \( U \), we can derive from (5.2.8)

\[ (D-L)u^{(k+1)} = (D-L)u^{(k)} - \tau A u^{(k)} + \tau b \]

\[ = (D-L)u^{(k)} - \tau (D-L)u^{(k)} + \tau U u^{(k)} + \tau b \]

\[ u^{(k+1)} = (1-\tau)u^{(k)} + \tau (D-L)^{-1}U u^{(k)} + \tau (D-L)^{-1}b \]

here, \( L^{-1} = (1-\tau)I + \tau (D-L)^{-1}U \).

Equations (5.2.8) or (5.2.10) is called the extrapolated Gauss-Seidel method with iteration matrix \( L^{-1} \). When \( \tau = 1 \) the method degenerates into the GS method. Equivalently, we can write (5.2.10) as,

\[ u^{(k+1)} = (1-\tau)u^{(k)} + \tau \left[ L u^{(k)} + r_{GS} \right] \]

where \( L \) is the GS iteration matrix and \( r_{GS} \) is its constant vector.
(see (4.3.11)). Equation (5.2.11) can be written in component form (using (4.3.8)) as,
\[
\begin{align*}
u^{(k+1)}_i &= (1-\tau)\nu^{(k)}_i + \tau \left[ b_i - \sum_{j=1}^{i-1} a_{ij} \nu^{(k+1)}_j - \sum_{j=i+1}^{n} a_{ij} \nu^{(k)}_j \right]/a_{ii} \\
&= \frac{1}{a_{ii}} \left[ (1-\tau)\nu^{(k)}_i + \tau \left( b_i - \sum_{j=1}^{i-1} a_{ij} \nu^{(k+1)}_j - \sum_{j=i+1}^{n} a_{ij} \nu^{(k)}_j \right) \right],
\end{align*}
\]  
for i = 1, 2, \ldots, n.  \quad (5.2.12)

3 - Extrapolated Successive Overrelaxation (ESOR) Method

This method can be obtained by letting \( R = D - \omega L \), where \( \omega \) is a real parameter which plays the role of the overrelaxation parameter in the SOR method. From (5.1.4) we have
\[
\begin{align*}
u^{(k+1)} &= L_{\tau,\omega} \nu^{(k)} + \tau (D - \omega L)^{-1} b \\
\text{where} \quad L_{\tau,\omega} &= I - \tau (D - \omega L)^{-1} A.
\end{align*}
\]  \quad (5.2.13)

Equation (5.2.13) can now be expanded in terms of L and U in the following steps
\[
\begin{align*} 
(D - \omega L)u^{(k+1)} &= \left[ (1-\tau)D + (1-\omega)L + \tau U \right] u^{(k)} + \tau b \\
or 
\begin{align*}
u^{(k+1)} &= L_{\tau,\omega} \nu^{(k)} + \tau (D - \omega L)^{-1} b \\
\text{where} \quad L_{\tau,\omega} &= (D - \omega L)^{-1} \left[ (1-\tau)D + (1-\omega)L + \tau U \right].
\end{align*}
\]  \quad (5.2.15)

Equations (5.2.13) or (5.2.16) represent the ESOR method. It can be written in the following component form
\[
\begin{align*}
u^{(k+1)}_i &= (1-\tau)\nu^{(k)}_i - \frac{\omega}{a_{ii}} \sum_{j=1}^{i-1} a_{ij} \nu^{(k+1)}_j - \frac{(1-\omega)}{a_{ii}} \sum_{j=i+1}^{n} a_{ij} \nu^{(k)}_j - \\
&\quad - \frac{\tau}{a_{ii}} \sum_{j=i+1}^{n} a_{ij} \nu^{(k)}_j + \frac{b_i}{a_{ii}}, \quad \text{for } i = 1, 2, \ldots, n.
\end{align*}
\]  \quad (5.2.18)
It is notable that for $\tau = \omega$, ESOR reduces to the SOR method and $L_{\tau, \omega}$ reduces to $L_{\omega}$, the SOR iteration matrix. Furthermore for $\omega = 1$, ESOR method reduces to the EGS method. More details about ESOR will be given in Section 5.4.1.

5.3 CONVERGENCE ANALYSIS AND THEOREM

In this section we investigate the convergence of the extrapolated iterative methods derived in the previous section, assuming that $A$ is a consistently ordered matrix with non-zero diagonal elements, and the Jacobi iteration matrix $B = I - D^{-1}A$ possesses only real eigenvalues $\lambda_i$, $i = 1, 2, \ldots, n$ such that $\mu_i \in (\underline{\mu}, \overline{\mu})$ where $\mu$ and $\overline{\mu}$ are the lower and upper bounds of $\lambda_i$.

From (5.2.5), we have the iteration matrix of the EJ method as,

$$B_{\tau} = (1-\tau)I + \tau B$$

(5.3.1)

Now, we can state that if the Jacobi method is convergent, i.e. $\overline{\mu} = \rho(B) < 1$, then the EJ method converges for $0 < \tau < 1$. To prove this we assume that $\lambda_{i1}$ is the eigenvalue of $B_{\tau}$, then from (5.3.1) we have,

$$\lambda_{i1} = (1-\tau) + \tau \mu_{i1}$$

(5.3.2)

or

$$\lambda_{i1}^2 = \left[ (1-\tau) + \tau \mu_{i1} \right]^2 < 1$$

for $\tau < 1$

and consequently, as $\mu_{i1} < 1$,

$$|\lambda_{i1}| < 1$$

(5.3.3)

i.e. the EJ converges for $0 < \tau < 1$ provided $\rho(B) < 1$.

We now state some theorems governing the convergence of the EGS method.
Theorem 5.3.1

Consider the coefficient matrix $A$ and the Jacobi iteration matrix $B$. Then the EGS method converges if and only if

$$\overline{\mu} < 1 \quad \text{and} \quad 0 < \tau < 2/(1-\overline{\mu}^2)$$  \hspace{1cm} (5.3.4)

or

$$\mu > 1 \quad \text{and} \quad 2/(1-\mu^2) < \tau < 0.$$  \hspace{1cm} (5.3.5)


Corollary 5.3.1

Under the assumption of theorem 5.3.1 and if $\mu = 0$, then the EGS method converges if and only if

$$\overline{\mu} < 1 \quad \text{and} \quad 0 < \tau < 2.$$  \hspace{1cm} (5.3.6)

Proof:

When $\mu = 0$, (5.3.4) reduces to (5.3.6) and (5.3.5) is no longer valid.

Theorem 5.3.2

For the EGS method, the optimal value of $\tau$ is given by

$$\tau_{\text{opt.}} = \frac{2}{2 - \mu^2 - \overline{\mu}^2}$$  \hspace{1cm} (5.3.7)

and the corresponding value of $\rho(L_{\tau,1})$ is given by

$$\rho(L_{\tau,1}) = \begin{cases} \frac{(\mu^2-\mu^2)/(2-\mu^2-\overline{\mu}^2)}{2 - \mu^2 - \overline{\mu}^2}, & \text{if } \overline{\mu} < 1 \\ \frac{(\mu^2-\mu^2)/(\mu^2+\mu^2-2)}{2 - \mu^2 - \overline{\mu}^2}, & \text{if } \mu > 1 \end{cases}$$  \hspace{1cm} (5.3.8)

Corollary 5.3.2

Under the assumption of theorem 5.3.2 and if $\mu = 0$, then the optimal value of $\tau$ is given by

$$\tau_{\text{opt.}} = \frac{2}{2 - \mu^2}$$  \hspace{1cm} (5.3.9)

and the corresponding minimized $\rho(L_{\tau,1})$ is

$$\rho(L_{\tau,1}) = \frac{\mu^2}{2 - \mu^2}.$$  \hspace{1cm} (5.3.10)

Proof:

It follows immediately by letting $\mu = 0$ in equation (5.3.7) and (5.3.8).

5.4 ACCELERATING THE SOR METHOD

Although the SOR method is proved to be a very efficient method, successful attempts to accelerate its convergence have been made, in particular, by Evans and Missirlis, Hadjidimos and Neithammer who discovered and developed the following methods: the Extrapolated SOR, the Accelerated Overrelaxation and the Two Parametric methods, respectively.

In this section, we derive each of the above mentioned methods together with the analysis of convergence and the optimal estimates of the acceleration parameters. We consider the linear system

$$Au = b$$  \hspace{1cm} (5.4.1)

with the same splitting in (5.2.1) and

$$A = [a_{ij}], \hspace{0.5cm} i = 1, 2, \ldots, n.$$
5.4.1 Extrapolated SOR (ESOR) Method

The ESOR method (derived in section 5.2) is given by

\[ u^{(k+1)} = L_{\tau,\omega} u^{(k)} + \tau (D-\omega L)^{-1} b, \]  

(5.4.2)

where

\[ L_{\tau,\omega} = I - \tau (D-\omega L)^{-1} A, \]  

(5.4.3)

with \( \omega, \tau(\neq 0) \) are the real overrelaxation and extrapolation parameters respectively. The necessary and sufficient condition for ESOR to converge is

\[ \rho(L_{\tau,\omega}) < 1. \]  

(5.4.4)

Analysis of Convergence

Here, we discuss the convergence of the ESOR method according to the form of the coefficient matrix \( A \). We consider three cases: the irreducible matrix, L-matrix and the consistently ordered matrix. We are more concerned with the later case.

(i) The irreducible matrix

If \( A \) is an irreducible matrix with weak diagonal dominance, then we can prove the following theorem:

Theorem 5.4.1

Let \( A \) be an irreducible matrix with weak diagonal dominance. Then the ESOR method converges for \( 0 < \tau \leq 1 \) and \( 0 \leq \omega \leq 1 \).

Proof:

The proof can be achieved by contradiction where we assume that \( \rho(L_{\tau,\omega}) \geq 1 \), and for some eigenvalues \( \lambda \) of \( L_{\tau,\omega} \) we have
\[ |\lambda| \geq 1. \text{ We have the following characteristic equation} \]
\[ \det(L_{\tau,\omega} - \lambda I) = 0 \quad (5.4.5) \]

or after a series of transformations

\[ \det(Q) = 0 \]

where

\[ Q = I - \left( \frac{\tau - \omega + \omega \lambda}{\lambda + \tau - 1} \right) D^{-1} L - \left( \frac{\tau}{\lambda + \tau - 1} \right) D^{-1} U \quad (5.4.6) \]

The coefficients of \( D^{-1} L \) and \( D^{-1} U \) are less than one in modulus.

To prove this, we let \( \lambda^{-1} = q e^{i\theta} \) where \( q \) and \( \theta \) are real and \( 0 < q = 1 \), we have

\[ \left| \frac{\tau - \omega + \omega \lambda}{\lambda + \tau - 1} \right| = \left| \frac{(\tau - \omega)^2 + 2\omega q(\tau - \omega) \cos \theta + q^2}{2(1 - 2q(1 - \tau) \cos \theta + q^2(1 - \tau)^2} \right| \leq \frac{\omega + q(\tau - \omega)}{1 - q(1 - \tau)} , \]

since \( q \leq 1, 0 < \tau \leq 1 \) and \( 0 \leq \omega \leq 1 \). Moreover we show

\[ 1 - \frac{\omega + q(\tau - \omega)}{1 - q(1 - \tau)} = \frac{(1 - q)(1 - \omega)}{1 - q(1 - \tau)} \geq 0 , \]

and hence

\[ \left| \frac{\tau}{\lambda + \tau - 1} \right| \leq \left| \frac{\tau + \omega(\lambda - 1)}{\lambda + \tau - 1} \right| \leq 1 . \quad (5.4.7) \]

Now, \( D^{-1} A = I - D^{-1} L - D^{-1} U \) is also irreducible and has weak diagonal dominance, so the matrix \( Q \) in (5.4.6) with the use of (5.4.7) is also irreducible with weak diagonal dominance and whence it is non-singular, i.e. \( \det(Q) \neq 0 \) which contradicts (5.4.5) and therefore \( \rho(L_{\tau,\omega}) < 1 \), which is the sufficient condition for convergence.
(ii) **L-Matrix**

We give the following theorem:

**Theorem 5.4.2**

If $A$ is an L-matrix of order $n$ and $0 \leq \omega \leq \tau \leq 1$ ($\tau \neq 0$).

Then

(a) $\rho(L_{\tau,\omega}) < 1$ if and only if $\rho(B) < 1$ \hspace{1cm} (5.4.8)

(b) $\rho(L_{\tau,\omega}) < 1$ and $\rho(B) < 1$ if and only if $A$ is an M-matrix;
if $\rho(B) < 1$ then $\rho(L_{\tau,\omega}) \leq 1 - \tau + \tau \rho(B)$ . \hspace{1cm} (5.4.9)

(c) If $\rho(B) \geq 1$ and $\rho(L_{\tau,\omega}) \geq 1$, then $\rho(L_{\tau,\omega}) \geq 1 - \tau + \tau \rho(B) \geq 1$.

**Proof:** See Missirlis, 1985.

**Corollary 5.4.1**

If $A$ is an L-matrix of order $n$ and if $0 \leq \omega \leq 1$.

Then

(a) $\rho(L_{1,\omega}) < 1$ if and only if $\rho(B) < 1$

(b) $\rho(L_{1,\omega}) < 1$ and $\rho(B) < 1$ if and only if $A$ is an M-matrix,
if $\rho(B) < 1$ then $\rho(L_{1,\omega}) \leq \rho(B)$ .

(c) If $\rho(B) \geq 1$ and $\rho(L_{1,\omega}) \geq 1$ then $\rho(L_{1,\omega}) \geq \rho(B) \geq 1$. 
(iii) **Consistently Ordered Matrix**

Here we discuss the convergence of the ESOR method when \( A \) is consistently ordered (2-cyclic) and the Jacobi iteration matrix \( B \) possesses only real eigenvalues.

**Theorem 5.4.3**

Let \( A \) be a consistently ordered matrix with non-vanishing diagonal elements and let \( \mu \) be an eigenvalue of \( B = I - D^{-1} A \) satisfying the relationship

\[
(1 - \lambda^2) = \mu^2 (1 - \lambda \omega),
\]

then \( \lambda \) is an eigenvalue of the matrix

\[
\Lambda_\omega = (D - \omega L)^{-1} A,
\]

and vice versa.

**Proof:**

The proof is analogous to theorem 5.2.2 in Young, 1971.

A necessary and sufficient condition for the ESOR method to converge is that \( \rho(L_{\tau,\omega}) < 1 \). Thus, if we let \( \lambda = a + ib \), where \( a, b \) are real numbers and \( i = \sqrt{-1} \) be a complex eigenvalue of \( \Lambda_\omega \), then the ESOR method converges if and only if

\[
|1 - \tau(a + ib)| < 1,
\]

where \( 1 - \tau(a + ib) \) is an eigenvalue of \( L_{\tau,\omega} \) with modulus

\[
\left[ (1 - \tau a)^2 + \tau^2 b^2 \right]^{1/2},
\]

so from (5.4.12) we have

\[
\tau^2(a^2 + b^2) < 2\tau a,
\]
which implies that $\tau a > 0$. From (5.4.13) we equivalently have
the following

$$a > 0, \ 0 < \tau < \frac{2a}{a^2 + b^2} \ \text{or} \ a < 0 \ \text{and} \ \frac{2a}{a^2 + b^2} < \tau < 0. \quad (5.4.14)$$

**Theorem 5.4.4**

Let $A$ be a consistently ordered matrix with non-vanishing
diagonal elements such that the matrix $B = I - D^{-1}A$ has real
eigenvalues $\mu_i$, $i = 1, 2, \ldots, n$ with $\nu = \min \left| \mu_i \right| \neq 0$ and
$\bar{\mu} = \max \left| \mu_i \right| = \rho(B)$. Then the ESOR method converges if and only
if $\rho(B) < 1$ and the parameters $\omega, \tau$ lie in any of the following
domains given in table 5.4.1.

<table>
<thead>
<tr>
<th>$\omega$-Domain</th>
<th>$\tau$-Domain</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$-\infty &lt; \omega \leq \omega'(\mu)$</td>
<td>$0 &lt; \tau &lt; \frac{2}{\lambda_+ (\mu)}$</td>
<td>$-$</td>
</tr>
<tr>
<td>$\omega'(\mu) \leq \omega \leq 2$</td>
<td>$0 &lt; \tau &lt; h(\omega, \mu)$</td>
<td>$-$</td>
</tr>
<tr>
<td>$2 \leq \omega &lt; \frac{2}{\mu^2}$</td>
<td>$0 &lt; \tau &lt; h(\omega, \mu)$</td>
<td>$-$</td>
</tr>
<tr>
<td>$\frac{2}{\mu^2} \leq \omega &lt; \omega''(\mu)$</td>
<td>$h(\omega, \mu) &lt; \tau &lt; 0$</td>
<td>$1 - \frac{\mu^2}{(1-\mu^2)^4}$</td>
</tr>
<tr>
<td>$\omega''(\mu) \leq \omega &lt; +\infty$</td>
<td>$\frac{2}{\lambda_- (\mu)} &lt; \tau &lt; 0$</td>
<td>$-$</td>
</tr>
</tbody>
</table>

*Table (5.4.1)*

where $\lambda_+ (\mu)$, $\omega'(\mu)$, $\omega''(\mu)$ and $h(\omega, \mu)$ will be defined later in
the proof.

**Proof:**

If $\mu$ and $\lambda$ are some eigenvalues of $B$ and $\Lambda_\omega$ respectively,
then from (5.4.10) we have
\[ \lambda^2 - (2-\omega \mu^2)\lambda + 1 - \mu^2 = 0 \] (5.4.15)
whose roots are given by
\[ \lambda_{\pm}(\mu) = (2-\omega \mu^2 \pm \Delta^{\frac{1}{2}})/2 \] , (5.4.16)
where
\[ \Delta = \Delta(\mu) = \mu^2 \left[ \omega^2 \mu^2 - 4(\omega-1) \right] . \] (5.4.17)
The value of \( \lambda_{\pm}(\mu) \) (real or complex) depends on the sign of \( \Delta \)
which is dependent on the value of \( \omega \). \( \omega \) is bounded by the two
quantities
\[ \omega'(\mu) = 2/\left[ 1 + (1-\mu^2)^{\frac{1}{2}} \right] \quad \text{and} \quad \omega''(\mu) = 2/\left[ 1 - (1-\mu^2)^{\frac{1}{2}} \right] \] (5.4.18)
which apparently are the roots of \( \Delta \). Examining the sign of \( \Delta \)
involve the following three basic cases:

Case I : All the \( \lambda_{\pm}(\mu) \) are complex.

Case II : All the \( \lambda_{\pm}(\mu) \) are real.

Case III : Some \( \lambda_{\pm}(\mu) \) are complex and others are real.

Case I : All \( \lambda_{\pm}(\mu) \) are complex and for all \( \mu^2 \) such that
\[ 0 < \mu^2 < \mu^2 \leq \mu^2 \] when \( \Delta(\mu) < 0 \), hence \( \omega'(\mu) \leq \omega \leq \omega''(\mu) \).
Evidently, \[ |\lambda_{\pm}(\mu)|^2 = 1-\mu^2 \] and \( \text{Re} \lambda_{\pm}(\mu) = (2-\omega \mu^2)/2 > 0 \) which
implies \( \omega < 2/\mu^2 \), whereas if \( \text{Re} \lambda_{\pm}(\mu) < 0 \) then we must have
\[ 2/\mu^2 < \omega''(\mu) \text{ or} \]
\[ 1 - \mu^2 < (1-\mu^2)^{\frac{1}{2}} \] (5.4.19)
implying that \( \omega > 2/\mu^2 \). We therefore split the above interval
of \( \omega \) into two cases:

(i) \( \omega'(\mu) \leq \omega < 2/\mu^2 \) and

(ii) \( 2/\mu^2 < \omega \leq \omega''(\mu) \)

and examine each case separately. In case (i), \( \text{Re} \lambda_\pm(\mu) > 0 \), and from (5.4.14), \( \tau \) must lie in the range

\[
0 < \tau < \min_{\mu^2 \in \mu^2} h(\omega, \mu)
\]

where

\[
h(\omega, \mu) = (2-\omega \mu^2)/(1-\mu^2)
\]

(5.4.20)

(5.4.21)

However, sign \([ \partial h(\omega, \mu)/\partial (\mu^2) ] = \text{sign} (2-\omega) \), hence for this subcase the ranges for \( \omega \) and \( \tau \) such that the ESOR method converges are

\[
\omega'(\mu) \leq \omega < \omega''(\mu) \quad \text{and} \quad 0 < \tau < h(\omega, \mu)
\]

or

\[
2 \leq \omega < 2/\mu^2 \quad \text{and} \quad 0 < \tau < h(\omega, \mu).
\]

Alternatively, if (ii) is valid, then (5.4.19) must hold and the range of \( \tau \) is (see (5.4.14))

\[
h(\omega, \mu) < \tau < 0.
\]

Case II : All \( \lambda_\pm(\mu) \) are real, if they are all positive, then

\(-\infty < \omega \leq \omega'(\mu) \) and (5.4.14) yields \( 0 < \tau < 2/\lambda_+(\mu) \) (since \( b = 0 \)).

Since \( \lambda_+(\mu) \) is an increasing function of \( \mu^2 \), then \( 0 < \tau < 2/\lambda_+(\mu) \). Similarly if all \( \lambda_+(\mu) \) are negative, then \( \omega''(\mu) \leq \omega < +\infty \) and

\[
2/\lambda_-(\mu) < \tau < 0.
\]
Case III: Some of $\lambda_{\pm}(\mu)$ are complex and others are real, $\omega$ lies in either of the following ranges:

(i) $\omega'(\mu) \leq \omega \leq \omega'(\overline{\mu})$

or

(ii) $\omega''(\mu) \leq \omega \leq \omega''(\overline{\mu})$

We first suppose that $\omega$ lies in (i), then the real $\lambda_{\pm}(\mu)$ are positive and $0 < \tau < 2/\lambda_{+}(\mu)$ (Case II), whereas for the complex ones, their real parts are positive and $0 < \tau < h(\omega, \mu)$ (Case I).

However, it is verified that $0 < 2/\lambda_{+}(\mu) < h(\omega, \mu)$ implying that when $\omega$ lies in (i) then the interval for $\tau$ is $0 < \tau < 2/\lambda_{+}(\overline{\mu})$.

Secondly, we assume that $\omega$ lies in (ii), the real $\lambda_{\pm}(\mu)$ are negative if and only if $2/\mu^2 < \omega$, hence the range becomes

$$\max\left\{2/\mu^2, \omega''(\mu)\right\} \leq \omega \leq \omega''(\overline{\mu}).$$

For this range we have $2/\lambda_{-}(\mu) < \tau < 0$ (Case II). For complex $\lambda_{\pm}(\mu)$, the range of $\tau$ is $h(\omega, \mu) < \tau < 0$ (Case I). But $h(\omega, \mu) < 2/\lambda_{-}(\mu) < 0$, therefore the necessary and sufficient conditions in this case are

$$\max\left\{2/\mu^2, \omega''(\mu)\right\} \leq \omega \leq \omega''(\overline{\mu}) \quad \text{and} \quad 2/\lambda_{-}(\overline{\mu}) < \tau < 0.$$


Corollary 5.4.2

Under the hypothesis of theorem 5.4.4 and if $\mu = 0$, then the ESOR method converges if and only if $\overline{\mu} < 1$ and either

$$-\infty < \omega < 1 \quad \text{and} \quad 0 < \tau < 2/\lambda_{+}(\overline{\mu}), \quad (5.4.22)$$

or

$$1 \leq \omega \leq 2 \quad \text{and} \quad 0 < \tau < 2, \quad (5.4.23)$$

or

$$2 \leq \omega < 2/\mu^2 \quad \text{and} \quad 0 < \tau < h(\omega, \overline{\mu}), \quad (5.4.24)$$
Corollary 5.4.3

Under the hypothesis of theorem 5.4.4 and if \(0 < \mu = \bar{\mu} = \mu < 1\), the ESOR method converges if and only if \(\mu < 1\) and the parameters \(\omega, \tau\) lie in any of the corresponding domains given in table 5.4.2.

<table>
<thead>
<tr>
<th>(\omega)-Domain</th>
<th>(\tau)-Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>(- \infty &lt; \omega \leq \omega'(\mu))</td>
<td>(0 &lt; \tau &lt; 2/\lambda_+(\mu))</td>
</tr>
<tr>
<td>(\omega'(\mu) &lt; \omega &lt; 2/\mu^2)</td>
<td>(0 &lt; \tau &lt; h(\omega, \mu))</td>
</tr>
<tr>
<td>(2/\mu^2 &lt; \omega \leq \omega''(\mu))</td>
<td>(h(\omega, \mu) &lt; \tau &lt; 0)</td>
</tr>
<tr>
<td>(\omega''(\mu) &lt; \omega &lt; +\infty)</td>
<td>(2/\lambda_-(\mu) &lt; \tau &lt; 0)</td>
</tr>
</tbody>
</table>

Table (5.4.2)

The proofs of the preceding two corollaries can be obtained by following a similar approach to the proof of theorem 5.4.4.

Optimal Parameters

We need to determine the optimal values of \(\omega\) and \(\tau\) which minimize \(\rho(L_{\tau, \omega})\). The following theorem (without proof) with two corollaries deal with this problem for the case of consistently ordered matrices.

Theorem 5.4.5

If \(A\) is a consistently ordered matrix with non-vanishing diagonal elements such that \(B\) has real eigenvalues \(\mu_i\), \(i = 1, 2, \ldots, n\)
with \( \mu = \min_{i} |\mu_i| \neq 0 \) and \( \bar{\mu} = \max_{i} |\mu_i| < 1 \). Also if
\[ 1 - \mu^2 < (1 - \mu^2)^{\frac{1}{2}} \]
then the optimal values of \( \omega \) and \( \tau \) are
\[
\begin{align*}
\omega_{\text{opt.}} &= \omega_b = \omega'(\mu) = 2 \left/ \left[ 1 + (1 - \mu^2)^{\frac{1}{2}} \right] \right., \\
\tau_{\text{opt.}} &= \tau_b = \left( 2 - \omega_b\mu^2 \right) / \left[ 2(1 - \mu^2) \right],
\end{align*}
\]
and the corresponding minimum \( \rho(L_{\tau, \omega}) \) is given by
\[
\rho(L_{\tau_b, \omega_b}) = \left[ \mu \left( \frac{\mu^2 - \mu^2}{2} \right) \right] / \left[ \left( 1 - \mu^2 \right)^{\frac{1}{2}} (1 + (1 - \mu^2)^{\frac{1}{2}}) \right]. \tag{5.4.26}
\]

**Corollary 5.4.4**

Under the hypothesis of theorem 5.4.5 and if \( \mu = 0 \), then
\( \rho(L_{\tau, \omega}) \) is minimized at \( \tau_b = \omega_b = \omega'(\mu) \) and its corresponding value is
\[
\rho(L_{\omega_b, \omega_b}) = \omega_b - 1 = \left[ 1 - (1 - \mu^2)^{\frac{1}{2}} \right] / \left[ 1 + (1 - \mu^2)^{\frac{1}{2}} \right]. \tag{5.4.27}
\]

**Corollary 5.4.5**

Under the assumptions of theorem 5.4.5 and if \( 0 < \mu = \bar{\mu} = \mu < 1 \) then for either
\[
\omega_b = \omega'(\mu) \text{ and } \tau_b = \frac{1}{\sqrt{1 - \mu^2}} \quad \text{or} \quad \omega_b = \omega''(\mu) \text{ and } \tau_b = \frac{-1}{1 - \mu^2}
\]
we have
\[
\rho(L_{\tau_b, \omega_b}) = 0. \tag{5.4.29}
\]
5.4.2 Accelerated Overrelaxation (AOR) Method

This method was introduced by A. Hadjidimos, 1978, it requires two real parameters; the overrelaxation parameter $\omega$ and another acceleration parameter $r$. To derive the method he proposed a general linear stationary iterative scheme whose matrix coefficients are linear functions of the components of $A$. The scheme has the form

$$(a_1D+a_2L)u^{(k+1)} = (a_3D+a_4L+a_5U)u^{(k)} + a_6b, \quad k = 0,1,2,...$$

(5.4.30)

where $a_i$, $i = 1,2,...,6$ are constants ($a_i \neq 0$) and $u^{(0)}$ is an arbitrary initial approximation to the solution $u$ of the system (5.4.1). We divide through (5.4.30) by $a_1$ to obtain

$$(D+a'_2L)u^{(k+1)} = (a'_3D + a'_4L + a'_5U)u^{(k)} + a'_6b$$

(5.4.31)

where $a'_i = \frac{a_i}{a_1}$, $i = 2,3,...,6$. The procedure (5.4.31) is consistent with the system (5.4.1) if

$$(1-a'_3)D + (a'_2-a'_4)L - a'_5U = a'_6 A, \quad a'_6 \neq 0$$

which implies the determination of $a'_i$'s in terms of $a'_6$ as follows

$$1 - a'_3 = a'_6, \quad a'_2 - a'_4 = -a'_6 \quad \text{and} \quad a'_5 = -a'_6.$$

We consider the following two-parameter solution

$$a'_2 = -r, \quad a'_3 = 1-\omega, \quad a'_4 = \omega-r, \quad a'_5 = \omega \quad \text{and} \quad a'_6 = \omega,$$

where $r, \omega (\neq 0)$ are some two real parameters. So, equation (5.4.31) can be written as

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

(5.4.32)
or
\[ u^{(k+1)} = L_{r,\omega} u^{(k)} + \omega(D-rL)^{-1} b, \quad (5.4.33) \]

where
\[ L_{r,\omega} = (D-rL)^{-1} \left[ (1-\omega)D + (\omega-r)L + \omega U \right] \]
is the iteration matrix of the AOR method (5.4.33). Specific values of the parameters \( r \) and \( \omega \) will reduce the AOR method (5.4.33) to some well-known iterative methods namely:

- \( r = 0, \omega = 1 \) will give the Jacobi method
- \( r = 1, \omega = 1 \) will give the Gauss-Seidel method
- \( r = 0, \omega \) will give the Jacobi overrelaxation (JOR) method
- \( r = \omega \) will give the SOR method.

Another approach to derive the method is to use the splitting \( A = Q - C \) (see (4.3.1)) such that
\[ Q = \frac{1}{\omega} (D-rL), \quad C = \frac{1}{\omega} \left[ (1-\omega)D + (\omega-r)L + \omega U \right]. \quad (5.4.34) \]
By substituting (5.4.34) in the general iterative method
\[ Q u^{(k+1)} = C u^{(k)} + b \] (see (4.3.2)), we obtain the AOR method (5.4.33).

Analysis of Convergence

We study the convergence of the AOR according to the following three types of the coefficients matrix \( A \).

(i) The Irreducible matrix

If \( A \) is irreducible matrix with weak diagonal dominance, then it is non-singular with non-vanishing diagonal elements (2-cyclic). We have the following theorem:
Theorem 5.4.6

If A is an irreducible matrix with weak diagonal dominance, then the AOR method converges for all $0 \leq r \leq 1$ and $0 < \omega \leq 1$.

Proof:

This is similar to the proof of theorem 5.4.1.

(ii) The L-Matrix

If the coefficients matrix A is an L-matrix, then we have the theorem:

Theorem 5.4.7

If A is an L-matrix and $0 \leq r \leq \omega \leq 1$ ($\omega \neq 0$), then the AOR method converges if and only if $\rho(B) < 1$.


(iii) Consistently Ordered Matrix

For a consistently ordered matrix A with non-vanishing diagonal elements, if $\mu$ is an eigenvalue of $B = D^{-1}(L+U)$ and $\lambda$ satisfies

$$ (\lambda-1+\omega)^2 = \omega \mu^2 \left[ r(\lambda-1) + \omega \right] $$

(5.4.35)

then $\lambda$ is an eigenvalue of $L_{r,\omega}$ and vice versa.

Theorem 5.4.8

If A is a consistently ordered matrix with non-vanishing diagonal elements, and if $\mu_i$, $i = 1, 2, \ldots, n$ are the eigenvalues
of $B$ such that $\mu = \min_i |\mu_i|$ and $\overline{\mu} = \max_i |\mu_i|$, then the AOR method converges (i.e. $\rho(L_{r,\omega}) < 1$) if and only if the Jacobi method converges (i.e. $\rho(B) < 1$) and the parameters of $\omega$ and $r$ take the values from the intervals $I_\omega$ and $I_r$, respectively defined as follows:

for $\mu \neq 0$:

$I_\omega = (-2/\sqrt{1-\mu^2}, 0)$ and $I_r = (\beta(\mu^2), \alpha(\mu^2))$

or $I_\omega = (0,2]$ and $I_r = (\alpha(\mu^2), \beta(\mu^2))$

or $I_\omega = [2,2/\sqrt{1-\mu^2}]$ and $I_r = (\alpha(\mu^2), \beta(\mu^2))$

for $\mu = 0$: $I_\omega = (0,2)$ and $I_r = (\alpha(\mu^2), \beta(\mu^2))$

where

$$\alpha(z) = \frac{1}{wz} \left( \frac{w^2}{2} - \frac{w^2}{2} + 2w^2 \right)$$

and

$$\beta(z) = \frac{1}{2} (\omega z - \omega + 2).$$

If $0 < \mu < \mu = \overline{\mu} < 1$, then for $(\omega, r) = \left( \frac{-1}{\sqrt{1-\mu^2}} , \frac{2(1+\sqrt{1-\mu^2})}{\mu^2} \right)$

or

$$\left( \frac{1}{\sqrt{1-\mu^2}} , \frac{2}{1+\sqrt{1-\mu^2}} \right), \quad \rho(L_{r,\omega}) = 0.$$


5.4.3 Two Parametric Method

Given a linear system

$$Au = b \quad (5.4.36)$$
Then by splitting \( A \) as \( A = Q - C \) (see (4.3.1)) we can obtain the basic iterative matrix as

\[
Q u^{(k+1)} = C u^{(k)} + b .
\] (5.4.37)

Sisler in (1973-75) introduced and studied the two-Parametric method for the case of cyclic matrices. W. Neithammer (1979) proved that the two-parametric method is closely related to the SOR method and derived some results for the case when \( A \) is noncyclic.

Now the SOR method can be obtained by letting

\[
Q = \left( \frac{1}{\omega} \right) D - L \quad \text{and} \quad C = \left( \frac{1}{\omega} - 1 \right) D + U , \quad \omega \neq 0 .
\] (5.4.38)

So, \( L_\omega = (D-\omega L)^{-1} \left[ (1-\omega)D + \omega U \right] \),

where \( L_\omega \) is the SOR iteration matrix.

Another iterative method can be obtained by letting

\[
Q = D - \beta L , \quad C = (1 - \beta) L + U
\]

which will have the iteration matrix \( L_\beta \),

\[ L_\beta = (D - \beta L)^{-1} \left[ (1 - \beta) L + U \right] . \]

Now, when the two methods are combined, we have similar splitting where

\[
Q = \frac{1}{\psi} D - \beta L \quad \text{and} \quad C = \left( \frac{1}{\psi} - 1 \right) D + (1 - \beta) L + U
\]

and the resulting iterative method has the following iteration matrix

\[
V(\psi, \beta) = (D - \psi \beta L)^{-1} \left[ (1 - \psi) D + (1 - \beta) \psi L + \psi U \right] .
\] (5.4.39)

Actually, \( V(\psi, \beta) \) is the iteration matrix of the two-parametric method, i.e.
Now we have the following lemma which relates the SOR method to the two-parametric method:

**Lemma 5.4.1**

If $L_\omega$ is the SOR iteration matrix and $V(\psi,\beta)$ is the two-parametric iteration matrix, then the following holds:

$$V(\psi,\beta) = \eta L_\omega + (1-\eta) \mathbf{D} = z(\omega,\eta) \quad (5.4.41)$$

where $\psi = \omega \eta$ and $\beta = \frac{1}{\eta} \quad (\eta, \beta \neq 0)$.

**Proof:**

By substituting the parameters $\psi$ and $\beta$ (in terms of $\omega$ and $\eta$) into (5.4.39), we get

$$V(\psi,\beta) = (D-\omega L)^{-1} \left[ (1-\omega \eta) D + \omega \eta (1 - \frac{1}{\eta}) L + \omega \eta U \right]$$

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

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

$$= \eta L_\omega + (1-\eta) \mathbf{D} = z(\omega,\eta) \quad .$$

Now, by using the transformation (5.4.41) we can write the two-parametric method as

$$\mathbf{u}^{(k+1)} = (D-\omega L)^{-1} \left[ (1-\omega \eta) D + \omega (\eta-1) L + \omega \eta U \right] \mathbf{u}^{(k)} + (D-\omega L)^{-1} \omega \eta \mathbf{b} \quad (5.4.42)$$

where $\omega$ is the relaxation parameter and $\eta$ is the acceleration parameter.
It can easily be seen that for \( n = 1 \), the two-parametric method reduces to the SOR method, i.e. \( L_\omega = z(\omega, 1) \).

If \( \lambda(\omega) \) and \( \xi(\omega, \eta) \) are eigenvalues of \( L_\omega \) and \( z(\omega, \eta) \) respectively, then from (5.4.41), we have the relationship

\[
\xi(\omega, \eta) = \eta \lambda(\omega) + (1 - \eta)
\]

(5.4.43)

If \( k(c, d) \) is the circle with centre \( c \) through the point \( d \), \( \overline{k}(c, d) \) is the interior of \( k(c, d) \), then it follows from (5.4.43) that,

\[
|\xi(\omega, \eta)| < 1 \quad \text{if and only if} \quad |\lambda(\omega) - (1 - \frac{1}{\eta})| < \frac{1}{\eta} \quad \text{for} \quad \eta > 0.
\]

Also, \( |\xi(\omega, \eta)| < 1 \) if \( \lambda(\omega) \in \overline{k}(1 - \frac{1}{\eta}, 1) \), so \( z(\omega, \eta) \) is convergent if and only if the eigenvalues of \( L_\omega \) are in the open disk \( \overline{k}_\eta = \overline{k}(1 - \frac{1}{\eta}, 1) \) with centre \((1 - \frac{1}{\eta})\) and radius \( \frac{1}{\eta} \).

**Regions of Convergence**

(i) **Non-cyclic Case**

In this case, a region \( R \) in the \((\omega, \eta)\)-plane has to be determined such that for \((\omega, \eta) \in R \), the iteration matrix \( z(\omega, \eta) \) is convergent (i.e. \( \rho(\omega(\omega, \eta)) < 1 \)). The following lemma describes the region of convergence for the two-parametric method.

**Lemma 5.4.2**

If \( A \) is symmetric positive definite with eigenvalues \( \lambda_i \), \( i = 1, 2, \ldots, n \) such that \( 0 < \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n < 1 \), then \( z(\omega, \eta) \) converges for all \((\omega, \eta)\) with

\[
0 < \omega \leq 2, \quad 0 < \eta < (\gamma + \lambda) / \lambda, \quad \text{where} \quad \gamma = \frac{(1 - \omega)}{\omega} \quad (5.4.44a)
\]
\[ 2 < \omega < 2/(1-\lambda), \quad 0 < \eta < (\gamma+\lambda)/\lambda. \quad (5.4.44b) \]

The region \( R \) described in (5.4.44) is sketched in figure 5.4.1(a).

Since \( z(\omega,1) = L_{\omega} \), the points \((\omega,1)\) for \( 0 < \omega < 2 \) are contained in \( R \). \( z(\omega,\eta) \) is convergent for \((\omega,\eta) \in R\) but it is not necessarily divergent for points outside the region \( R \).

![Figure 5.4.1: Region of convergence, (a) B-symmetric (b) B-skew symmetric.](image)

If \( A \) be skew-symmetric (i.e. \( A^T = -A \)), then we define the following lemma:

**Lemma 5.4.3**

For the matrix \( A \), if \( B = D^{-1}(L+U) \) is skew-symmetric and if \( \mu = \mu_1 < ... < \bar{\mu} = \mu_n \) are its eigenvalues then \( z(\omega,\eta) \) is convergent for
\[
0 < \omega < \frac{2}{1 + \mu}, \quad 0 < \eta \leq 1 \quad \text{(5.4.45a)}
\]
\[
\frac{2}{1 + \mu} \leq \omega < \frac{2}{\mu}, \quad 0 < \eta < \frac{2}{\omega} - \frac{\mu}{\mu} \quad \text{(5.4.45b)}
\]

For \( \mu > 1 \), no point \( \eta = \frac{1}{\omega} \) is contained in the region described by (5.4.45) as sketched in Figure 5.4.1b.

(ii) Cyclic case

To study the case of cyclic matrices we let

\[ A = I - D^{-1}(L+U) = I - B \]

where \( B \) is a \( P \)-cyclic, consistently ordered matrix with \( \rho(B) < 1 \) and \( \omega, \eta > 0 \). If \( \lambda \) is an eigenvalue of \( L_\omega \) and if \( \mu \) satisfies

\[(\lambda + \omega - 1)^P = \lambda^{P-1} \omega^p \mu^p, \quad \text{(5.4.46)}\]

then \( \mu \) is an eigenvalue of \( B \) and vice versa.

From equation (5.4.43), we get the relation

\[ \lambda(\omega) = \frac{1}{\eta} (\xi(\omega, \eta) + \eta - 1) \quad \text{,} \quad \eta \neq 0 \quad \text{(5.4.47)} \]

and, by inserting (5.4.47) into (5.4.46) we derive the following result (Sisler, 1973).

**Lemma 5.4.4**

Let \( A = I - B \) be a consistently ordered \( P \)-cyclic matrix. If \( \xi \neq (1-\eta) \) is an eigenvalue of \( z(\omega, \eta) \) and if \( \mu \) satisfies

\[(\xi - 1 + \omega \eta)^P = \eta(\xi + \eta - 1)^{P-1} \omega^p \mu^p, \quad \eta, \omega \neq 0, \quad \text{(5.4.48)}\]

then \( \mu \) is an eigenvalue of \( B \) and vice versa.

We are interested in finding the region \( R \) of convergence
for the 2-cyclic case. Let \( A = I - B \) be a consistently ordered 2-cyclic (i.e. \( P = 2 \)) positive definite matrix, for an eigenvalue \( \mu \) of \( B \) we assume

\[
0 \leq \mu^2 \leq \mu_2^2 \leq -\mu^2 = p^2(B) < 1.
\]

(5.4.49)

We now solve equation (5.4.46) for \( \lambda \) at \( P = 2 \), to obtain

\[
\lambda_{1,2}(\mu) = 1 - \omega + (\omega^2 \mu^2 \pm |\omega_1| \sqrt{\omega^2 \mu^2 + 4(1-\omega)/2})
\]

(5.4.50)

where \( \lambda_{1,2} \) denotes the two roots in (5.4.50), i.e. \( \lambda_1 \) and \( \lambda_2 \).

We introduce

\[
\omega' = \omega'(\mu) = 2/(1+\sqrt{1-\mu^2})
\]

(5.4.51)

for \( 0 < \omega \leq \omega' \) we have \( 0 \leq \lambda_{1,2}(\mu) < 1 \), for \( \omega > \omega' \) the values \( \lambda_{1,2}(\mu) \) become conjugate complex. For \( 0 < \omega \leq 1 \), and from (5.4.50) we have the result

\[
0 \leq \lambda_2(\mu) \leq \lambda_2(\mu) \leq \lambda_1(\mu) \leq \lambda_1(\mu) < 1.
\]

Theorem 5.4.9

Let \( A = I - B \) be a symmetric, positive definite matrix and let \( B \) be a weakly 2-cyclic consistently ordered matrix with eigenvalues \( \mu \) and \( 0 \leq \mu^2 \leq \mu_2^2 \leq -\mu^2 = p^2(B) < 1 \). Let \( \lambda_{1,2}(\mu) \) be the eigenvalues of \( L_\omega \) (by (5.4.50)) corresponding to \( \pm \mu \) of \( B \).

The region defined by \( R \) such that \( z(\omega,\eta) \) is convergent for \( (\omega,\eta) \in R \) is described by:

- (a) \( \omega < 0 \) \( \Rightarrow \), \( 2/(1 - \lambda_1(\mu)) < \eta < 0 \),
- (b) \( 0 < \omega < \omega'(\mu) \), \( 0 < \eta < 2/(1 - \lambda_2(\mu)) \),
- (c) \( \omega'(\mu) \leq \omega \leq 2 \), \( 0 < \eta < (2 - \omega^2) / \omega(1 - \mu) \).
Whenever the matrix $A$ is $n \times n$ with $n$ odd, then $\mu^2$ is necessarily zero, and often, for $n$ even, a positive lower bound $\mu^2$ for $\mu^2$ cannot be found. In practice, $\mu^2$ may be zero or very small for $n$ odd and only a small increase in the rate of convergence can be expected.

The region $R$ described by theorem 5.4.9 is sketched in figure 5.4.2 for $\mu^2 = 0.2$, $\frac{1}{\mu} = 0.8$.

Figure 5.4.2: Region of convergence - B symmetric.
For the case of skew-symmetric, we have the following theorem:

**Theorem 5.4.10**

Let \( A = I - B \) and let \( B \) be a skew-symmetric, consistently ordered weakly 2-cyclic matrix with eigenvalues \( \pm i\mu \) (\( \mu \) real) and \(-\rho^2(B) = -\overline{\mu}^2 < -\mu^2 \leq 0\). Let \( \lambda_2(\overline{\mu}) \) be the smaller of the two eigenvalues of \( L_\omega \) corresponding to the eigenvalues \( \pm i\mu \) of \( B \).

Then the region \( R \) of convergence of \( z(\omega, \eta) \) for \((\omega, \eta) \in R \) is described by

(a) \(-2/\overline{\mu}^2 < \omega < 0\), \(2 + \omega \overline{\mu}^2)/\omega(1 + \overline{\mu}^2) < \eta < 0\),

(b) \(0 < \omega \leq \omega_0 = 2/(1 + \sqrt{1 + \overline{\mu}^2})\), \(0 < \eta < (2 + \omega \overline{\mu}^2)/\omega(1 + \overline{\mu}^2)\),

(c) \(\omega_0 < \omega\), \(0 < \eta < 2/(1 - \lambda_2(\overline{\mu}))\).

The region \( R \) is sketched in figure 5.4.3 for \( \overline{\mu}^2 = 0.8 \).

\[\text{Figure 5.4.3 : Region of Convergence B skew-symmetric.}\]
Optimal Parameters

Let us denote the optimal parameters $\omega_{\text{opt.}}, \eta_{\text{opt.}}$ by $\omega_b, \eta_b$ respectively. $\omega_b$ and $\eta_b$ maximizes the rate of convergence of the two-parametric method. Since the negative values of $\omega$ and $\eta$ are only of theoretical interest, we concentrate here on the positive values of $\omega$ and $\eta$.

Theorem 5.4.11

Let $A = I - B$ be symmetric positive definite and $B$ be a weakly 2-cyclic consistently ordered matrix such that $B$ is singular with eigenvalues $\mu$ and $0 \leq \mu^2 \leq \overline{\mu}^2 = \rho^2(B) < 1$, then there is a unique pair of optimal parameters, $\omega_b, \eta_b$;

$$\omega_b = \frac{2}{1 + \sqrt{1 - \overline{\mu}^2}} \quad , \quad \eta_b = 1 \quad \text{and}$$

$$\rho(\varphi(\omega_b, 1)) = \omega_b - 1 = \frac{1 - \sqrt{1 - \overline{\mu}^2}}{1 + \sqrt{1 - \overline{\mu}^2}} \quad .$$

Now we consider the skew-symmetric matrices $B$:

Theorem 5.4.12

Under the assumptions of theorem 5.4.10, the optimal parameters are

$$\omega_b = \frac{2}{1 + \sqrt{1 + \mu^2}} \quad , \quad \eta_b = 1 \quad \text{and}$$

$$\rho(\varphi(\omega_b, 1)) = 1 - \omega_b = \frac{1 - \sqrt{1 + \mu^2}}{1 + \sqrt{1 + \mu^2}} \quad .$$

Remark

If $A = I - B$ is a symmetric positive definite consistently ordered 2-cyclic matrix and if 0 is an eigenvalue of $B$, then the
two-parametric method is not superior to the SOR. If $0 < \mu^2 \leq \mu^2$
for all eigenvalues $\mu$ of $B$ then there is a non-empty region $\overline{R} \subset R$
such that $\rho(z(\omega,\eta)) < \rho(z(\omega_b,1)) = \rho(L_{\omega_b})$ for all $(\omega,\eta) \in \overline{R}$.

Clearly, $(\omega_b,1)$ lies on the boundary of $\overline{R}$. In most applications, $\mu^2$ is very small and close to zero for $n$ odd.
5.5 MILDLY NONLINEAR ELLIPTIC EQUATIONS AND APPLICATIONS.

We recall the theory in section 4.9.1 which explained our strategy to solve mildly nonlinear elliptic equations iteratively. Here we apply the same strategy, i.e. the inner-outer iteration strategy (see Figure 4.9.1). Again we use the mixed convergence test between every two (or more) consecutive iterations, i.e.

\[
\frac{|u_{ij}^{(k+1)} - u_{ij}^{(k)}|}{1 + |u_{ij}^{(k)}|} < \varepsilon_{l}, \quad l = 1, 2, \quad (5.5.1)
\]

for every \(i, j = 1, 2, \ldots, n\), where \(k\) represents the \(k\th\) iterate in either the inner or the outer iteration and \(\varepsilon_1 = 10^{-3}\) for the inner iteration and \(\varepsilon_2 = 5 \times 10^{-6}\) for the outer iteration.

We apply the three methods described earlier, namely: the ESOR, AOR and Two-Parametric methods to the four mildly nonlinear problems, discussed earlier.

The problems are again defined on a unit square region \(R\) with boundary \(C\). The four problems are:

(1) \(\nabla^2 u = u^2\) \hspace{1cm} (5.5.2)
    \[\text{with } u = 10 \text{ on } C.\]

(2) \(\nabla^2 u = u^4\) \hspace{1cm} (5.5.3)
    \[\text{with } u = 0 \text{ on } C.\]

(3) \(\nabla^2 u = e^u\) \hspace{1cm} (5.5.4)
    \[\text{with } u = 0 \text{ on } C.\]

(4) \(\nabla^2 u = -\lambda^2 \sin u, \lambda = 1\) \hspace{1cm} (5.5.5)
    \[\text{with } u = 0 \text{ on } C.\]
We obtain the optimal values of the relaxation parameter and the acceleration parameters for all the methods experimentally by a multi-dimensional search procedure, i.e. we solve the problem for a range of values for the first of the two parameters (corresponds to each method) while fixing the second parameter. Then the whole procedure is applied conversely to the first parameter. For each case we choose the value which gives the lowest number of iterations. The tables (5.5.1) to (5.5.4) contain the results of applying the ESOR, AOR and two-parametric methods to the given test problems. The graphs in figures (5.5.1) to (5.5.4) represent the results in these tables, and show the number of iterations versus the number of mesh points (i.e. $h^{-1}$).

**Computational Complexity.**

Here we give the computational complexity for the three methods when applied to a mildly nonlinear elliptic partial differential equation. We notice that all the methods have the same computational complexity (i.e. multiplications and additions) per iteration which is

$$3n^2 \text{ multiplications} + 9n^2 \text{ additions} + \text{r.h.s. unit},$$

(5.5.6)

where $n^2$ is the number of mesh points in the region $R$. The tables (5.5.5) to (5.5.8) contain the computational complexity where $M$ and $A$ denote the multiplication and the addition operations respectively.

From the experimental results in tables (5.5.1) to (5.5.4), and since all the three methods have the same computational
complexity, it can be noticed that the ESOR and the AOR methods are closely similar for the given four test problems in terms of number of inner and outer iterations.

For the two-parametric method, it is observed that the optimum value of $\omega$ rises quickly to 1.5 for moderate or larger $h^{-1}$ when applied to the four problems. This method is not better than the other two methods but is not worse.

Furthermore, the three methods are convergent for the four problems and the increase in the number of iterations is nearly monotonic for increasing $h^{-1}$ which is clearly presented in the graphs depicted by figures (5.5.1) to (5.5.4) where the regression lines closely coincide or are nearly parallel. The slope of the line corresponding to the first problem is less than 1 (about 0.6) and the slope of these lines corresponding to the rest of the problems is 1. We conclude that the three methods are more effective on problems of the first test problem type.
<table>
<thead>
<tr>
<th>$h^{-1}$</th>
<th>Method</th>
<th>$\omega_{\text{exp}}$</th>
<th>$\tau$</th>
<th>No. of iterations</th>
<th>$\omega_{\text{exp}}$</th>
<th>$\tau$</th>
<th>No. of iterations</th>
<th>$\omega_{\text{exp}}$</th>
<th>$\eta$</th>
<th>No. of iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>ESOR</td>
<td>1.52</td>
<td>1.41</td>
<td>188</td>
<td>28</td>
<td>1.52</td>
<td>1.5</td>
<td>181</td>
<td>27</td>
<td>1.41</td>
</tr>
<tr>
<td>20</td>
<td>ESOR</td>
<td>1.69</td>
<td>1.68</td>
<td>284</td>
<td>37</td>
<td>1.7</td>
<td>1.68</td>
<td>282</td>
<td>34</td>
<td>1.4</td>
</tr>
<tr>
<td>30</td>
<td>ESOR</td>
<td>1.78</td>
<td>1.76</td>
<td>360</td>
<td>47</td>
<td>1.67</td>
<td>1.65</td>
<td>402</td>
<td>49</td>
<td>1.5</td>
</tr>
<tr>
<td>40</td>
<td>ESOR</td>
<td>1.81</td>
<td>1.77</td>
<td>425</td>
<td>52</td>
<td>1.81</td>
<td>1.83</td>
<td>421</td>
<td>56</td>
<td>1.48</td>
</tr>
<tr>
<td>50</td>
<td>ESOR</td>
<td>1.83</td>
<td>1.8</td>
<td>480</td>
<td>64</td>
<td>1.83</td>
<td>1.71</td>
<td>506</td>
<td>70</td>
<td>1.5</td>
</tr>
<tr>
<td>60</td>
<td>ESOR</td>
<td>1.87</td>
<td>1.8</td>
<td>529</td>
<td>76</td>
<td>1.86</td>
<td>1.84</td>
<td>515</td>
<td>70</td>
<td>1.4</td>
</tr>
<tr>
<td>80</td>
<td>ESOR</td>
<td>1.89</td>
<td>1.88</td>
<td>605</td>
<td>112</td>
<td>1.85</td>
<td>1.88</td>
<td>610</td>
<td>122</td>
<td>1.4</td>
</tr>
</tbody>
</table>

Number of iterations versus the acceleration parameters for $\nabla^2 u = u^2$, with initial guess 7.

*Table (5.5.1)*
<table>
<thead>
<tr>
<th>Method $h^{-1}$</th>
<th>ESOR $\omega_{exp}$</th>
<th>$\tau$</th>
<th>No. of iterations</th>
<th>AOR $\omega_{exp}$</th>
<th>$\tau$</th>
<th>No. of iterations</th>
<th>TWO-PARAMETRIC $\omega_{exp}$</th>
<th>$\eta$</th>
<th>No. of iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1.55</td>
<td>1.45</td>
<td>18</td>
<td>1.5</td>
<td>1.58</td>
<td>19</td>
<td>1.35</td>
<td>1.16</td>
<td>19</td>
</tr>
<tr>
<td>20</td>
<td>1.73</td>
<td>1.65</td>
<td>38</td>
<td>1.68</td>
<td>1.74</td>
<td>39</td>
<td>1.4</td>
<td>1.25</td>
<td>41</td>
</tr>
<tr>
<td>30</td>
<td>1.80</td>
<td>1.75</td>
<td>56</td>
<td>1.75</td>
<td>1.8</td>
<td>56</td>
<td>1.5</td>
<td>1.21</td>
<td>59</td>
</tr>
<tr>
<td>40</td>
<td>1.85</td>
<td>1.8</td>
<td>71</td>
<td>1.85</td>
<td>1.87</td>
<td>80</td>
<td>1.5</td>
<td>1.24</td>
<td>80</td>
</tr>
<tr>
<td>50</td>
<td>1.88</td>
<td>1.8</td>
<td>87</td>
<td>1.82</td>
<td>1.89</td>
<td>98</td>
<td>1.15</td>
<td>1.26</td>
<td>99</td>
</tr>
<tr>
<td>60</td>
<td>1.9</td>
<td>1.82</td>
<td>103</td>
<td>1.83</td>
<td>1.91</td>
<td>118</td>
<td>1.5</td>
<td>1.27</td>
<td>119</td>
</tr>
<tr>
<td>80</td>
<td>1.91</td>
<td>1.9</td>
<td>141</td>
<td>1.88</td>
<td>1.92</td>
<td>135</td>
<td>1.5</td>
<td>1.29</td>
<td>154</td>
</tr>
</tbody>
</table>

Number of iterations versus the acceleration parameters for $v^2u = u^4$, with initial guess 0.01.

Table (5.5.2)
<table>
<thead>
<tr>
<th>Method</th>
<th>ESOR</th>
<th>AOR</th>
<th>TWO-PARAMETRIC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\omega_{\text{exp}}$</td>
<td>$\tau$</td>
<td>No. of iterations</td>
</tr>
<tr>
<td></td>
<td>inner</td>
<td>outer</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1.56</td>
<td>1.51</td>
<td>25</td>
</tr>
<tr>
<td>20</td>
<td>1.67</td>
<td>1.75</td>
<td>44</td>
</tr>
<tr>
<td>30</td>
<td>1.79</td>
<td>1.72</td>
<td>52</td>
</tr>
<tr>
<td>40</td>
<td>1.84</td>
<td>1.80</td>
<td>72</td>
</tr>
<tr>
<td>50</td>
<td>1.88</td>
<td>1.80</td>
<td>90</td>
</tr>
<tr>
<td>60</td>
<td>1.90</td>
<td>1.80</td>
<td>108</td>
</tr>
<tr>
<td>80</td>
<td>1.92</td>
<td>1.85</td>
<td>136</td>
</tr>
</tbody>
</table>

Number of iterations versus the acceleration parameters for $v_u^2 = e^u$, with initial guess -0.01.

*Table (5.5.3)*
<table>
<thead>
<tr>
<th>Method</th>
<th>ESOR</th>
<th></th>
<th>AOR</th>
<th></th>
<th>TWO-PARAMETRIC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\omega_{\text{exp}}$</td>
<td>$\tau$</td>
<td>No. of iterations</td>
<td>$\omega_{\text{exp}}$</td>
<td>$\tau$</td>
</tr>
<tr>
<td></td>
<td>inner</td>
<td>outer</td>
<td></td>
<td>inner</td>
<td>outer</td>
</tr>
<tr>
<td>10</td>
<td>1.6</td>
<td>1.5</td>
<td>21</td>
<td>15</td>
<td>1.54</td>
</tr>
<tr>
<td>20</td>
<td>1.75</td>
<td>1.6</td>
<td>37</td>
<td>28</td>
<td>1.75</td>
</tr>
<tr>
<td>30</td>
<td>1.81</td>
<td>1.7</td>
<td>58</td>
<td>46</td>
<td>1.81</td>
</tr>
<tr>
<td>40</td>
<td>1.86</td>
<td>1.77</td>
<td>71</td>
<td>55</td>
<td>1.8</td>
</tr>
<tr>
<td>50</td>
<td>1.89</td>
<td>1.75</td>
<td>89</td>
<td>71</td>
<td>1.82</td>
</tr>
<tr>
<td>60</td>
<td>1.9</td>
<td>1.83</td>
<td>105</td>
<td>83</td>
<td>1.83</td>
</tr>
<tr>
<td>80</td>
<td>1.92</td>
<td>1.9</td>
<td>147</td>
<td>109</td>
<td>1.8</td>
</tr>
</tbody>
</table>

Number of iterations versus the acceleration parameters for $\nabla^2 u = \lambda^2 \sin u$, $\lambda = 1$, with initial guess 0.01.

Table (5.5.4)
Legend
- ESOR
- AOR
- TWO-PARAM.

Figure (5.5.1)
Figure (5.5.2)
Figure (5,5,3)

Figure (5,5,4)
<table>
<thead>
<tr>
<th>Method</th>
<th>ESOR</th>
<th>AOR</th>
<th>TWO-PARAMETRIC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M</td>
<td>A</td>
<td>M</td>
</tr>
<tr>
<td>10</td>
<td>564 n²</td>
<td>1692 n²</td>
<td>543 n²</td>
</tr>
<tr>
<td>20</td>
<td>852 n²</td>
<td>2556 n²</td>
<td>846 n²</td>
</tr>
<tr>
<td>30</td>
<td>1080 n²</td>
<td>3240 n²</td>
<td>1206 n²</td>
</tr>
<tr>
<td>40</td>
<td>1275 n²</td>
<td>3825 n²</td>
<td>1263 n²</td>
</tr>
<tr>
<td>50</td>
<td>1440 n²</td>
<td>4320 n²</td>
<td>1518 n²</td>
</tr>
<tr>
<td>60</td>
<td>1587 n²</td>
<td>4761 n²</td>
<td>1545 n²</td>
</tr>
<tr>
<td>80</td>
<td>1815 n²</td>
<td>5445 n²</td>
<td>1830 n²</td>
</tr>
</tbody>
</table>

Total computational work for \( V^2u = u^2 \) corresponding to table (5.5.1).

Table (5.6.5)
<table>
<thead>
<tr>
<th>Method $h^{-1}$</th>
<th>ESOR</th>
<th>AOR</th>
<th>TWO-PARAMETRIC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M</td>
<td>A</td>
<td>M</td>
</tr>
<tr>
<td>10</td>
<td>$54n^2$</td>
<td>$162n^2$</td>
<td>$57n^2$</td>
</tr>
<tr>
<td>20</td>
<td>$114n^2$</td>
<td>$342n^2$</td>
<td>$117n^2$</td>
</tr>
<tr>
<td>30</td>
<td>$168n^2$</td>
<td>$504n^2$</td>
<td>$168n^2$</td>
</tr>
<tr>
<td>40</td>
<td>$213n^2$</td>
<td>$639n^2$</td>
<td>$240n^2$</td>
</tr>
<tr>
<td>50</td>
<td>$261n^2$</td>
<td>$783n^2$</td>
<td>$294n^2$</td>
</tr>
<tr>
<td>60</td>
<td>$309n^2$</td>
<td>$927n^2$</td>
<td>$354n^2$</td>
</tr>
<tr>
<td>80</td>
<td>$423n^2$</td>
<td>$1269n^2$</td>
<td>$405n^2$</td>
</tr>
</tbody>
</table>

Total computational work for $V^2u = u^4$ corresponding to table (5.5.2).

**Table (5.5.6).**
<table>
<thead>
<tr>
<th>Method ( h^{-1} )</th>
<th>M ( n^2 )</th>
<th>A ( n^2 )</th>
<th>M ( n^2 )</th>
<th>A ( n^2 )</th>
<th>M ( n^2 )</th>
<th>A ( n^2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>75</td>
<td>225</td>
<td>75</td>
<td>225</td>
<td>81</td>
<td>243</td>
</tr>
<tr>
<td>20</td>
<td>132</td>
<td>396</td>
<td>117</td>
<td>351</td>
<td>129</td>
<td>387</td>
</tr>
<tr>
<td>30</td>
<td>156</td>
<td>468</td>
<td>156</td>
<td>468</td>
<td>171</td>
<td>513</td>
</tr>
<tr>
<td>40</td>
<td>216</td>
<td>648</td>
<td>237</td>
<td>711</td>
<td>252</td>
<td>756</td>
</tr>
<tr>
<td>50</td>
<td>270</td>
<td>810</td>
<td>279</td>
<td>837</td>
<td>300</td>
<td>900</td>
</tr>
<tr>
<td>60</td>
<td>324</td>
<td>972</td>
<td>327</td>
<td>981</td>
<td>375</td>
<td>1125</td>
</tr>
<tr>
<td>80</td>
<td>408</td>
<td>1224</td>
<td>441</td>
<td>1323</td>
<td>495</td>
<td>1485</td>
</tr>
</tbody>
</table>

Total computational work for \( V^2 u = e^u \) corresponding to table (5.5.3).

Table (5.5.7).
Total computational work for $\nabla^2 u = -\lambda^2 \sin u$, $\lambda = 1$, corresponding to table (5.5.4).

Table (5.5.8).
5.6  NEWTON-SOR METHOD FOR NONLINEAR EQUATIONS.

Nonlinear elliptic problems are generally solved by transforming them into an iterative sequence of linear problems which can each (or together) be solved by any solution method available for linear elliptic systems.

Consider the nonlinear system of equations

\[ Lu = f(u) , \]  

(5.6.1)

defined in a closed region \( \Omega \) with boundary \( \Gamma \), \( Lu \) is a linear elliptic partial differential operator. The system (5.6.1) has the exact solution \( \bar{u} \) and may be linearized by using Picard's method (Ames, 1965) which introduces a sequence of functions \( \{u^{(k)}\} \) that satisfy the given boundary conditions and the linear partial differential equation

\[ Lu^{(k+1)} = f(u^{(k)}) \]  

(5.6.2)

where \( f(u^{(k)}) \) is a constant vector obtained by evaluating \( f(u) \) in equation (5.6.1) at \( u^{(k)} \). It is well known that the sequence \( \{u^{(k)}\} \) from Picard's method converges (if it does) linearly, i.e.

as \( k \to \infty \), \( u^{(k+1)} - \bar{u} = 0 \) \( u^{(k)} - \bar{u} \).

Alternatively, if \( f(u) \) in (5.6.1) is differentiable, then it can be replaced by the Taylor expansion (Bellman, 1961) about \( u^{(k)} \), i.e.

\[ f(u^{(k)}) + \{u^{(k+1)} - u^{(k)}\} f'(u^{(k)}) . \]  

(5.6.3)

We now have the new system of equations

\[ Lu^{(k+1)} - f'(u^{(k)}) u^{(k+1)} = f(u^{(k)}) - u^{(k)} f'(u^{(k)}) . \]  

(5.6.4)
The sequence \( \{u^{(k)}\} \) in this case, when convergent is usually quadratically convergent (provided the guess solution is close to the exact solution) i.e.

\[
\text{as } k \to \infty, \quad u^{(k+1)} - u = 0 \left( u^{(k)} - u \right)^2,
\]

We may use the SOR method in conjunction with the Newton's method to solve the nonlinear system (5.6.1). We will have a composite Newton-SOR iteration with Newton's method as the primary iteration and the SOR method as the secondary iteration or a composite SOR-Newton iteration with the primary and the secondary iteration being swapped from above. Alternatively, additional new methods for nonlinear problems can be obtained by substituting for SOR in the above, the ESOR and AOR methods discussed earlier.

**Newton-SOR Method.**

Suppose we have the nonlinear mapping:

\[
F : \mathbb{R}^n \to \mathbb{R}^n \quad \text{(5.6.5)}
\]

which can be considered as a system of nonlinear equations in the form

\[
F(u) = 0 \quad \text{(5.6.6)}
\]

Assume that \( u^{(k)} \) has been determined, then we apply the Newton's method. The following linear system results

\[
F'(u^{(k)}) u^{(k+1)} = F'(u^{(k)}) u^{(k)} - F(u^{(k)}) \quad \text{(5.6.7)}
\]

The linear system (5.6.7) can be solved now by the SOR method. We apply the SOR method to the linear system

\[
Au = b \quad \text{(5.6.8)}
\]
where \( A \) is an \((n \times n)\) real matrix, \( u \) is an \((n \times 1)\) unknown vector and \( b \) is an \((n \times 1)\) constant vector, and \( A = D - L - U \) where \( D \), \( L \) and \( U \) are the diagonal, strictly lower and strictly upper matrices respectively. The SOR method can be written (see (4.3.16)) in the following form:

\[
\begin{align*}
\underline{u}^{(k+1)} &= (D - \omega L)^{-1} \left[ (1-\omega)D + \omega U \right] \underline{u}^{(k)} + \omega(D-\omega L)^{-1} b \\
\text{or} \quad \underline{u}^{(k+1)} &= \underline{u}^{(k)} - \omega(D-\omega L)^{-1}(Au^{(k)} - b) .
\end{align*}
\]  

(5.6.9)  
(5.6.10)

Now, by letting

\[
B = \omega^{-1}(D-\omega L), \quad C = \omega^{-1}[(1-\omega)D + \omega U] \quad \text{and} \quad H = B^{-1}C,
\]

equation (5.6.9) can be expressed in terms of \( \underline{u}^{(0)} \) as

\[
\begin{align*}
\underline{u}^{(k+1)} &= H \underline{u}^{(k)} + B^{-1}b = H^{k+1} \underline{u}^{(0)} + (H^{k} + H^{k-1} + \ldots + I)B^{-1}b \\
\text{or} \quad \underline{u}^{(k+1)} &= \underline{u}^{(0)} + (H^{k+1} - I)\underline{u}^{(0)} + (H^{k} + H^{k-1} + \ldots + I)B^{-1}(Au^{(0)} - Au^{(0)} + b).
\end{align*}
\]  

(5.6.11)

Thus, since

\[
B^{-1}A = B^{-1}(B-C) = I - H
\]

and

\[
(I + \ldots + H^{k-1} + H^{k})(I-H) = I - H^{k+1}
\]

then

\[
\begin{align*}
\underline{u}^{(k+1)} &= \underline{u}^{(0)} - \omega(H^{k} + H^{k-1} + \ldots + I)(D-\omega L)^{-1}(Au^{(0)} - b) .
\end{align*}
\]  

(5.6.12)

The Jacobian matrix in equation (5.6.7) is decomposed as

\[
F'(\underline{u}^{(k)}) = D_k - L_k - U_k ,
\]

where again \( D_k \), \( L_k \) and \( U_k \) are diagonal, strictly lower and strictly upper matrices respectively. We assume that \( D_k \) is nonsingular.
and we define
\[ H_k = (D_k - \omega L_k)^{-1} \left[ (1-\omega)D_k + \omega U_k \right]. \] (5.6.13)

Now, we apply the SOR method to equation (5.6.7) and denote the SOR iterate by \( u^{(k,m)} \). Then by equation (5.6.12), noting that, here \( A = F'(u^{(k)}) \) and \( b = F'(u^{(k)}) u^{(k)} - F(u^{(k)}) \), we have
\[
 u^{(k,m)}(k) = u^{(k,0)} - \omega \left[ H_k^{m-1} + \ldots + I \right] (D_k - \omega L_k)^{-1} \left[ F'(u^{(k)}) (u^{(k,0)} - u^{(k)}) + F(u^{(k)}) \right],
\] (5.6.14)
for \( m = 1, 2, \ldots \) is the total number of the SOR iterations at the \( k \)th stage. If we set \( u^{(k,0)} = u^{(k)} \) as the starting approximation for the SOR iteration and \( u^{(k,m)} = u^{(k+1)} \) then equation (5.6.14) reduces to
\[
 u^{(k+1)} = u^{(k)} - \omega (H_k^{m-1} + \ldots + I) (D_k - \omega L_k)^{-1} F(u^{(k)}). \] (5.6.15)

We call equation (5.6.15) the \( m \)-step general Newton-SOR iteration.

If we set \( m = 1 \) (the simplest choice) then equation (5.6.15) reduces to
\[
 u^{(k+1)} = u^{(k)} - \omega (D_k - \omega L_k)^{-1} F(u^{(k)}),
\] (5.6.16)
which we call the one-step Newton-SOR method.

**SOR - Newton Method.**

Suppose we have a nonlinear system of equations which can be written in the form
\[
 F(u) = 0
\] (5.6.17) with \( n \) equations of \( n \) variables.
Consider now the \( i \)th equation in the system (5.6.17) to be as

\[ F_i(u_i) = F_i(u_1^{(k+1)}, \ldots, u_i^{(k+1)}, u_i^{(k)}, u_{i+1}^{(k)}, \ldots, u_n^{(k)}) = 0, \]  

(5.6.18)

We solve this nonlinear equation of one variable \( u_i \), for \( u_i \) using Newton method,

\[ u_i^{(k+1)} = u_i^{(k)} - \frac{F_i(u_i^{(k)})}{\partial F_i(u_i^{(k)})/\partial u_i}, \]  

(5.6.19)

where \( \partial F_i / \partial u_i \).

Now, an application of the SOR iteration gives

\[ u_i^{(k+1)} = u_i^{(k)} + \omega(u_i^{(k)} - u_i^{(k)}) \]  

(5.6.20)

which, by using equation (5.6.19), will give

\[ u_i^{(k+1)} = u_i^{(k)} - \omega \frac{F_i(u_i^{(k)})}{\partial F_i(u_i^{(k)})/\partial u_i}. \]  

(5.6.21)

Equation (5.6.21) represents the SOR - Newton method which is identical to the nonlinear overrelaxation (NLOR) method described by Ames (1964, p.408). Equation (5.6.21) is applied to the remaining equations in the system (5.6.17), i.e. for all \( i = 1, 2, \ldots, n \).

We can summarize the two methods in the flowcharts in Figures (5.6.1) and (5.6.2).
Nonlinear Equations

By Newton Linearization

Linear Equations

By SOR method

Solution $u_i$

Figure (5.6.1) - Newton - SOR Method

Nonlinear Equation of One Variable

By Newton and GS methods

Solution $\hat{u}_i$

SOR Iteration

$u_i^{(k+1)} = u_i^{(k)} + \omega (\hat{u}_i - u_i^{(k)})$

Figure (5.6.2) - SOR - Newton Method
5.7 EXPERIMENTAL RESULTS.

Again we will apply the Newton - SOR method and the SOR - Newton method to the four test problems given in section 5.5.

We will now derive the applicable (computational) forms of the two methods.

1 - Newton - SOR Method.

The system (5.6.1), by the Newton linearization process may be written in the form

\[ L_{u}^{(k+1)} = f(u^{(k)}) + (u^{(k+1)} - u^{(k)}) f'_{u}(u^{(k)}) \]  

(5.7.1)

or

\[ L_{u}^{(k+1)} - f'_{u}(u^{(k)}) u^{(k+1)} = b^{(k)} \]  

(5.7.2)

where

\[ b^{(k)} = f(u^{(k)}) - u^{(k)} f'_{u}(u^{(k)}) \]  

(5.7.3)

Now, by using the five point difference replacement, i.e.

\[ L_{u} = \frac{1}{h^2} \left( u_{i+1,j} + u_{i,j-1} + u_{i,j+1} + 4u_{i,j} \right) \]

and by omitting the upper subscripts in (5.7.2) to leave it in a suitable form to apply the GS iteration. So equation (5.7.2) becomes

\[ u_{i,j} (4 + h^2 f'_{u}(u_{i,j})) = u_{i+1,j} + u_{i,j-1} + u_{i,j+1} + u_{i-1,j} + u_{i,j+1} - h^2 b \]

or

\[ u_{i,j} = \frac{1}{k} \left[ u_{i,j} + u_{i,j-1} + u_{i-1,j} + u_{i,j+1} - h^2 b \right] \]  

(5.7.4)

where

\[ k = 4 + h^2 f'_{u}(u_{i,j}) \]  

(5.7.5)
Equation (5.7.4) represents the Newton - GS iteration. Now, we introduce the parameter \( \omega \) (overrelaxation real parameter) to get the Newton - SOR iteration, we have

\[
    u_{i,j} = \frac{\omega}{k} \left[ u_{i+1,j} + u_{i,j-1} + u_{i-1,j} + u_{i,j+1} - h^2 b \right] - (\omega - 1) u_{i,j}
\]

Equation (5.7.6) is the computational form of this method. It is clear that when \( \omega = 1 \) the method reduces to the Newton - GS method.

The Computational Complexity.

The number of arithmetic operations involved per iteration are

\[ 2n^2 \text{ multiplications} + 5n^2 \text{ additions} + \text{r.h.s. unit} \]

where \( n^2 \) is the number of mesh points. The r.h.s. unit here includes the computation in the outer iteration to compute \( b \) and \( k \) as described in the formulas (5.7.3) and (5.7.5) respectively.

2 - SOR - Newton Method.

The system (5.6.1) can be written in the form

\[
    Lu - F(u) = 0
\]

which can be represented by

\[
    F(u) = 0
\]

Using the five point finite difference formula for the linear operator \( Lu \), we obtain
\[ F(u) = \frac{1}{h^2} \left[ u_{i+1,j} + u_{i,j-1} + u_{i-1,j} + u_{i,j+1} - 4u_{i,j} \right] - f(u_{i,j}) = 0. \]  
(5.7.9)

Recalling the SOR - Newton iteration applied to our case, we have
\[ u_{i,j}^{(k+1)} = u_{i,j}^{(k)} - \omega \frac{F_i(u_{ij}^{(k)})}{\partial F_i/u_{ij}^{(k)}}. \]  
(5.7.10)

where \( F_i(u_{ij}) \) is the \( i \)th equation of the system (5.7.9) and \( \partial F_i/u_{ij} \) is its first derivative with respect to \( u_{i,j} \).

To carry out the iteration (5.7.10), we evaluate the function (5.7.9) by applying the GS idea of using the most recently obtained values of \( u_{i,j} \). In this case, \( \partial F_i/u_{ij} \) will be
\[ \partial F_i(u_{ij}) = -\frac{4}{h^2} - \frac{df}{du} (u_{ij}). \]  
(5.7.11)

Again when \( \omega = 1 \), this method reduces to the GS - Newton method.

The Computational Complexity.

The computational complexity of the SOR - Newton method has the form
\[ 2n^2 \text{ multiplications} + 7n^2 \text{ additions} + \text{r.h.s. unit}. \]
The r.h.s. unit here includes the computation involved in evaluating \( f(u_{ij}) \) in equation (5.7.9) and \( \frac{df}{du} (u_{ij}) \) in equation (5.7.11).

The above two methods were implemented in FORTRAN 77 programs according to the convergence tests described earlier, the inner-outer iteration strategy (described in section 5.5) for the
Newton - SOR method and just the inner iteration for the SOR - Newton method.

Tables (5.7.1) to (5.7.4) contain the results of applying the two methods to the given test problems. We notice that the SOR - Newton has nearly the same number of iterations when applied to the four problems despite the acceleration parameters being different corresponding to each value of $h^{-1}$. The Newton - SOR method has close values of the numbers of iterations but they are not as close as in the case of the SOR - Newton method.

Tables (5.7.5) to (5.7.8) contain the computational complexity of the two methods corresponding to the results given in tables (5.7.1) to (5.7.4) which show that the SOR - Newton method is better than the Newton - SOR method for the first problem, and worse for the remaining problems in terms of the total number of computational operations (multiplications and additions) needed in each iteration.

Figures (5.7.1) to (5.7.4) shows that the slope of the regression lines is nearly 1, which confirms the convergence of the methods as predicted by the SOR theory.
<table>
<thead>
<tr>
<th>$h^{-1}$</th>
<th>$\omega_{\text{exp}}$</th>
<th>Newton - SOR</th>
<th>SOR - Newton</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number of Iterations</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\omega_{\text{exp}}$</td>
<td>inner</td>
<td>outer</td>
</tr>
<tr>
<td>10</td>
<td>1.42</td>
<td>36</td>
<td>13</td>
</tr>
<tr>
<td>20</td>
<td>1.65</td>
<td>68</td>
<td>23</td>
</tr>
<tr>
<td>30</td>
<td>1.75</td>
<td>99</td>
<td>35</td>
</tr>
<tr>
<td>40</td>
<td>1.80</td>
<td>129</td>
<td>49</td>
</tr>
<tr>
<td>50</td>
<td>1.83</td>
<td>153</td>
<td>58</td>
</tr>
<tr>
<td>60</td>
<td>1.86</td>
<td>187</td>
<td>68</td>
</tr>
<tr>
<td>80</td>
<td>1.88</td>
<td>234</td>
<td>97</td>
</tr>
</tbody>
</table>

Table (5.7.1) : $V^2u = u^2$ with initial guess 7.
<table>
<thead>
<tr>
<th>$h^{-1}$</th>
<th>$\omega_{\text{exp}}$</th>
<th>Number of Iterations</th>
<th>$\omega_{\text{exp}}$</th>
<th>Number of Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>inner</td>
<td>outer</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1.52</td>
<td>20</td>
<td>13</td>
<td>1.52</td>
</tr>
<tr>
<td>20</td>
<td>1.71</td>
<td>40</td>
<td>30</td>
<td>1.71</td>
</tr>
<tr>
<td>30</td>
<td>1.79</td>
<td>60</td>
<td>45</td>
<td>1.79</td>
</tr>
<tr>
<td>40</td>
<td>1.83</td>
<td>80</td>
<td>61</td>
<td>1.83</td>
</tr>
<tr>
<td>50</td>
<td>1.86</td>
<td>85</td>
<td>60</td>
<td>1.86</td>
</tr>
<tr>
<td>60</td>
<td>1.88</td>
<td>100</td>
<td>70</td>
<td>1.9</td>
</tr>
<tr>
<td>80</td>
<td>1.91</td>
<td>129</td>
<td>87</td>
<td>1.91</td>
</tr>
</tbody>
</table>

*Table (5.7.2): $\nabla^2 u = u^4$ with initial guess 0.01.*
<table>
<thead>
<tr>
<th>$h^{-1}$</th>
<th>$\omega_{\exp}$</th>
<th>Number of Iterations</th>
<th>$\omega_{\exp}$</th>
<th>Number of Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>inner</td>
<td>outer</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1.54</td>
<td>21</td>
<td>10</td>
<td>1.55</td>
</tr>
<tr>
<td>20</td>
<td>1.72</td>
<td>39</td>
<td>22</td>
<td>1.73</td>
</tr>
<tr>
<td>30</td>
<td>1.79</td>
<td>58</td>
<td>36</td>
<td>1.80</td>
</tr>
<tr>
<td>40</td>
<td>1.84</td>
<td>76</td>
<td>50</td>
<td>1.84</td>
</tr>
<tr>
<td>50</td>
<td>1.87</td>
<td>95</td>
<td>65</td>
<td>1.88</td>
</tr>
<tr>
<td>60</td>
<td>1.89</td>
<td>114</td>
<td>80</td>
<td>1.9</td>
</tr>
<tr>
<td>80</td>
<td>1.91</td>
<td>144</td>
<td>101</td>
<td>1.91</td>
</tr>
</tbody>
</table>

Table (5.7.3): $\nabla^2 u = e^u$ with initial guess $-0.01$. 
<table>
<thead>
<tr>
<th>$h^{-1}$</th>
<th>$\omega_{\text{exp}}$</th>
<th>Number of Iterations</th>
<th>$\omega_{\text{exp}}$</th>
<th>Number of Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>inner</td>
<td>outer</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1.53</td>
<td>20</td>
<td>13</td>
<td>1.53</td>
</tr>
<tr>
<td>20</td>
<td>1.75</td>
<td>41</td>
<td>29</td>
<td>1.73</td>
</tr>
<tr>
<td>30</td>
<td>1.79</td>
<td>60</td>
<td>46</td>
<td>1.79</td>
</tr>
<tr>
<td>40</td>
<td>1.86</td>
<td>81</td>
<td>56</td>
<td>1.85</td>
</tr>
<tr>
<td>50</td>
<td>1.86</td>
<td>88</td>
<td>64</td>
<td>1.86</td>
</tr>
<tr>
<td>60</td>
<td>1.88</td>
<td>104</td>
<td>74</td>
<td>1.88</td>
</tr>
<tr>
<td>80</td>
<td>1.91</td>
<td>131</td>
<td>89</td>
<td>1.91</td>
</tr>
</tbody>
</table>

Table (5.7.4): $\nabla^2 u = -\lambda^2 \sin u$, $\lambda = 1$ with initial guess 0.01.
Problem 1

Legend

- Newton-SOR
- SOR-Newton

1000

100

10

1

h⁻¹

Figure (5.7.1): \( \nabla^2 u = u^2 \)

Problem 2

Legend

- Newton-SOR
- SOR-Newton

1000

100

10

1

h⁻¹

Figure (5.7.2): \( \nabla^2 u = u^4 \)
Problem 3

Legend
- Newton-SOR
- SOR-Newton

Figure (5.7.3): $v^2 u = e^u$

Problem 4

Legend
- Newton-SOR
- SOR-Newton

Figure (5.7.4): $v^2 u = -\lambda^2 \sin(u), \lambda = 1$
<table>
<thead>
<tr>
<th>$h^{-1}$</th>
<th>Newton - SOR</th>
<th>SOR - Newton</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M</td>
<td>A</td>
</tr>
<tr>
<td>10</td>
<td>$72n^2$</td>
<td>$180n^2$</td>
</tr>
<tr>
<td>20</td>
<td>$136n^2$</td>
<td>$340n^2$</td>
</tr>
<tr>
<td>30</td>
<td>$198n^2$</td>
<td>$495n^2$</td>
</tr>
<tr>
<td>40</td>
<td>$258n^2$</td>
<td>$645n^2$</td>
</tr>
<tr>
<td>50</td>
<td>$306n^2$</td>
<td>$765n^2$</td>
</tr>
<tr>
<td>60</td>
<td>$374n^2$</td>
<td>$935n^2$</td>
</tr>
<tr>
<td>80</td>
<td>$468n^2$</td>
<td>$1170n^2$</td>
</tr>
</tbody>
</table>

Total computational work for $\nabla^2 u = u^2$, corresponding to table (5.7.1)

Table (5.7.5)
<table>
<thead>
<tr>
<th>h^{-1}</th>
<th>\text{Newton - SOR}</th>
<th>\text{SOR - Newton}</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M</td>
<td>A</td>
</tr>
<tr>
<td>10</td>
<td>40 , n^2</td>
<td>100 , n^2</td>
</tr>
<tr>
<td>20</td>
<td>80 , n^2</td>
<td>200 , n^2</td>
</tr>
<tr>
<td>30</td>
<td>120 , n^2</td>
<td>300 , n^2</td>
</tr>
<tr>
<td>40</td>
<td>160 , n^2</td>
<td>400 , n^2</td>
</tr>
<tr>
<td>50</td>
<td>170 , n^2</td>
<td>425 , n^2</td>
</tr>
<tr>
<td>60</td>
<td>200 , n^2</td>
<td>500 , n^2</td>
</tr>
<tr>
<td>80</td>
<td>258 , n^2</td>
<td>645 , n^2</td>
</tr>
</tbody>
</table>

Total computational work for $V^2 u = u^4$, corresponding to table (5.7.2).

\textit{Table (5.7.6)}
<table>
<thead>
<tr>
<th>h^{-1}</th>
<th>Newton - SOR</th>
<th>SOR - Newton</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M</td>
<td>A</td>
</tr>
<tr>
<td>10</td>
<td>42 , n^2</td>
<td>105 , n^2</td>
</tr>
<tr>
<td>20</td>
<td>78 , n^2</td>
<td>195 , n^2</td>
</tr>
<tr>
<td>30</td>
<td>116 , n^2</td>
<td>290 , n^2</td>
</tr>
<tr>
<td>40</td>
<td>152 , n^2</td>
<td>380 , n^2</td>
</tr>
<tr>
<td>50</td>
<td>190 , n^2</td>
<td>475 , n^2</td>
</tr>
<tr>
<td>60</td>
<td>228 , n^2</td>
<td>570 , n^2</td>
</tr>
<tr>
<td>80</td>
<td>288 , n^2</td>
<td>720 , n^2</td>
</tr>
</tbody>
</table>

Total computational work for $v^2u = e^u$, corresponding to table (5.7.3).

Table (5.7.7)
<table>
<thead>
<tr>
<th>( h^{-1} )</th>
<th>( \text{M} )</th>
<th>( \text{A} )</th>
<th>( \text{M} )</th>
<th>( \text{A} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>40 ( n^2 )</td>
<td>100 ( n^2 )</td>
<td>40 ( n^2 )</td>
<td>140 ( n^2 )</td>
</tr>
<tr>
<td>20</td>
<td>82 ( n^2 )</td>
<td>205 ( n^2 )</td>
<td>82 ( n^2 )</td>
<td>287 ( n^2 )</td>
</tr>
<tr>
<td>30</td>
<td>120 ( n^2 )</td>
<td>300 ( n^2 )</td>
<td>120 ( n^2 )</td>
<td>420 ( n^2 )</td>
</tr>
<tr>
<td>40</td>
<td>162 ( n^2 )</td>
<td>405 ( n^2 )</td>
<td>162 ( n^2 )</td>
<td>567 ( n^2 )</td>
</tr>
<tr>
<td>50</td>
<td>176 ( n^2 )</td>
<td>440 ( n^2 )</td>
<td>200 ( n^2 )</td>
<td>700 ( n^2 )</td>
</tr>
<tr>
<td>60</td>
<td>208 ( n^2 )</td>
<td>520 ( n^2 )</td>
<td>240 ( n^2 )</td>
<td>840 ( n^2 )</td>
</tr>
<tr>
<td>80</td>
<td>262 ( n^2 )</td>
<td>655 ( n^2 )</td>
<td>320 ( n^2 )</td>
<td>1120 ( n^2 )</td>
</tr>
</tbody>
</table>

Total computational work for \( \nabla^2 u = - \lambda^2 \sin u, \lambda = 1 \), corresponding to table (5.7.4).

Table (5.7.8)
Chapter Six

Alternating Direction Implicit Methods

6.1 Introduction
   6.1.1 The Matrix Problem
   6.1.2 Basic ADI Operators

6.2 The Stationary Case
   6.2.1 Consistency and Convergence
   6.2.2 Selection of Parameters

6.3 The Commutative Case

6.4 Good Acceleration Parameters

6.5 The Peaceman-Rachford (P-R) Method
   6.5.1 Introduction
   6.5.2 Choice of Parameters and Convergence

6.6 The Douglas-Rachford (D-R) Method
   6.6.1 Introduction
   6.6.2 Choice of Parameters and Convergence

6.7 Mildly Nonlinear Problems and Applications
   6.7.1 The Model Problem
   6.7.2 Method of Solution
   6.7.3 Application and Results
   6.7.4 Discussion and Comments
6.1 INTRODUCTION.

Alternating direction implicit methods (or ADI methods) constitute powerful techniques for solving elliptic and parabolic partial difference equations. An extremely rapid convergence can be realized for certain model and other problems in rectangular regions but it may suffer from lack of generality when used by itself. This can be overcome by combining an ADI inner iteration with an outer Lanczo's type iteration (Wachspress, 1966).

6.1.1 The Matrix Problem.

Consider the self-adjoint partial differential equation

\[ gu - \frac{\partial}{\partial x} \left[ a \frac{\partial u}{\partial x} \right] - \frac{\partial}{\partial y} \left[ b \frac{\partial u}{\partial y} \right] = s \] (6.1.1)

where \( g \) (nonnegative), \( a \) and \( b \) (positive) are functions of \( x \) and \( y \). The solution \( u = u(x,y) \) is defined in a closed region \( R \) of boundary \( \mathcal{C} \). To find an approximate solution of equation (6.1.1), we cover the region \( R \) with a rectangular mesh having mesh lengths \( h \) and \( k \).

We now take the values \( u(x_i, y_j) \) of \( u \) for the interior meshpoints as unknowns. We approximate

\[ -hk \frac{\partial}{\partial x} \left[ a \frac{\partial u}{\partial x} \right] \] by \( H \) and

\[ -hk \left[ b \frac{\partial u}{\partial y} \right] \] by \( V \), where \( H \) and \( V \) are difference operators of the form,

\[ Hu(x,y) = -a_1(x,y) \ u(x+h,y) + 2b_1(x,y)u(x,y) - \gamma_1(x,y) \ u(x-h,y) \] (6.1.2)

\[ Vu(x,y) = -a_2(x,y) \ u(x,y+h) + 2b_2(x,y)u(x,y) - \gamma_2(x,y) \ u(x,y-k) \] (6.1.3)
The most common choices (see Birkhoff et al., 1962) for \( \alpha_1, \beta_1, \gamma_1, \alpha_2, \beta_2 \) and \( \gamma_2 \) are

\[
\alpha_1 = k \frac{a(x+h/2, y)}{h}, \quad \gamma_1 = k \frac{a(x-h/2, y)}{h}, \quad 2\beta_2 = \alpha_1 + \gamma_1,
\]

(6.1.4)

\[
\alpha_2 = h \frac{c(x, y+k/2)}{k}, \quad \gamma_2 = h \frac{c(x, y-k/2)}{k}, \quad 2\beta_2 = \alpha_2 + \gamma_2.
\]

(6.1.5)

\( H \) and \( V \) now become symmetric matrices. We will consider the case \( h = k > 0 \) of a square network. The equation (6.1.1) can be replaced, after using the finite difference approximation technique, by the system of linear equations

\[
(H + V + E) \mathbf{u} = \mathbf{b},
\]

(6.1.6)

where \( E \) is a nonnegative diagonal matrix whose \( \ell \text{th} \) diagonal entry associated with the interior mesh point \( u_\ell = (x_i, y_j) \), is \( h^2 g(x_i, y_j) \), the vector \( \mathbf{b} \) is computed by adding the terms in (6.1.2) and (6.1.3) associated with points on the boundary \( \partial \Omega \) of \( \Omega \) to the source terms \( h^2 s(x_i, y_j) \), i.e. the right hand side.

To obtain a larger rate of convergence, the matrices \( E, H \) and \( V \) should have particular properties. \( H \) and \( V \) have positive diagonal elements and nonpositive off-diagonal elements, they are diagonally dominant and hence they are positive definite. By ordering the mesh-points by rows, \( H \) becomes tridiagonal and by ordering them by columns, \( V \) becomes tridiagonal. \( H \) and \( V \) cannot be both tridiagonal simultaneously.
6.1.2 Basic ADI Operators.

Equation (6.1.6) is equivalent, for any matrices $D$ and $E$, to each of the following equations:

$$
(H + \Sigma + D) \mathbf{u} = \mathbf{b} - (V - D) \mathbf{u}
$$

(6.1.7)

$$
(V + \Sigma + E) \mathbf{u} = \mathbf{b} - (H - E) \mathbf{u}
$$

(6.1.8)

provided $(H + \Sigma + D)$ and $(H + \Sigma + E)$ are nonsingular. Peaceman and Rachford (1955) considered the case $\Sigma = 0$, $D = E = \rho I$ a scalar matrix, and where $\rho$ is a real positive parameter, so equations (6.1.7) and (6.1.8) reduce to

$$
(H + \rho I) \mathbf{u} = \mathbf{b} - (V - \rho I) \mathbf{u}
$$

(6.1.9)

$$
(V + \rho I) \mathbf{u} = \mathbf{b} - (H - \rho I) \mathbf{u}
$$

(6.1.10)

To solve equation (6.1.6) Peaceman and Rachford proposed to choose a sequence of positive numbers $\rho$ and calculate the sequence of vectors $\mathbf{u}^{(k)}$, $\mathbf{u}^{(k+1)}$ by the following formulas and by the use of the sequence of matrices $D_k = E_k = \rho_k I$, $k = 0, 1, \ldots$

$$
(H + \Sigma + D_k) \mathbf{u}^{(k+1)} = \mathbf{b} - (V - D_k) \mathbf{u}^{(k)}
$$

(6.1.11)

$$
(V + \Sigma + E_k) \mathbf{u}^{(k+1)} = \mathbf{b} - (H - E_k) \mathbf{u}^{(k+1)}
$$

(6.1.12)

Since by definition the matrices to be inverted are nonsingular, Peaceman and Rachford suggested the choice $D_k = \rho_k I$ and $E_k = \beta_k I$ which will define what is called the Peaceman - Rachford (P-R) method.

$$
\mathbf{u}^{(k+1)} = (H + \Sigma + \rho_k I)^{-1} \left[ \mathbf{b} - (V - \rho_k I) \mathbf{u}^{(k)} \right]
$$

(6.1.13)
\[ u^{(k+1)} = (V + \Sigma + \tilde{\rho}_k I)^{-1} \left[ b - (H - \tilde{\rho}_k I) u^{(k+1)} \right], \quad (6.1.14) \]

where \( \rho_k \) and \( \tilde{\rho}_k \) are real positive parameters.

Douglas and Rachford (1956) considered a more general case, i.e. \( \Sigma \geq 0 \), so from equations (6.1.11) and (6.1.12) we obtain after some rearrangements

\[ u^{(k+1)} = (H_1 + \rho_k I)^{-1} \left[ b - (V_1 - \rho_k I) u^{(k)} \right], \quad (6.1.15) \]

\[ u^{(k+1)} = (V_1 + \rho_k I)^{-1} \left[ V_1 u_k + \rho_k u^{(k+1)} \right], \quad (6.1.16) \]

where \( H_1 = H + \frac{1}{2} \Sigma \), \( V_1 = V + \frac{1}{2} \Sigma \) and with setting

\[ D_k = E_k = \rho_k I - \frac{1}{2} \Sigma. \]

The above method is called the \textit{Douglas-Rachford (D-R) method}.

A general variant of the basic Peaceman-Rachford method which can reduce to the Douglas-Rachford method, can be obtained by introducing a new parameter \( \omega \) in equation (6.1.14) such that

\[ (H + \rho_k I) u^{(k+1)} = b - (V - \rho_k I) u^{(k)} \quad (6.1.17) \]

\[ (V + \rho_k I) u^{(k+1)} = (V - (1-\omega)\rho_k I) u^{(k)} + (2-\omega)\rho_k u^{(k+1)}. \quad (6.1.18) \]

For \( \omega = 0 \), we have the Peaceman-Rachford method and for \( \omega = 1 \), we have the Douglas-Rachford method where \( H \) and \( V \) will be equivalent to \( H_1 \) and \( V_1 \) respectively.

6.2 \textbf{THE STATIONARY CASE}.

In this section we study the stationary case, i.e. \( D_k = D \) and \( E_k = E \) are independent of \( k \), so that \( \rho_k = \rho \) and \( \tilde{\rho}_k = \tilde{\rho} \) are independent of \( k \).
6.2.1 Consistency and Convergence.

For the P-R method (and similarly for D-R method), by eliminating $u^{(n+1)}$ in equations (6.1.13) and (6.1.14) we obtain

$$u^{(k+1)} = T_{\rho, \tilde{\rho}} u^{(k)} + k$$

(6.2.1)

where

$$T_{\rho, \tilde{\rho}} = (V + \Sigma + \tilde{\rho}I)^{-1}(H - \rho I)(H + \Sigma + \rho I)^{-1}(V - \rho I)$$

(6.2.2)

and

$$k = (\rho + \tilde{\rho})(V + \Sigma + \tilde{\rho}I)^{-1}(H + \Sigma + \rho I)^{-1} b$$

(6.2.3)

The iteration matrix $T_{\rho, \tilde{\rho}}$ can be written in the form

$$T_{\rho, \tilde{\rho}} = I - (\rho + \tilde{\rho})(V + \Sigma + \tilde{\rho}I)^{-1}(H + \Sigma + \rho I)^{-1} A$$

(6.2.4)

and hence

$$k = (I - T_{\rho, \tilde{\rho}}) A^{-1} b$$

(6.2.5)

which is the consistency condition (see (4.1.5)). Since $(I - T_{\rho, \tilde{\rho}})$ is nonsingular, then the method is completely consistent.

There are a few ways of eliminating $\Sigma$ from the equations above, we will consider the simplest choice, i.e. $\Sigma = 0$.

Theorem 6.2.1

For $\rho$ and $\tilde{\rho}$ positive, if $H + \frac{1}{2} (\rho + \tilde{\rho}) I$ and $V + \frac{1}{2} (\tilde{\rho} - \rho) I$ are positive definite then $\rho T_{\rho, \tilde{\rho}} < 1$.

Proof:

Define the matrix $\tilde{T}_{\rho, \tilde{\rho}}$ similar to $T_{\rho, \tilde{\rho}}$ such that
\[ T_{\rho,\tilde{\rho}} = (V + \tilde{\rho}I) T_{\rho,\tilde{\rho}} (V + \tilde{\rho}I)^{-1} \]
\[ = (H - \tilde{\rho}I)(H + \rho I)^{-1}(V - \rho I)(V + \tilde{\rho}I)^{-1} \quad (6.2.6) \]

Let \( \mu \) be any eigenvalue of \( H \), then \( (\mu - \tilde{\rho})/(\mu + \rho) \) is an eigenvalue of \((H - \tilde{\rho}I)(H + \rho I)^{-1}\). Given \( H + \frac{1}{2} (\rho - \tilde{\rho})I \) is positive definite which implies that \( \mu + \frac{1}{2} (\rho - \tilde{\rho}) > 0 \) and hence \( \mu + \rho > 0 \).

Now
\[ 1 - \frac{\mu - \tilde{\rho}}{\mu + \rho} = \frac{\rho + \tilde{\rho}}{\mu + \rho} > 0 \]
and
\[ 1 + \frac{\mu - \tilde{\rho}}{\mu + \rho} = \frac{2\mu + \rho - \tilde{\rho}}{\mu + \rho} > 0 \]

We conclude that
\[ \left| \frac{\mu - \tilde{\rho}}{\mu + \rho} \right| < 1 \]
which implies
\[ \rho \left[ (H - \tilde{\rho}I)(H + \rho I)^{-1} \right] < 1 \quad (6.2.7) \]

Similarly, we can conclude that
\[ \rho \left[ (V - \rho I)(V + \tilde{\rho}I)^{-1} \right] < 1 \quad (6.2.8) \]

So, from equation (6.2.6), we have
\[ \rho(T_{\rho,\tilde{\rho}}) = \rho(T_{\rho,\tilde{\rho}}) \]
\[ = \rho \left[ (H - \tilde{\rho}I)(H + \rho I)^{-1} \right] . \rho \left[ (V - \rho I)(V + \tilde{\rho}I)^{-1} \right] \]
or
\[ \rho(T_{\rho,\tilde{\rho}}) < 1 \]
which concludes the proof.

For the case \( \rho = \tilde{\rho} \), we define \( T_{\rho} = T_{\rho,\rho} \). Then we have
Corollary 6.2.1

If $H$ and $V$ are positive definite, then for any $\rho > 0$ we have

$$\rho(T_\rho) < 1.$$  \hfill (6.2.9)

**Theorem 6.2.2**

If $\rho$ and $\bar{\sigma}$ are positive and if there exists a nonsingular matrix $P$ such that $P^{-1}(H + \frac{1}{2}(\rho-\bar{\sigma})I)P$ and $P^{-1}(V + \frac{1}{2}(\bar{\sigma}-\rho)I)P$ are real positive definite matrices or, equivalently, if there exists a positive definite matrix $Q$ such that $Q(H + \frac{1}{2}(\rho-\bar{\sigma})I)$ and $Q(V + \frac{1}{2}(\bar{\sigma}-\rho)I)$ are real positive definite, then $\rho(T_{\rho,\bar{\rho}}) < 1.$

**Corollary 6.2.2**

If there exists a nonsingular matrix $P$ such that $P^{-1}HP$ and $P^{-1}VP$ are real positive definite matrices or, equivalently, if there exists a positive definite matrix $Q$ such that the matrices $QH$ and $QV$ are real positive definite, then $\rho(T_{\rho}) < 1$ for all $\rho > 0.$

**Theorem 6.2.3**

A necessary and sufficient condition that $\rho(T_{\rho,\bar{\rho}}) < 1$, is that the matrix

$$(H + V)^{-1}(HV + \frac{1}{2}(\rho-\bar{\sigma})(V - H) + \rho\bar{\sigma}I)$$  \hfill (6.2.10)

has its eigenvalues possessing positive real parts (such a matrix is called $N$-stable).

**Corollary 6.2.3**

A necessary and sufficient condition that $\rho(T_{\rho}) < 1$, is
that the matrix
\[(H + V)^{-1}(HV + \rho^2 I)\]  \hspace{1cm} (6.2.11)
is N-stable.

6.2.2 Selection of Parameters.

We aim to estimate the optimal values of \(\rho\) and \(\bar{\rho}\) which minimize the spectral radius of the iteration matrix \(T_{\rho, \bar{\rho}}\) in order to maximize the rate of convergence of the method.

We assume that the matrices \(H\) and \(V\) are positive definite matrices, \(\mu\) is an eigenvalue of \(H\) and \(\eta\) is that of \(V\) such that
\[0 < a \leq \mu \leq b \quad \text{and} \quad 0 < a \leq \eta \leq b\]  \hspace{1cm} (6.2.12)
where \((a, b)\) and \((\alpha, \beta)\) are the domains of the eigenvalues of \(H\) and \(V\) respectively. Now from equation (6.2.6), we have
\[
\rho(T_{\rho, \bar{\rho}}) = \rho \left[ (V + \bar{\rho}I)T_{\rho, \bar{\rho}}(V + \bar{\rho}I)^{-1} \right] \\
= \rho \left[ (H - \bar{\rho}I)(H + \rho I)^{-1}(V - \rho I)(V + \bar{\rho}I)^{-1} \right] \\
\leq \| (H - \bar{\rho}I)(H + \rho I)^{-1} \| \| (V - \rho I)(V + \bar{\rho}I)^{-1} \|
\]
or
\[
\rho(T_{\rho, \bar{\rho}}) \leq \max_{\alpha \leq \mu \leq \beta} \left| \frac{\mu - \bar{\rho}}{\mu + \rho} \right| \quad \max_{\alpha \leq \eta \leq \beta} \left| \frac{\eta - \rho}{\eta + \bar{\rho}} \right|
\]
\[
= \max_{\alpha \leq \mu \leq \beta} \left| \frac{\mu - \bar{\rho}}{\mu + \rho} \right| \left( \frac{\eta - \rho}{\eta + \bar{\rho}} \right) \quad \text{or} \quad \text{(6.2.13)}
\]

We need to select the values of \(\rho\) and \(\bar{\rho}\) which minimize the quantity
\[
\psi(a, b; \alpha, \beta; \rho, \bar{\rho}) = \max_{\alpha \leq \mu \leq \beta} \left| \frac{\mu - \bar{\rho}}{\mu + \rho} \right| \left( \frac{\eta - \rho}{\eta + \bar{\rho}} \right) \quad \text{or} \quad \text{(6.2.14)}
\]
We will discuss the optimum parameters for the cases when \( p = \tilde{p} \) and the ranges of the eigenvalues \( \mu \) and \( \eta \) are either

i) the same or ii) different.

i) Case where \( p = \tilde{p} \), same range for \( \mu \) and \( \eta \).

We have \( p = \tilde{p} \) and \( a \leq \mu, \eta \leq b \).

Theorem 6.2.4

There exists a unique optimum value of \( p \), namely,

\[ p^* = (ab)^{\frac{1}{1}} \]

such that

\[ \psi(a,b; \tilde{a},\tilde{b}; p^*,p^*) = \frac{(b^{\frac{1}{2}} - a^{\frac{1}{2}})^2}{(b^{\frac{1}{2}} + a^{\frac{1}{2}})} \]

and

\[ \psi(a,b; \tilde{a},\tilde{b}; p^*,p^*) < \psi(a,b; \tilde{a},\tilde{b}; \rho,\rho) \]

for all \( \rho \neq p^* \).

Proof:

From (6.2.14), we can write

\[ \psi(a,b; \tilde{a},\tilde{b}; \rho,\rho) = \phi(a,b; \rho)^2 \]

where

\[ \phi(a,b; \rho) = \max_{a \leq \mu \leq b} \frac{(\mu-\rho)/(\mu+\rho)}{2} \]

Now, for a given \( \rho \) in the interval \( a \leq \rho \leq b \), we have

\[ \frac{d\phi}{d\mu} = \frac{2\rho}{(\mu+\rho)^2} > 0 \]

and the extreme values of \( (\mu-\rho)/(\mu+\rho) \) occur at \( \mu = a \) and \( \mu = b \),
and hence (6.2.20) becomes

\[ \phi(a,b; \rho) = \max \left\{ \frac{b-a}{p+a}, \frac{b-p}{b+p} \right\} \]

i.e.

\[ \phi(a,b; \rho) = \begin{cases} 
\frac{(b-p)/(b+p)}{(p-a)/(p+a)}, & \text{for } a \leq \rho \leq (ab)^\frac{1}{2} \\
\frac{(p-a)/(p+a)}{(b-p)/(b+p)}, & \text{for } (ab)^\frac{1}{2} \leq \rho \leq b
\end{cases} \]  

(6.2.21)

Further, we have

\[ \frac{d}{d\rho} \left( \frac{b-p}{b+p} \right) = \frac{-2b}{(b+p)^2} < 0 \]

(6.2.22)

\[ \frac{d}{d\rho} \left( \frac{p-a}{p+a} \right) = \frac{2a}{(p+a)^2} > 0 \]

From (6.2.22), we see that \( \phi(a,b; \rho) \) is a decreasing function of \( \rho \) for \( a \leq \rho \leq (ab)^\frac{1}{2} \) and an increasing function of \( \rho \) for \( (ab)^\frac{1}{2} \leq \rho \leq b \). So we can assume the minimum when \( \rho = \rho^* = (ab)^\frac{1}{2} \)

and hence

\[ \phi(a,b; \rho^*) = \frac{b^\frac{1}{2} - a^\frac{1}{2}}{b^\frac{1}{2} + a^\frac{1}{2}} \]

and the proof follows from (6.2.19).

\( \tilde{\imath} \text{)} \text{ Case where } \rho = \beta, \text{ different ranges for } \mu \text{ and } \eta \)

We have \( \rho = \beta, a \leq \mu \leq b \text{ and } a \leq \eta \leq \beta \).

**Theorem 6.2.5**

If \( ab \leq a\beta \), then there exists an optimum value of \( \rho \), given by
\[
\rho^* = \begin{cases} 
(ab)^{\frac{1}{4}}, & \text{if } a \geq a \text{ or } a \leq a \text{ and } a \beta \geq b \alpha \\
(ab)^{\frac{1}{4}}, & \text{if } b \geq b \text{ or } b \leq b \text{ and } a \beta \leq b \alpha
\end{cases}
\]

such that

\[
\psi(a, b; \alpha, \beta; \rho^*, \rho^*) = \frac{\rho^* - a}{\rho^* + a} \left( \frac{b - \rho^*}{b + \rho^*} \right)
\]

\[
= \begin{cases} 
\left(\frac{b^\frac{1}{4} - a^\frac{1}{4}}{b^\frac{3}{4} + a^\frac{3}{4}}\right) \left(\frac{b - (ab)^{\frac{1}{4}}}{b + (ab)^{\frac{1}{4}}}\right), & \text{if } \rho^* = (ab)^{\frac{1}{4}} \\
\left(\frac{(a\beta)^{\frac{1}{4}} - a}{(a\beta)^{\frac{3}{4}} + a^\frac{3}{4}}\right) \left(\frac{b^\frac{1}{4} - a^\frac{1}{4}}{b^\frac{3}{4} + a^\frac{3}{4}}\right), & \text{if } \rho^* = (ab)^{\frac{1}{4}}
\end{cases}
\]

(6.2.24)

and 

\[
\psi(a, b; \alpha, \beta; \rho^*, \rho^*) < \psi(a, b; \alpha, \beta; \rho, \rho)
\]

for all \( \rho \neq \rho^* \).

**Proof:**

From (6.2.14) and (6.2.20), we have

\[
\psi(a, b; \alpha, \beta; \rho, \rho) = \phi(a, b; \rho) \phi(\alpha, \beta; \rho) .
\]

(6.2.25)

Now, similarly as in Theorem 6.2.4, we can show that \( \phi(a, b; \rho) \) and \( \phi(\alpha, \beta; \rho) \) are both decreasing functions of \( \rho \) for \( a \leq \rho \leq (ab)^{\frac{1}{4}} \) and are both increasing functions of \( \rho \) for \( (a\beta)^{\frac{1}{4}} \leq \rho \leq b \). Hence a minimum of \( \psi \) is assumed in the interval \( (ab)^{\frac{1}{4}} \leq \rho \leq (a\beta)^{\frac{1}{4}} \). In this interval we have

\[
\phi(a, b; \rho) = \frac{b - a}{\rho + a} \text{ and } \phi(\alpha, \beta; \rho) = \frac{\beta - \rho}{\beta + \rho}
\]

(6.2.26)

and
(d^2/d\rho^2) \log \psi(a,b; \rho) = -4 \frac{ab}{(\rho^2-a^2)^2} \quad (6.2.27)

(d^2/d\rho^2) \log \psi(a,\beta; \rho) = -4 \frac{\beta \rho}{(\beta^2-\rho^2)^2} \quad .

So that

(d^2/d\rho^2) \log \psi(a,b; \alpha,\beta; \rho,\rho) < 0 \quad . \quad (6.2.28)

Also, we have

(d^2/d\rho^2) \log \psi = \left[ \psi'' - (\psi')^2 \right] / \psi^2 \quad . \quad (6.2.29)

We now search for the minimum of \psi, if \psi' = 0, then \psi'' < 0

by (6.2.28) and since \psi > 0, thus there is not a local minimum

for \psi in the interval \((ab)^\frac{1}{4}, (\alpha \beta)^\frac{1}{4}\). Therefore the extreme

values must occur at the end of points of the interval, i.e. at

\rho = (ab)^\frac{1}{4} or \rho = (\alpha \beta)^\frac{1}{4}, so we have

\[ \min \psi(a,b; \alpha,\beta; \rho,\rho) = \min \left\{ \frac{b^{\frac{1}{4}}-a^{\frac{1}{4}}}{b^{\frac{1}{4}}+a^{\frac{1}{4}}} \left[ \frac{\beta-(ab)^\frac{1}{4}}{\beta+(ab)^\frac{1}{4}} \right] \left[ \frac{(\alpha \beta)^\frac{1}{4}-a^{\frac{1}{4}}}{(\alpha \beta)^\frac{1}{4}+a^{\frac{1}{4}}} \right] \right\} \quad (6.2.30) \]

Now, we define x and y using the two terms in the r.h.s. of

(6.2.30) such that

\[ S_1 = \frac{1-x}{1+x} = \frac{\left[ \frac{b^{\frac{1}{4}}-a^{\frac{1}{4}}}{b^{\frac{1}{4}}+a^{\frac{1}{4}}} \right] \left[ \frac{\beta-(ab)^\frac{1}{4}}{\beta+(ab)^\frac{1}{4}} \right]}{\left[ \frac{(\alpha \beta)^\frac{1}{4}-a^{\frac{1}{4}}}{(\alpha \beta)^\frac{1}{4}+a^{\frac{1}{4}}} \right]} \]

\[ S_2 = \frac{1-y}{1+y} = \frac{\left[ \frac{(\alpha \beta)^\frac{1}{4}-a^{\frac{1}{4}}}{(\alpha \beta)^\frac{1}{4}+a^{\frac{1}{4}}} \right]}{\left[ \frac{b^{\frac{1}{4}}-a^{\frac{1}{4}}}{b^{\frac{1}{4}}+a^{\frac{1}{4}}} \right] \left[ \frac{\beta-(ab)^\frac{1}{4}}{\beta+(ab)^\frac{1}{4}} \right]} \]

which involve

\[ x = \frac{(a/b)^\frac{1}{4} + [(ab)^\frac{1}{4}]/\beta}{1 + (a/\beta)} \]

\[ y = \frac{[a/(\alpha \beta)^\frac{1}{4}] + (a/\beta)^\frac{1}{4}}{1 + (a/\beta)} \]
Clearly, $S_1 \leq S_2$ if and only if $x \geq y$, i.e.

\[ (b^\frac{1}{2} + ba^\frac{1}{2}) / b^\frac{1}{2} \geq (\beta^\frac{1}{2} + \alpha^\frac{1}{2}) / \alpha^\frac{1}{2} \]

i.e. if and only if

\[ T_1 = (b/b)^{\frac{1}{2}} + (b/\beta)^{\frac{1}{2}} \geq (a/a)^{\frac{1}{2}} + (\alpha/a)^{\frac{1}{2}} = T_2. \]

Suppose now that $a \geq \alpha$, then $b \leq \beta$; otherwise, $ab \geq \alpha \beta$. Moreover, $a/\alpha \leq \beta/b$ since $ab \leq \alpha \beta$; hence $T_1 \geq T_2$, $S_1 \geq S_2$ and $\rho^* = (ab)^{\frac{1}{2}}$.

Similarly, if $b \geq \beta$, then $a \leq \alpha$ and $\alpha/a \geq b/\beta$.

Therefore, $T_2 \geq T_1$, $S_2 \geq S_1$ and $\rho^* = (\alpha \beta)^{\frac{1}{2}}$.

Now suppose $a \leq \alpha$ and $b \leq \beta$. If $ab \geq \beta a$, then $b/b \geq \alpha/a$ and $T_1 \geq T_2$, $S_1 \geq S_2$, $\rho^* = (ab)^{\frac{1}{2}}$.

Also, if $ab \leq \beta a$, then $b/\beta \leq \alpha/a$ and $T_1 \leq T_2$, $S_1 \leq S_2$, $\rho^* = (\alpha \beta)^{\frac{1}{2}}$, which completes the proof.

For the other two cases where $\rho$ and $\bar{\rho}$ may be different and the ranges of $\rho$ and $\eta$ are either the same or different, can be found in Young, (1971).

6.3 THE COMMUTATIVE CASE.

We now study the case where the matrices $H$, $V$ and $E$ are commutative, i.e. when

\[ HV = VH, \quad HE = \Sigma H, \quad VE = \Sigma V. \]

Further Birkhoff and Varga (1959) defined the following conditions

\[ HV = VH \]

\[ \Sigma = \sigma I, \text{ where } \sigma \text{ is a nonnegative constant} \]

\[ H \text{ and } V \text{ are similar to nonnegative diagonal matrices} \]
The following theorem relates conditions (6.3.1) and (6.3.2).

**Theorem 6.3.1**

If \( H \) and \( V \) are symmetric positive definite matrices and if \( H + V \) is irreducible, then the condition (6.3.1) is equivalent to the condition (6.3.2).

**Proof:** See Young, 1971.

The importance of the conditions (6.3.2) for the study of ADI methods depends on the following theorem of Frobenius (see Thrall and Tornheim, 1957).

**Theorem 6.3.2**

There exists a nonsingular matrix \( W \) such that \( W^{-1}HW \) and \( W^{-1}VW \) are diagonal matrices if and only if \( HV = VH \) and \( H \) and \( V \) are similar to diagonal matrices.

**Proof:**

Let \( W \) exist, then \( W^{-1}HW = \overline{H} \), \( W^{-1}VW = \overline{V} \), where \( \overline{H} \) and \( \overline{V} \) are diagonal matrices. We have

\[
HV = (\overline{WHW}^{-1})(\overline{WVW}^{-1}) = \overline{WHW}^{-1}
\]

\[
= W^{-1}WH = W^{-1}VW = VH
\]

Conversely, we suppose that \( HV = VH \) and that \( H \) and \( V \) are similar to diagonal matrices, so there exists a nonsingular matrix \( P \) such that \( P^{-1}VP \) is a diagonal matrix. Let \( \lambda_1, \lambda_2, \ldots, \lambda_p \) be
distinct eigenvalues of $V$, then $P^{-1}VP$ may have the form

$$P^{-1}VP = \begin{pmatrix}
\lambda_1 I_1 & 0 & 0 & \ldots & 0 \\
0 & \lambda_2 I_2 & 0 & \ldots & 0 \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
0 & 0 & 0 & \ldots & \lambda_p I_p
\end{pmatrix}$$

where $I_i$, $i = 1, \ldots, P$ are square identity matrices.

For simplicity, we let $P = 4$, then for the matrix $H$ we have

$$\hat{H} = P^{-1}HP = \begin{pmatrix}
\hat{H}_{11} & \hat{H}_{12} & \hat{H}_{13} & \hat{H}_{14} \\
\hat{H}_{21} & \hat{H}_{22} & \hat{H}_{23} & \hat{H}_{24} \\
\hat{H}_{31} & \hat{H}_{32} & \hat{H}_{33} & \hat{H}_{34} \\
\hat{H}_{41} & \hat{H}_{42} & \hat{H}_{43} & \hat{H}_{44}
\end{pmatrix}$$

by which we can write

$$P^{-1}(HV VH)P = \begin{pmatrix}
0 & \hat{H}_{12}(\lambda_2 - \lambda_1) & \hat{H}_{13}(\lambda_3 - \lambda_1) & \hat{H}_{14}(\lambda_4 - \lambda_1) \\
\hat{H}_{21}(\lambda_1 - \lambda_2) & 0 & \hat{H}_{23}(\lambda_3 - \lambda_2) & \hat{H}_{24}(\lambda_4 - \lambda_2) \\
\hat{H}_{31}(\lambda_1 - \lambda_3) & \hat{H}_{32}(\lambda_2 - \lambda_3) & 0 & \hat{H}_{34}(\lambda_4 - \lambda_3) \\
\hat{H}_{41}(\lambda_1 - \lambda_4) & \hat{H}_{42}(\lambda_2 - \lambda_4) & \hat{H}_{43}(\lambda_3 - \lambda_4) & 0
\end{pmatrix}$$

(6.3.3)

Since the matrix (6.3.3) vanishes, we have

$$\hat{H}_{i,j} = 0 \text{ for } i \neq j$$

and hence

$$\hat{H} = \begin{pmatrix}
\hat{H}_{11} & 0 & 0 & 0 \\
0 & \hat{H}_{22} & 0 & 0 \\
0 & 0 & \hat{H}_{33} & 0 \\
0 & 0 & 0 & \hat{H}_{44}
\end{pmatrix}$$
Let $Q$ be the nonsingular matrix which reduces the matrix $H$ to Jordan canonical form such that

$$Q^{-1}HQ = \begin{bmatrix}
J_1 & 0 & 0 & 0 \\
0 & J_2 & 0 & 0 \\
0 & 0 & J_3 & 0 \\
0 & 0 & 0 & J_4
\end{bmatrix}$$

is a diagonal matrix. It follows that $W = PQ$ reduces $H$ (and similarly $V$) to diagonal form. The theorem is proved.

If the matrices $H$ and $V$ satisfies condition (6.3.2), then by theorem 6.3.2 there exist a set of $n$ linearly independent vectors $v_i$, $i = 1, 2, \ldots, n$ which are eigenvectors of both $H$ and $V$. So if $v$ is any of these vectors then we have

$$Hv = \mu v, \quad Vv = \eta v.$$  \hspace{1cm} (6.3.4)

Now, for any positive $p$ and $\bar{p}$ we have

$$T_{p, \bar{p}} v = \left(\frac{(\mu-\bar{p})(\eta-p)}{(\mu+p)(\eta+\bar{p})}\right) v$$ \hspace{1cm} (6.3.5)

or

$$\prod_{i=1}^{m} T_{\rho_i, \bar{\rho}_i} v = \prod_{i=1}^{m} \left(\frac{(\mu-\bar{\rho}_i)(\eta-\rho_i)}{(\mu+\rho_i)(\eta+\bar{\rho}_i)}\right) \cdot v.$$ \hspace{1cm} (6.3.6)

Again if $a \leq \mu \leq b$ and $\alpha \leq \eta \leq \beta$ are the ranges of $\mu$ and $\eta$, then

$$p \left(\prod_{i=1}^{m} T_{\rho_i, \bar{\rho}_i}\right) \leq \psi_m(a, b; \alpha, \beta; p, \bar{p}).$$ \hspace{1cm} (6.3.7)

where

$$\rho = (\rho_1, \rho_2, \ldots, \rho_m), \quad \rho_i \leq \rho$$

and

$$\bar{\rho}_i = (\bar{\rho}_1, \bar{\rho}_2, \ldots, \bar{\rho}_m),$$

and
If $\rho_i = \bar{\rho}_i$, $i = 1, 2, \ldots, m$, then we have the $m$-parameter problem of minimizing

$$
\psi_m = \max_{a \leq \gamma \leq b} \max_{i=1}^{m} \left| \frac{(\gamma - \bar{\rho}_i) \sum_{j=1}^{m} (\gamma - \rho_j)}{(\gamma + \rho_i) \sum_{j=1}^{m} (\gamma + \rho_j)} \right|.
$$

(6.3.9)

for some eigenvalue $\gamma$ of $H$ and $V$.

### 6.4 GOOD ACCELERATION PARAMETER.

There are explicit formulas for the acceleration parameter $\rho$ which give values as good as the optimal parameters. Peaceman and Rachford (1955) suggested one way to determine the values, known as the Peaceman-Rachford parameters,

$$
\rho_i^{(P)} = \frac{b(a/b)^{(i-1)/m}}{2m}, \quad i = 1, 2, \ldots, m
$$

(6.4.1)

Wachspress (1957) defined what has become known as the Wachspress parameters,

$$
\rho_i^{(W)} = \frac{b(a/b)^{(i-1)/(m-1)}}{a}, \quad i = 1, 2, \ldots, m
$$

(6.4.2)

Where $a$ and $b$ are the lower and upper bounds of the eigenvalues of $H$ and $V$.

**Theorem 6.4.1**

For the Peaceman-Rachford parameters defined by (6.4.1), we have,

$$
\phi_m(a, b; \rho^{(P)}) \leq \left[ \frac{1 - z}{1 + z} \right],
$$

(6.4.3)
where
\[ z = (a/b)^{\frac{1}{m}} \quad \text{and} \quad \varphi = (\rho_1, \rho_2, \ldots, \rho_m). \]

**Proof:**

Given
\[ \phi_m(a, b, \varphi) = \max_{\gamma \in \gamma} \prod_{i=1}^{m} \frac{(|\gamma - \rho_i|)}{(|\gamma + \rho_i|)}. \quad (6.4.4) \]

Each factor in (6.4.4) is less than one in magnitude, so it is sufficient to obtain the proof for one factor. We have \[ \rho_i = b \cdot z^{2i-1} \]
and we may write \[ \rho_i, \ i = 1, 2, \ldots, m \] in the following order,
\[ b > \rho_1 > \rho_2 > \ldots > \rho_m > a. \]

In the interval \( a \leq \gamma < \rho_1 \), we have
\[
\frac{|\gamma - \rho_m|}{|\gamma + \rho_m|} < \frac{\rho_m - a}{\rho_m + a} = \frac{bz^{2m-1} - a}{bz^{2m-1} + a}
\]
\[
= \frac{z^{2m-1} - z^{2m}}{z^{2m-1} + z} = \frac{1 - z}{1 + z}.
\]

In the interval \( \rho_1 \leq \gamma < b \), the same inequality holds.

Now, in the interval \( \rho_i < \gamma < \rho_{i+1} \), the function
\[
\left[ \frac{\gamma - \rho_{i+1}}{\gamma + \rho_{i+1}} \right] \left( \frac{\gamma - \rho_i}{\gamma + \rho_i} \right)
\]
has a maximum at \( \gamma = (\rho_i, \rho_{i+1}) \), so
\[
\left| \frac{\gamma - \rho_{i+1}}{\gamma + \rho_{i+1}} \right| \left| \frac{\gamma - \rho_i}{\gamma + \rho_i} \right| \leq \left| \frac{(\rho_i, \rho_{i+1})^{1/2} - \rho_{i+1}}{(\rho_i, \rho_{i+1})^{1/2} + \rho_{i+1}} \right| \left| \frac{(\rho_i, \rho_{i+1})^{1/2} - \rho_i}{(\rho_i, \rho_{i+1})^{1/2} + \rho_i} \right|
\]
which is obtained directly by using the Peaceman-Rachford parameters given by (6.4.1), which concludes the proof.

Theorem 6.4.2

For the Wachspress parameters defined by (6.4.2), we have

\[ \phi_m^2 \leq \left( \frac{1-y}{1+y} \right)^2 , \]  \hspace{1cm} (6.4.5)

where

\[ y = \left( \frac{a}{b} \right)^{1/(2m-2)} . \] \hspace{1cm} (6.4.6)

Proof:

As in the proof of theorem 6.4.1, the maximum value of

\[ \left( \frac{Y-p_{i+1}}{Y+p_{i+1}} \right) \left( \frac{Y-p_i}{Y+p_i} \right) \] in the interval \((\rho_{i+1}, \rho_i)\) is

\[ \left( \frac{\rho_i - \rho_{i+1}}{\rho_i + \rho_{i+1}} \right)^2 = \left( \frac{1-y}{1+y} \right)^2 , \]

by using the Wachspress parameters, then the inequality (6.4.5) follows.

6.5 THE PEACEMAN-RACHFORD (P-R) METHOD.

6.5.1 Introduction.

Suppose we have the following system of equations to solve:

\[ (H + V + \Sigma)u = b , \] \hspace{1cm} (6.5.1)
where \( A = H + V + \Sigma \) such that \( H \) and \( V \) are symmetric matrices and \( \Sigma \) is a nonnegative diagonal matrix.

We now define the Peaceman-Rachford method for solving the above system as

\[
(H_1 + \rho_k I)u^{(k+1)} = b - (V_1 - \rho_k I)u^{(k)} \tag{6.5.2}
\]

\[
(V_1 + \rho_k I)u^{(k+1)} = b - (H_1 - \rho_k I)u^{(k+1)} \tag{6.5.3}
\]

where \( H_1 = H + \frac{1}{2} \Sigma \), \( V_1 = V + \frac{1}{2} \Sigma \) and \( \rho_k \) are acceleration parameters. This method is derived from equations (6.1.11) and (6.1.12) by letting \( D_k = E_k = \rho_k I - \frac{1}{2} \Sigma \). The Peaceman-Rachford method can be expressed in matrix form as

\[
u^{(k+1)} = T_{\rho_k} \nu^{(k)} + r_k, \quad k \geq 0, \tag{6.5.4}
\]

where the iteration matrix \( T_{\rho_k} \) is given by

\[
T_{\rho_k} = (V_1 + \rho_k I)^{-1}(H_1 - \rho_k I)(H_1 + \rho_k I)^{-1}(V_1 - \rho_k I), \tag{6.5.5}
\]

and the constant vector \( r_k \) is given by

\[
r_k = (V_1 + \rho_k I)^{-1}\left\{I - (H_1 - \rho_k I)(H_1 + \rho_k I)^{-1}\right\}b. \tag{6.5.6}
\]

Now, analogous to equations (6.3.6) and (6.3.7) we define

\[
\lambda(\mu', \eta') = \prod_{i=1}^{m} \frac{(\mu' - \rho_i)(\eta' - \rho_i)}{(\mu' + \rho_i)(\eta' + \rho_i)} \tag{6.5.7}
\]

as an eigenvalue of \( \prod_{i=1}^{m} T_{\rho_i} \). Also we have

\[
\lambda(\mu', \eta') = \prod_{i=1}^{m} \rho_i. \]
\[
\rho \left( \prod_{i=1}^{m} \rho_i \right) \leq \max_{a \leq \mu_i \leq b} \prod_{i=1}^{m} \left( \frac{(\mu_i - p_i)(n_i - p_i)}{(\mu_i + p_i)(n_i + p_i)} \right), \quad (6.5.8)
\]

where \( \mu_i \) and \( n_i \) are the eigenvalues of \( H_i \) and \( V_i \) respectively.

To obtain a minimum \( \rho \left( \prod_{i=1}^{m} \rho_i \right) \) we need to minimize the function

\[
\psi_m(a, b; \alpha, \beta; \rho) = \max_{a \leq \mu_i \leq b} \prod_{i=1}^{m} \left( \frac{(\mu_i - p_i)(n_i - p_i)}{(\mu_i + p_i)(n_i + p_i)} \right), \quad (6.5.9)
\]

where \( \rho = (\rho_1, \rho_2, \ldots, \rho_m) \).

It is convenient to use the inequality

\[
\psi_m(a, b; \alpha, \beta; \rho) \leq \left[ \phi_m(\bar{a}, \bar{b}, \rho) \right]^2 \quad (6.5.10)
\]

where \( \bar{a} = \min(a, b) \) and \( \bar{b} = \max(b, \beta) \) and where

\[
\phi_m(\bar{a}, \bar{b}, \rho) = \max_{a \leq y \leq b} \prod_{i=1}^{m} \left| \frac{\gamma - p_i}{\gamma + p_i} \right|.
\]

The problem of minimizing \( \psi_m \) and \( \phi_m \) for \( m = 1 \) (one parameter) is equivalent to the problem of determining the minimax of the continuous function

\[
F(a, b; \alpha, \beta) = \min_{\rho} \phi(a, b, \rho) \quad (6.5.11)
\]

where

\[
\phi(a, b, \rho) = \max_{a \leq \mu_i \leq b} \left| \frac{\mu_i - \rho}{\mu_i + \rho} \right| \quad (6.5.12)
\]

We can notice that \( \phi < 1 \) for any \( \rho > 0 \), and \( \phi > 1 \) for any \( \rho < 0 \), also \( \phi + 1 = \rho + \infty \). Hence \( F(a, b; \alpha, \beta) \) has a minimum for at least one finite positive value of \( \rho \) (i.e. the optimum \( \rho^* \)).
For a fixed $\rho > 0$, the function $\frac{\mu' - \rho}{\mu' + \rho}$ is increasing for $\mu' > \rho$ and decreasing for $\mu' < \rho$ and has a minimum at $\mu' = \rho$. Therefore it has maximum value at $\mu' = a$ or $\mu' = b$. We can obtain

$$\phi(a,b, \rho) = \begin{cases} \frac{(b-\rho)}{(b+\rho)} & \text{if } 0 \leq \rho \leq \sqrt{ab} \\ \frac{(\rho-a)}{(\rho+a)} & \text{if } \rho \geq \sqrt{ab} \end{cases}$$

(6.5.13)

Now, for the case when $m > 1$, we define the following functions:

$$F_m(a,b) = \min_{\rho} \phi_m(a,b; \rho),$$

where $\rho = (\rho_1, \rho_2, \ldots, \rho_m)$ and

$$\phi_m(a,b, \rho) = \max_{a \leq \mu' \leq b} \left| \prod_{i=1}^{m} \frac{\mu' - \rho_i}{\mu' + \rho_i} \right|$$

(6.5.15)

which are a generalization of the functions in (6.5.11) and (6.5.12).

6.5.2 Choice of Parameters and Convergence.

Let $\mu'$ and $\eta'$ be the eigenvalues of the matrices $H_1$ and $V_1$ respectively such that $a \leq \mu'$, $\eta' \leq b$ and let

$$c = \frac{a}{b}.$$  

(6.5.16)

Recalling (6.5.8), (6.5.9) and (6.5.10), then we have

$$\rho \left( \prod_{i=1}^{m} T_{\rho_i} \right) \leq \phi_m[(a,b, \rho)]^{-2},$$

(6.5.17)

where

$$\phi_m(a,b, \rho) = \max_{a \leq \gamma \leq b} \left| \prod_{i=1}^{m} \frac{\gamma - \rho_i}{\gamma + \rho_i} \right|.$$  

(6.5.18)
We define the average rate of convergence \( \bar{R}_m \) for the P-R method for a fixed \( m \) as

\[
\bar{R}_m = -\frac{1}{m} \log \left( \rho \left( \prod_{i=1}^{m} T_{\rho_i} \right) \right). \tag{6.5.19}
\]

By the inequality (6.5.17), we have

\[
\bar{R}_m \geq \bar{R}_m = -\frac{2}{m} \log \phi_m(a, b, \varphi). \tag{6.5.20}
\]

Now, we consider the case of the P-R parameters.

**Theorem 6.5.1**

For a fixed \( m \) if the P-R parameters (given by (6.4.1)) are used, then

\[
\rho \left( \prod_{i=1}^{m} T_{\rho_i} \right) \leq \phi_m \left( a, b, \varphi \right)^2 \leq \delta^2, \tag{6.5.21}
\]

where

\[
\delta = \frac{1-z}{1+z} \tag{6.5.22}
\]

and

\[
z = c^m, \quad c = a/b. \tag{6.5.23}
\]

Moreover, as \( c \to 0 \)

\[
\bar{R}_m \geq \bar{R}_m = \frac{4}{m} z + O(z^2). \tag{6.5.24}
\]

**Proof:**

The inequality (6.5.21) was given and proved in theorem 6.4.1. For the second part of the theorem, by (6.5.20), (6.5.21) and (6.5.22) we have

\[
\bar{R}_m \geq -\frac{2}{m} \log \delta = \frac{4}{m} z + O(z^2). \tag{6.5.25}
\]
Also, by (6.5.18), (6.4.1) and (6.5.20) we have

\[
\overline{R}_m \leq - \frac{2}{m} \log \prod_{i=1}^m \frac{b-p_i}{b+p_i} \leq - \frac{2}{m} \log \prod_{i=1}^m \frac{1-z_i}{1+z_i} = \frac{4}{m} z + O(z^2) .
\]  

(6.5.26)

From (6.5.25) and (6.5.26), we obtain

\[
\overline{R}_m = \frac{4}{m} z + O(z^2) .
\]

We can estimate the average rate of convergence from (6.5.19), (6.5.20) and (6.5.21) as

\[
\overline{R}_m = - \frac{2}{m} \log \delta .
\]  

(6.5.27)

We want to determine the optimal \( m \) for a given \( c \), so by (6.5.22) and (6.5.23) for \( m \), we obtain

\[
m = \frac{1}{2} \log \frac{c}{\log((1-\delta)/(1+\delta))} .
\]  

(6.5.28)

Then equation (6.5.27) becomes

\[
\overline{R}_m = - \frac{4}{m} \log \frac{\delta \log((1-\delta)/(1+\delta))}{\log c} .
\]  

(6.5.29)

By differentiating (6.5.29) and equating to zero, we get

\[
\frac{1-\delta^2}{2} \log \frac{1-\delta}{1+\delta} = \delta \log \delta .
\]  

(6.5.30)

Lemma 6.5.1

The function \( \overline{R}_m \) given by (6.5.29) is maximized when
\[ \delta = \sqrt{2} - 1 \approx 0.414 \quad (6.5.31) \]

and the corresponding value of \( R_m^{(P)} \) is

\[ R_m^{(P)} = \frac{4(\log \delta)^2}{-\log c} \approx \frac{3.11}{-\log c} \quad (6.5.32) \]

In practice we follow the procedure:

1. Estimate \( a \) and \( b \) and compute \( c = \frac{a}{b} \).
2. Determine the smallest integer \( m \) such that

\[ (\delta)^{2m} \leq c \quad (6.5.33) \]

where \( \delta = 0.414 \).
3. Determine the iteration parameters by

\[ \rho_i^{(P)} = b\left(\frac{a}{b}\right)^{(2i-1)/2m} \quad (6.5.34) \]

4. The estimated average rate of convergence is given by

\[ \bar{\rho}_m^{(P)} = -\frac{2}{m} \log \delta \]

where

\[ \delta = \frac{1-c^{1/2m}}{1+c^{1/2m}} \]

For the above procedure we have the following theorem.

**Theorem 6.5.2**

If for given \( a \) and \( b \) the number of iteration parameters \( m \) is chosen as the smallest integer satisfying (6.5.33), and if the iteration parameters are chosen by (6.5.34), then for any \( \nu > 0 \) and for sufficiently small \( c \)
Moreover,

\[ \lim_{c \to 0} \left\{ \frac{R_m}{\log c} \right\} \geq 4 \left( \log \delta \right)^2 \approx 3.11 \]

and

\[ \lim_{c \to 0} \left\{ \frac{R_m}{\log c} \right\} \leq 4 \left| \log \delta \right| \left( \left| \log \delta \right| + \delta \right) = 4.57. \]

**Proof:** See Birkhoff, et al., 1962.

We now consider the case of the Wachspress parameters given by the formula

\[ \rho_i^{(W)} = \frac{b(a/b)^{(i-1/m-1)}}{m \geq 2, i = 1, 2, \ldots, m} \]  

**Theorem 6.5.3**

For a given \( m \), if the Wachspress iteration parameters are used, then,

\[ \rho \left( \prod_{i=1}^{m} T_{\rho_i} \right) \leq \phi_m (a, b; \rho)^2 \leq c^2, \]

where

\[ c = \left( \frac{1-y}{1+y} \right)^2, \]

and

\[ y = c^{\frac{1}{m-1}} \quad , \quad c = \frac{a}{b}. \]

Moreover, as \( c \to 0 \),

\[ R_m \geq \frac{8}{m} \frac{y + 0(y^2)}{m} \]

\[ (6.5.42) \]
Proof:

The inequality (6.5.39) was proved in the proof of theorem 6.4.2. To prove (6.5.42), we have by (6.5.20), (6.5.39) and (6.5.40)

\[ \overline{R}_m \geq - \frac{2}{m} \log \varepsilon = \frac{4}{m} y + 0(y^2) \quad . \quad \quad (6.5.43) \]

Also we have by (6.5.18), (6.4.2) and (6.5.20)

\[ \overline{R}_m \leq - \frac{2}{m} \log \prod_{i=1}^{m} \left| \frac{b_{y-p_i}}{b_{y+p_i}} \right| , \quad (6.5.44) \]

and hence by (6.4.2),

\[ \overline{R}_m \leq - \frac{4}{m} \log \left( \frac{1-y}{1+y} \right) - \frac{2}{m} \log \prod_{i=3}^{m} \left( \frac{1-y^{2i-3}}{1+y^{2i-3}} \right) \]

\[ = \frac{4}{m} y + 0(y^2) \quad . \quad (6.5.45) \]

Now, by comparing (6.5.43) and (6.5.45), we obtain the inequality (6.5.42).

We now seek to find the optimal \( m \) which maximizes the average rate of convergence as given by

\[ \overline{R}_m^{(W)} = - \frac{2}{m} \log \varepsilon \quad . \quad (6.5.46) \]

where \( \varepsilon \) is given by (6.5.40). By the inequalities (6.5.19) and (6.5.39), we note that

\[ R_m > \overline{R}_m > \overline{R}_m^{(W)} \quad . \]

If we replace \( m \) by \( m-1 \) in (6.5.46), then we will have by (6.5.40) and (6.5.41),
By lemma 6.5.1, the optimum value of \( \sqrt{\varepsilon} \) is \( \sqrt{\varepsilon}_1 = 0.414 \) and \( \varepsilon \)

would be

\[ \overline{\varepsilon} = \frac{\varepsilon^2}{(\sqrt{2} - 1)^2} \approx 0.172 \quad (6.5.48) \]

The value 0.172 is really inaccurate because of the replacement of \( m \) by \((m-1)\), also by this value we would obtain a non-integer value of \( m \) by (6.5.40) and (6.5.41). In practice, we would follow this procedure

1. Estimate \( a, b \) and compute \( c = a/b \).
2. Determine the smallest integer \( m \) such that

\[ \frac{\varepsilon^2 (m-1)}{\delta} \leq c \quad (6.5.49) \]

where \( \delta = \sqrt{2} - 1 = 0.414 \).
3. Determine the iteration parameters by (6.5.38).
4. The average rate of convergence is estimated by the formula

\[ \frac{R_m(W)}{R_m} = - \frac{2}{m} \log \varepsilon , \]

where

\[ \varepsilon = \frac{[1-c \frac{1}{2}(m-1)]^2}{[1+c \frac{1}{2}(m-1)]} . \]

We have the following theorem for the above procedure.

**Theorem 6.5.4**

If for given \( a,b \) the number of iteration parameters \( m \) is chosen as the smallest integer satisfying (6.5.49), and if the iteration parameters are estimated by (6.5.38), then for any
\[ \nu > 0 \text{ and for sufficiently small } c, \]
\[ R_m - R_m \ge 16(\log \delta)^2 - \frac{\nu}{-\log c} \approx \frac{6.22 - \nu}{-\log c} \quad (6.5.50) \]

where \( \delta = \sqrt{\tau} - 1 \approx 0.414 \). Moreover

\[
\lim_{c \to 0} \left\{ R_m (-\log c) \right\} \approx 16(\log \delta)^2 \approx 6.22 \quad (6.5.51)
\]

and

\[
\lim_{c \to 0} \left\{ R_m (-\log c) \right\} \approx 8|\log \delta|\left\{ |\log \delta| + \frac{1}{2} \delta \right\} \approx 7.66. \quad (6.5.52)
\]

**Proof:** See Birkhoff, et al., 1962.

Theorems 6.5.2 and 6.5.4 show that if the value of \( m \) is
determined by (6.5.49) and (6.5.33), then the Wachspress parameters
are superior to the Peaceman-Rachford parameters by a factor of
approximately 2.

### 6.6 THE DOUGLAS-RACHFORD (D-R) METHOD.

#### 6.6.1 Introduction.

The Douglas-Rachford methods arise from a different variant
which originally is derived from the Peaceman-Rachford and
Richardson's methods. One variant was given in equations (6.1.15)
and (6.1.16). Another variant is given by

\[
(H + I + \rho_k I)u^{(k+\frac{1}{2})} = b - (V - \rho_k I)u^{(k)} \quad (6.6.1)
\]

\[
(V + \Sigma + \rho_k I)u^{(k+1)} = (V + \frac{1}{2} \Sigma)u^{(k)} + \left( \frac{1}{2} \Sigma + \rho_k I \right)u^{(k+\frac{1}{2})}. \quad (6.6.2)
\]

By eliminating the intermediate term \( u^{(k+\frac{1}{2})} \) we obtain the iteration
matrix of the method for $\rho_k = \rho$ (i.e. $m = 1$) which has the form

$$U_\rho = (V + \Sigma + \rho I)^{-1}\left\{\left(\frac{1}{2} \Sigma + \rho I\right)(H + \Sigma + \rho I)^{-1}(\rho I - V) + V + \frac{1}{2} \Sigma \right\}. \quad (6.6.3)$$

For the variant of the method given by equations (6.1.15) and (6.1.16), it has the following iteration matrix

$$W_\rho = (V_1 + \rho I)^{-1}(H_1 + \rho I)^{-1}(H_1 V_1 + \rho^2 I) \quad (6.6.4)$$

$$= \left[H_1 V_1 + \rho (V_1 + H_1) + \rho^2 I\right]^{-1}(H_1 V_1 + \rho^2 I)$$

where, $H_1 = H + \frac{1}{2} \Sigma$ and $V_1 = V + \frac{1}{2} \Sigma$.

If $\Sigma$ is a constant matrix of the form $\sigma I$ where $\sigma$ is some constant.

Then if $\mu$ and $\eta$ are the eigenvalues of $H$ and $V$ respectively, we can define the eigenvalues of $U_\rho$ and $W_\rho$, $\lambda_U$ and $\lambda_W$ respectively;

$$\lambda_U = \frac{(\mu + \sigma)(\eta/2 + \sigma) + \rho \sigma + \rho^2 + \mu \eta/2}{(\mu + \sigma + \rho)(\eta + \sigma + \rho)} \quad (6.6.5)$$

and

$$\lambda_W = \frac{(\mu + \sigma')(\eta + \sigma') + \rho^2}{(\mu + \sigma' + \rho)(\eta + \sigma' + \rho)} \quad (6.6.6)$$

where $\sigma' = \sigma/2$.

If $\sigma = 0$, then both variants of the Douglas-Rachford method are identical. For $\sigma > 0$ and for all positive $\mu$ and $\eta$, we notice that $\lambda_W$ is smaller than $\lambda_U$ and hence $\rho(W_\rho) < \rho(U_\rho)$, furthermore for any $\rho_1, \rho_2, \ldots, \rho_m$, we have...
\[
\rho\left(\prod_{i=1}^{m} W_i^{\rho_i}\right) \leq \rho\left(\prod_{i=1}^{m} U_i^{\rho_i}\right), \quad (6.6.7)
\]

i.e. the second variant ((6.1.15) and (6.1.16)) converges faster than the first variant (6.6.1) and (6.6.2). Henceforth we shall consider only the second variant. Let \(\mu'\) and \(\eta'\) be the eigenvalues of \(H_1 = H + \sigma^1 I\) and \(V_1 = V + \sigma^1 I\), respectively, then from (6.6.6) we define

\[
\lambda(\mu', \eta') = \prod_{i=1}^{m} \frac{\mu'\eta' + \rho_i^2}{(\mu' + \rho_i)(\eta' + \rho_i)}, \quad (6.6.8)
\]

as an eigenvalue of \(\prod_{i=1}^{m} W_i^{\rho_i}\). Also we define the functions

\[
\psi_m^D(a; \alpha; \beta; \rho) = \max_{a \leq \mu', \eta' \leq b} \prod_{i=1}^{m} \frac{\mu'\eta' + \rho_i^2}{(\mu' + \rho_i)(\eta' + \rho_i)}, \quad (6.6.9)
\]

and

\[
\phi_m^D(a, b; \rho) = \max_{a \leq \mu', \eta' \leq b} \prod_{i=1}^{m} \frac{\mu'\eta' + \rho_i^2}{(\mu' + \rho_i)(\eta' + \rho_i)} \quad (6.6.10)
\]

\[
= \max_{a \leq \mu', \eta' \leq b} \prod_{i=1}^{m} \left[1 + \frac{1}{2} \left(\frac{\mu' - \rho_i}{\mu' + \rho_i}\right) \left(\frac{\eta' - \rho_i}{\eta' + \rho_i}\right)\right].
\]

Evidently, we have

\[
\rho\left(\prod_{i=1}^{m} W_i^{\rho_i}\right) \leq \psi_m^D(a; \alpha; \beta; \rho) \leq \phi_m^D(\bar{a}, \bar{b}; \rho) \leq \Phi_m^D(\bar{a}, \bar{b}; \rho) \quad (6.6.11)
\]

where again, \(\bar{a} = \min(a, \alpha)\) and \(\bar{b} = \max(b, \beta)\). The average rate of convergence can be expressed as

\[
R_m = -\frac{1}{m} \log\left(\rho\left(\prod_{i=1}^{m} W_i^{\rho_i}\right)\right) \quad (6.6.12)
\]
Evidently by (6.6.11) we have

\[ R_m \geq \bar{R}_m = -\frac{1}{m} \log \Phi^D_m(\bar{a}, \bar{b}, \rho) . \]  

(6.6.13)

Now, we define the function

\[ F^D(a, b; \alpha, \beta) = \min_{\rho > 0} \max_{a \leq \alpha \leq \beta} \frac{\mu'_\alpha + \rho}{(\mu'_\alpha + \rho)(\eta'_\alpha + \rho)} . \]  

(6.6.14)

The minimum here is assumed for some finite positive optimum value \( \rho = \rho^* \), since \( 0 < (\mu'_\alpha + \rho^2)/[(\eta'_\alpha + \rho)(\eta'_\alpha + \rho)] < 1 \) for \( \rho > 0 \) and positive \( \mu' \) and \( \eta' \).

The Douglas-Rachford method can be generalized to solve three-dimensional problems (Douglas, 1962). For example, suppose we want to solve the Dirichlet problem

\[ \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} = 0 \]

defined on the region \( R \) and \( u(x, y, z) \) is specified on the boundary \( C \). After applying the standard seven-point molecule, we obtain the linear system

\[ (X + Y + Z) \bar{u} = \bar{b} \]

where \( X, Y \) and \( Z \) are symmetric positive definite matrices, and which after appropriate re-ordering (in one direction, in turn) are block diagonal with each block being tridiagonal. The Douglas-Rachford for the three-dimensional problem can be written as

\[ (X + \rho_{k+1} I) \bar{u}^{(k+1/3)} = 2\bar{b} + (\rho_{k+1} I - X - 2Y - 2Z) \bar{u}^{(k)} \]
\[(Y + \rho_{k+1} I) u^{(k+2/3)} = Y u^{(k)} + \rho_{k+1} u^{(k+1/3)}\]

\[(Z + \rho_{k+1} I) u^{(k+1)} = Z u^{(k)} + \rho_{k+1} u^{(k+2/3)}\]

Each of the above three equations is carried out in a sweep through mesh lines parallel to the three coordinate axes in turn; each stage of them consists of solving tridiagonal systems.

Again, if \(X, Y\) and \(Z\) commute, the method is convergent for any acceleration parameter \(\rho > 0\). We notice that the first equation (compared with equation (6.6.1)) is modified to achieve a wider effect of the error propagation through the \(X, Y, Z\) dimensions.

### 6.6.2 Selection of Parameters and Convergence

Let \(\mu\) and \(\eta\) are the eigenvalues of the matrices \(H\) and \(V\) respectively, and lie in the region \(a \leq \mu, \eta \leq b\).

**Theorem 6.6.1**

For a fixed \(m\), if the P-R parameters are used then

\[\rho \left( \prod_{i=1}^{m} W_{i} \right) \leq \phi^{(D)}_{m}(a,b;\rho) \leq \delta_{D}, \quad (6.6.15)\]

where \(\delta_{D} = \frac{1}{2} (1+\delta^2)\), \(\quad (6.6.16)\)

and where \(\delta\) is given by (6.5.22). Moreover, as \(c \to 0\) we have

\[\overline{R}_{m} = \frac{2}{m} z + O(z^2) \quad (6.6.17)\]

where \(z\) is given by (6.5.23).
Proof:

The inequality (6.6.15) can be proved in a similar manner to that used in theorem 6.5.1. For the second part of the theorem, we have by (6.6.10), (6.6.12), (6.6.15) and (6.6.16)

\[ \overline{R}_m \geq - \frac{1}{m} \log \delta_D = - \frac{1}{m} \log \frac{1+z^2}{(1+z)^2} = \frac{2}{m} z + O(z^2) \] \hspace{1cm} (6.6.18)

Also, by (6.6.10), (6.6.11) and (6.6.13), we have

\[ \overline{R}_m \leq - \frac{1}{m} \log \Pi_{i=1}^m \left[ \frac{1}{2} + \frac{1}{2} \left( \frac{b-p_i}{b+p_i} \right)^2 \right] \],

and by the P-R parameters

\[ \overline{R}_m \leq - \frac{1}{m} \log \Pi_{i=1}^m \left[ \frac{1}{2} + \frac{1}{2} \left( \frac{2i-1}{1+z^{2i-1}} \right)^2 \right] = \frac{2}{m} z + O(z^2) \] \hspace{1cm} (6.6.19)

and by comparing (6.6.18) and (6.6.19), the proof is obtained.

The average rate of convergence is estimated by

\[ \overline{R}_m(D) = - \frac{1}{m} \log \delta_D \] \hspace{1cm} (6.6.20)

Now, by obtaining \( m \) from (6.6.16) (in a similar way to that of the P-R method) we get

\[ \overline{R}_m(D) = - \frac{2 \log \left[ \frac{1-\delta}{1+\delta} \right] \log \left( \frac{1}{2} + \frac{1}{2} \delta^2 \right)}{\log c} \]

Differentiate w.r. to \( \delta \) and equate to zero, we obtain

\[ \delta(1-\delta^2) \log \frac{1-\delta}{1+\delta} = (1+\delta^2) \log \left( \frac{1}{2} + \frac{1}{2} \delta^2 \right) \] \hspace{1cm} (6.6.21)

Let \( \delta \) be the solution of (6.6.21) which has the value
The actual value of $\delta_D$ is: $\delta_D \approx 0.68$ (Birkhoff, et al., 1962).

In practice we use the following procedure

1. Estimate $a$ and $b$ and compute $c = a/b$.
2. Determine the smallest integer $m$ such that
   \[
   \left( \frac{1-\overline{\delta}}{1+\overline{\delta}} \right)^{2m} \leq c \tag{6.6.23}
   \]
   where $\overline{\delta}$ satisfies (6.6.21) and (6.6.22) or by using (6.6.22), so that
   \[
   (0.25)^{2m} \leq c \tag{6.6.24}
   \]
3. Determine the iteration parameters by the P-R parameters.
4. The average rate of convergence is estimated by
   \[
   R_m^{(D)} = -\frac{1}{m} \log \delta_D \tag{6.6.25}
   \]
   where
   \[
   \delta_D = \frac{1}{2} \left( 1 + \overline{\delta}^2 \right) = \frac{(1+c^{1/m})}{(1+c^{\frac{1}{2m}})} \tag{6.6.26}
   \]

**Theorem 6.6.2**

If for given $a$ and $b$ the number of iteration parameters $m$ is chosen as the smallest integer such that (6.6.23) is satisfied, and if the iteration parameters are chosen by the P-R parameters, then for any $\nu > 0$ and for sufficiently small $c$,

\[
R_m \gg R_m = \frac{2 \log \left( \frac{1-\overline{\delta}}{1+\overline{\delta}} \right) \log \left( \frac{1}{2} + \frac{1}{2} \overline{\delta}^2 \right) - \nu}{-\log c} \approx \frac{1.07 - \nu}{-\log c} \tag{6.6.27}
\]
where $\delta$ satisfies (6.6.21) and $\delta \approx 0.68$. Moreover

$$\lim_{c \to 0} \left\{ \frac{R}{m} (-\log c) \right\} \geq 2 \log \left( \frac{1-\delta}{1+\delta} \right) \log \left( \frac{1}{2} + \frac{1}{2} \delta^2 \right) \approx 1.07 \quad (6.6.28)$$

and

$$\lim_{c \to 0} \left\{ \frac{R}{m} (-\log c) \right\} \leq 2 \left\{ \log \left( \frac{1-\delta}{1+\delta} \right) \right\} \left\{ \left\{ \log \left( \frac{1}{2} + \frac{1}{2} \delta^2 \right) - \frac{(1-\delta^2)}{2 \delta(1+\delta)} \right\} \approx 1.16 \right. \quad (6.6.29)$$

**Proof:**

By (6.5.22) and (6.5.23), we let

$$\xi = \frac{1-\delta}{1+\delta} \quad (6.6.30)$$

If $m$ satisfies (6.6.23), then

$$\frac{-2m}{\xi^2} < c \quad , \quad \frac{\delta}{\xi} < z \quad , \quad \delta < \delta \quad \text{and} \quad \delta \leq \left( \frac{1}{2} + \frac{\delta^2}{2} \right) \quad (6.6.31)$$

From the inequalities (6.6.13) and (6.6.15), we obtain

$$\frac{R}{m} \leq - \frac{1}{m} \log \delta \leq - \frac{1}{m} \log \left( \frac{1}{2} + \frac{\delta^2}{2} \right) = - \frac{1}{(m-1)} \frac{m-1}{m} \log \left( \frac{1}{2} + \frac{\delta^2}{2} \right) \quad (6.6.32)$$

Since $\frac{\delta^2}{m-1} > c$, we have

$$\log c \leq 2(m-1) \log \xi \quad , \quad (6.6.33)$$

and from (6.6.32) and (6.6.33), we can obtain

$$\frac{R}{m} (-\log c) \geq 2 \left\{ \log \frac{1}{2} + \frac{\delta^2}{2} \right\} \left( \frac{m-1}{m} \right) \quad (6.6.34)$$
Now, as \( c \to 0, \ m \to \infty \), so the inequality \( (6.6.28) \) follows from \( (6.6.34) \) since \( \lim_{m \to \infty} \frac{m-1}{m} = 1 \).

Now, from \( (6.6.34) \) and by \( (6.6.13) \), the inequality \( (6.6.27) \) holds.

From the inequality \( (6.6.19) \), we have

\[
\overline{R}_m \leq - \frac{1}{m} \log m \sum_{i=1}^{m} \frac{1+z^{2(2i-1)}}{(1+z)^2} \]

\[
= - \frac{1}{m} \log \frac{1+z^2}{(1+z)^2} + \frac{1}{m} \sum_{i=2}^{m} \log \left( 1 + \frac{2z^{2i-1}}{1+z^{2(2i-1)}} \right) \]

\[
\leq - \frac{1}{m} \log \frac{1+z^2}{(1+z)^2} + \frac{2}{m} \sum_{i=2}^{m} z^{2i-1} \]

\[
\leq - \frac{1}{m} \log \frac{1+z^2}{(1+z)^2} + \frac{2}{m} \frac{z^3}{1+z^2} . \quad (6.6.35) \]

By \( (6.6.23) \), we have

\[
- \log c \leq - 2m \log (\overline{z}) . \quad (6.6.36) \]

Again, by multiplying \( (6.6.35) \) and \( (6.6.36) \), we obtain

\[
\overline{R}_m (\log c) \leq 2 \left| \log \overline{z} \right| \left( \left| \log \frac{1+z^2}{(1+z)^2} \right| - 2 \frac{z^3}{1-z^2} \right) . \quad (6.6.37) \]

Because of \( (6.6.15) \), \( (6.6.23) \) and \( (6.6.30) \), it follows that

\[
\lim_{c \to 0} z = \frac{1-\delta}{1+\delta} = \overline{z} \approx 0.25 . \quad (6.6.38) \]

Now, by taking the limits as \( c \to 0 \), on both sides of \( (6.6.37) \), the inequality \( (6.6.29) \) follows, and hence the proof is complete.
6.7 MILDLY NONLINEAR PROBLEMS AND APPLICATIONS.

In this section we apply the Peaceman-Rachford and Douglas-Rachford methods to the four mildly nonlinear test problems given earlier with the use of cycles of acceleration parameters of 1, 2 and 3 groups in turn.

6.7.1 The Model Problem.

Consider the mildly nonlinear elliptic equation

\[ \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(u), \quad (6.7.1) \]

defined in a square unit region with specified boundary conditions. Using the standard procedure of approximating the left hand side of equation (6.7.1), i.e. by dividing the region into uniform meshes of size \( \Delta x = \Delta y = h \) with interior points \( u_{i,j} = (ih, jh) \) numbered in horizontal order (in rows), so equation (6.7.1) is approximated by the five-point molecule such that

\[ (-u_{i+1,j} + 2u_{i,j} - u_{i-1,j}) + (-u_{i,j+1} + 2u_{i,j} - 2u_{i,j-1}) = -hf_{ij} \]

(6.7.2)

for \( i,j = 1,2,\ldots,n \), where

\[ f_{ij} = f(u_{i,j}) \]

Equation (6.7.2) can be written in matrix form as

\[ A\mathbf{u} = \mathbf{b} \]

(6.7.3)

where \( \mathbf{b} \) is obtained from the given data and \( A \) has the form
Here $I_n$ is an $n \times n$ identity matrix and $D = 4I_n - (L_n + L_n^T)$ is a tridiagonal matrix with

$$L_n = \begin{bmatrix}
0 \\
1 & 0 \\
& 1 & 0 \\
& & 1 & 0
\end{bmatrix}.$$

Now consider the splitting $A = H + V$ where $H$ and $V$ have the forms

$$H = 2I - (L + L_n^T) \quad \text{and} \quad V = 2I - (B + B_n^T),$$

where

$$L = \begin{bmatrix} L_n & \ldots & L_n \\
& L_n & \ldots & L_n \\
& & \ldots & L_n
\end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} 0 & \ldots & 0 \\
I_n & \ldots & 0 \\
& I_n & \ldots & 0
\end{bmatrix}.$$

Clearly, $H$ is a tridiagonal matrix.

For the three dimensional case, analogous to equation (6.7.2)

$$(- u_{i+1,j,k} + 2u_{i,j,k} - u_{i-1,j,k}) + (- u_{i,j+1,k} + 2u_{i,j,k} - u_{i,j-1,k}) +$$

$$+ (- u_{i,j,k+1} + 2u_{i,j,k} - u_{i,j,k-1}) = - h^2 f_{i,j,k},$$

(6.7.5)
where \( f_{i,j,k} = f(u_{i,j,k}) \).

The matrix \( A \), here is split as \( A = X + Y + Z \) where \( X, Y \) and \( Z \) are sparse matrices corresponding to each bracketed terms in equation (6.7.5) respectively.

### 6.7.2 Method of Solution.

Each iteration of the Peaceman-Rachford or the Douglas-Rachford consists of two sweeps: the horizontal sweep which takes the mesh points along the rows and the vertical sweep which takes the mesh points along the columns of the grid. Each sweep is equivalent to one of the two equations representing the methods.

The Peaceman-Rachford method is given by

\[
(H + \rho_{k+1} I) u^{(k+1)} = b - (V - \rho_{k+1} I) u^{(k)} \tag{6.7.5}
\]

\[
(V + \rho_{k+1} I) u^{(k+1)} = b - (H - \rho_{k+1} I) u^{(k+1)} \tag{6.7.6}
\]

and the Douglas-Rachford method is given by

\[
(H + \rho_{k+1} I) u^{(k+1)} = b - (V - \rho_{k+1} I) u^{(k)} \tag{6.7.7}
\]

\[
(V + \rho_{k+1} I) u^{(k+1)} = V u^{(k)} + \rho_{k+1} u^{(k+1)} \tag{6.7.8}
\]

We notice that both methods have similar first sweeps, but the second sweep is different as the Douglas-Rachford method replaces the term \( b - H u^{(k+1)} \) (in Peaceman-Rachford) by \( V u^{(k)} \), clearly this does not look a better approximation.

In each iteration, \( b \) in the right hand side of equations (6.7.5) - (6.7.7) which includes \( f(u) \) of the given problem will be updated by evaluating \( f(u) \) at either \( u^{(k)} \) or \( u^{(k+1)} \) depending
on which part (sweep) of the iteration. For the acceleration parameters, it is efficient to use them in a cyclic manner, i.e.

\[ 1', 2', ..., m', 1', 2', ..., m', 1', ... \]

for a suitable \( m \).

Each sweep of the two methods will produce a tridiagonal linear system which we solve using the well known Gaussian Elimination.

### 6.7.3 Applications and Results

To accelerate the convergence of the ADI method, Young and Ehrich (1960) gave the following formula to estimate the acceleration parameters

\[ \rho_k = \frac{1}{4} \left\{ \sin \frac{n \pi h}{2} \right\}^{1-2k} \frac{1}{m} \]  

where \( h \) is the mesh size.

The values obtained by the above formula remain theoretical estimates used only as a guide to obtain the optimal values experimentally using the process described in the last two chapters. The theoretical values calculated by the above formula (6.7.9) are given in Young and Ehrlich, (1960), for different values of \( h^{-1} \) and corresponding to \( m = 1, 2, 3 \) and 4.

Again, we use the \textit{mixed} convergence test with a stopping tolerance of \( \epsilon = 5 \times 10^{-6} \). We applied the Peaceman-Rachford and Douglas-Rachford methods to the four test problems stated earlier (see section 5.5) for cycles of 1, 2 and 3, acceleration parameters. Tables (6.7.1) to (6.7.12) contain the results when
the P-R method is applied to the four test problems. Similar tables (6.7.13) to (6.7.24) contain the corresponding results for the D-R method. These tables are represented by the logarithmic-scale graphs depicted in figures (6.7.1) to (6.7.8).

**Computational Complexity.**

The computational complexity per iteration at each mesh point is as follows:

1) The Peaceman-Rachford method

\[ 14n^2 \text{ multiplications} + 12n^2 \text{ additions} + 2 \text{ r.h.s. unit.} \]

2) The Douglas-Rachford method

\[ 13n^2 \text{ multiplications} + 12n^2 \text{ additions} + 1 \text{ r.h.s. unit.} \]

The r.h.s. unit involves the evaluation of the r.h.s. (nonlinear function) of the problem under investigation.

The total computational complexity for the two methods are given in tables (6.7.25) to (6.7.28) corresponding to the results given by tables (6.7.1) to (6.7.24).
<table>
<thead>
<tr>
<th>$h^{-1}$</th>
<th>Acceleration Parameter</th>
<th>Number of Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>2.1</td>
<td>7</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>13</td>
</tr>
<tr>
<td>15</td>
<td>.6</td>
<td>18</td>
</tr>
<tr>
<td>20</td>
<td>.45</td>
<td>23</td>
</tr>
<tr>
<td>30</td>
<td>.31</td>
<td>33</td>
</tr>
<tr>
<td>40</td>
<td>.11</td>
<td>42</td>
</tr>
<tr>
<td>50</td>
<td>.08</td>
<td>52</td>
</tr>
<tr>
<td>60</td>
<td>.07</td>
<td>61</td>
</tr>
<tr>
<td>80</td>
<td>.048</td>
<td>78</td>
</tr>
</tbody>
</table>

P-R method for $\nabla^2 u = u^2$, with initial guess 7.

Table (6.7.1)

<table>
<thead>
<tr>
<th>$h^{-1}$</th>
<th>Acceleration Parameter</th>
<th>Number of Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>.85</td>
<td>6</td>
</tr>
<tr>
<td>10</td>
<td>.40</td>
<td>10</td>
</tr>
<tr>
<td>15</td>
<td>.25</td>
<td>14</td>
</tr>
<tr>
<td>20</td>
<td>.15</td>
<td>17</td>
</tr>
<tr>
<td>30</td>
<td>.09</td>
<td>24</td>
</tr>
<tr>
<td>40</td>
<td>.06</td>
<td>30</td>
</tr>
<tr>
<td>50</td>
<td>.05</td>
<td>38</td>
</tr>
<tr>
<td>60</td>
<td>.04</td>
<td>44</td>
</tr>
<tr>
<td>80</td>
<td>.026</td>
<td>54</td>
</tr>
</tbody>
</table>

P-R method for $\nabla^2 u = u^4$, with initial guess .01

Table (6.7.2)
<table>
<thead>
<tr>
<th>$h^{-1}$</th>
<th>Acceleration Parameter</th>
<th>Number of Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.80</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>0.35</td>
<td>9</td>
</tr>
<tr>
<td>15</td>
<td>0.20</td>
<td>12</td>
</tr>
<tr>
<td>20</td>
<td>0.15</td>
<td>16</td>
</tr>
<tr>
<td>30</td>
<td>0.10</td>
<td>24</td>
</tr>
<tr>
<td>40</td>
<td>0.06</td>
<td>30</td>
</tr>
<tr>
<td>50</td>
<td>0.05</td>
<td>36</td>
</tr>
<tr>
<td>60</td>
<td>0.04</td>
<td>42</td>
</tr>
<tr>
<td>80</td>
<td>0.026</td>
<td>54</td>
</tr>
</tbody>
</table>

P-R method for $\nabla^2 u = e^u$, with initial guess $-0.01$.

Table (6.7.3)

<table>
<thead>
<tr>
<th>$h^{-1}$</th>
<th>Acceleration Parameter</th>
<th>Number of Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.80</td>
<td>6</td>
</tr>
<tr>
<td>10</td>
<td>0.35</td>
<td>10</td>
</tr>
<tr>
<td>15</td>
<td>0.25</td>
<td>14</td>
</tr>
<tr>
<td>20</td>
<td>0.15</td>
<td>18</td>
</tr>
<tr>
<td>30</td>
<td>0.08</td>
<td>24</td>
</tr>
<tr>
<td>40</td>
<td>0.06</td>
<td>32</td>
</tr>
<tr>
<td>50</td>
<td>0.05</td>
<td>39</td>
</tr>
<tr>
<td>60</td>
<td>0.04</td>
<td>46</td>
</tr>
<tr>
<td>80</td>
<td>0.026</td>
<td>56</td>
</tr>
</tbody>
</table>

P-R method for $\nabla^2 u = -\lambda^2 \sin u$, $\lambda = 1$, with initial guess $0.01$.

Table (6.7.4)
<table>
<thead>
<tr>
<th>h^{-1}</th>
<th>Acc. Parameters</th>
<th>No. of Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>.40</td>
<td>8</td>
</tr>
<tr>
<td>15</td>
<td>.20</td>
<td>10</td>
</tr>
<tr>
<td>20</td>
<td>.10</td>
<td>11</td>
</tr>
<tr>
<td>30</td>
<td>.04</td>
<td>13</td>
</tr>
<tr>
<td>40</td>
<td>.04</td>
<td>16</td>
</tr>
<tr>
<td>50</td>
<td>.03</td>
<td>18</td>
</tr>
<tr>
<td>60</td>
<td>.028</td>
<td>22</td>
</tr>
<tr>
<td>80</td>
<td>.018</td>
<td>24</td>
</tr>
</tbody>
</table>

P-R method for $\nabla^2 u = u^2$, with initial guess 7.

Table (6.7.5)

<table>
<thead>
<tr>
<th>h^{-1}</th>
<th>Acc. Parameters</th>
<th>No. of Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>.38</td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td>.25</td>
<td>8</td>
</tr>
<tr>
<td>15</td>
<td>.17</td>
<td>10</td>
</tr>
<tr>
<td>20</td>
<td>.09</td>
<td>10</td>
</tr>
<tr>
<td>30</td>
<td>.05</td>
<td>12</td>
</tr>
<tr>
<td>40</td>
<td>.03</td>
<td>14</td>
</tr>
<tr>
<td>50</td>
<td>.03</td>
<td>16</td>
</tr>
<tr>
<td>60</td>
<td>.02</td>
<td>16</td>
</tr>
<tr>
<td>80</td>
<td>.02</td>
<td>18</td>
</tr>
</tbody>
</table>

P-R method for $\nabla^2 u = u^4$, with initial guess 0.01.

Table (6.7.6)
<table>
<thead>
<tr>
<th>$h^{-1}$</th>
<th>$\rho_1$</th>
<th>$\rho_2$</th>
<th>No. of Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>$5$</td>
<td>.42</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>$10$</td>
<td>.18</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>$15$</td>
<td>.07</td>
<td>1.3</td>
<td>7</td>
</tr>
<tr>
<td>$20$</td>
<td>.05</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>$30$</td>
<td>.04</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>$40$</td>
<td>.02</td>
<td>1.3</td>
<td>12</td>
</tr>
<tr>
<td>$50$</td>
<td>.02</td>
<td>2.2</td>
<td>14</td>
</tr>
<tr>
<td>$60$</td>
<td>.02</td>
<td>2.9</td>
<td>18</td>
</tr>
<tr>
<td>$80$</td>
<td>.01</td>
<td>3.3</td>
<td>22</td>
</tr>
</tbody>
</table>

P-R method for $V^2u = e^u$, with initial guess $-0.01$.

Table (6.7.7)

<table>
<thead>
<tr>
<th>$h^{-1}$</th>
<th>$\rho_1$</th>
<th>$\rho_2$</th>
<th>No. of Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>$5$</td>
<td>.35</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>$10$</td>
<td>.14</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>$15$</td>
<td>.1</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>$20$</td>
<td>.07</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>$30$</td>
<td>.03</td>
<td>1.2</td>
<td>10</td>
</tr>
<tr>
<td>$40$</td>
<td>.02</td>
<td>1.1</td>
<td>12</td>
</tr>
<tr>
<td>$50$</td>
<td>.02</td>
<td>1.1</td>
<td>14</td>
</tr>
<tr>
<td>$60$</td>
<td>.015</td>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>$80$</td>
<td>.01</td>
<td>1.1</td>
<td>18</td>
</tr>
</tbody>
</table>

P-R method for $V^2u = -\lambda^2 \sin u$, $\lambda = 1$, with initial guess $0.01$.

Table (6.7.8)
<table>
<thead>
<tr>
<th>$h^{-1}$</th>
<th>Acc. Parameters</th>
<th>No. of Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\rho_1$</td>
<td>$\rho_2$</td>
</tr>
<tr>
<td>5</td>
<td>.28</td>
<td>1.3</td>
</tr>
<tr>
<td>10</td>
<td>.25</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>.18</td>
<td>.2</td>
</tr>
<tr>
<td>20</td>
<td>.1</td>
<td>1.9</td>
</tr>
<tr>
<td>30</td>
<td>.05</td>
<td>.9</td>
</tr>
<tr>
<td>40</td>
<td>.02</td>
<td>.1</td>
</tr>
<tr>
<td>50</td>
<td>.01</td>
<td>.1</td>
</tr>
<tr>
<td>60</td>
<td>.01</td>
<td>.09</td>
</tr>
<tr>
<td>80</td>
<td>.01</td>
<td>.04</td>
</tr>
</tbody>
</table>

P-R method for $\nabla^2 u = u^2$, with initial guess 7.

Table (6.7.9)

<table>
<thead>
<tr>
<th>$h^{-1}$</th>
<th>Acc. Parameters</th>
<th>No. of Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\rho_1$</td>
<td>$\rho_2$</td>
</tr>
<tr>
<td>5</td>
<td>.4</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>.1</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>.1</td>
<td>.12</td>
</tr>
<tr>
<td>20</td>
<td>.07</td>
<td>.1</td>
</tr>
<tr>
<td>30</td>
<td>.01</td>
<td>1</td>
</tr>
<tr>
<td>40</td>
<td>.01</td>
<td>.1</td>
</tr>
<tr>
<td>50</td>
<td>.01</td>
<td>.2</td>
</tr>
<tr>
<td>60</td>
<td>.01</td>
<td>.2</td>
</tr>
<tr>
<td>80</td>
<td>.005</td>
<td>.2</td>
</tr>
</tbody>
</table>

P-R method for $\nabla^2 u = u^4$, with initial guess .01.

Table (6.7.10)
<table>
<thead>
<tr>
<th>$h^{-1}$</th>
<th>Acc. Parameters</th>
<th>No. of Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\rho_1$</td>
<td>$\rho_2$</td>
</tr>
<tr>
<td>5</td>
<td>0.5</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>0.13</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>0.07</td>
<td>0.9</td>
</tr>
<tr>
<td>20</td>
<td>0.03</td>
<td>0.8</td>
</tr>
<tr>
<td>30</td>
<td>0.03</td>
<td>0.4</td>
</tr>
<tr>
<td>40</td>
<td>0.015</td>
<td>0.4</td>
</tr>
<tr>
<td>50</td>
<td>0.01</td>
<td>0.36</td>
</tr>
<tr>
<td>60</td>
<td>0.009</td>
<td>0.30</td>
</tr>
<tr>
<td>80</td>
<td>0.007</td>
<td>0.25</td>
</tr>
</tbody>
</table>

P-R method for $\nabla^2 u = e^u$, with initial guess $-0.01$.

**Table (6.7.11)**

<table>
<thead>
<tr>
<th>$h^{-1}$</th>
<th>Acc. Parameters</th>
<th>No. of Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\rho_1$</td>
<td>$\rho_2$</td>
</tr>
<tr>
<td>5</td>
<td>0.4</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>0.2</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>0.04</td>
<td>1.5</td>
</tr>
<tr>
<td>20</td>
<td>0.04</td>
<td>0.8</td>
</tr>
<tr>
<td>30</td>
<td>0.02</td>
<td>0.7</td>
</tr>
<tr>
<td>40</td>
<td>0.02</td>
<td>0.6</td>
</tr>
<tr>
<td>50</td>
<td>0.015</td>
<td>0.5</td>
</tr>
<tr>
<td>60</td>
<td>0.01</td>
<td>0.4</td>
</tr>
<tr>
<td>80</td>
<td>0.005</td>
<td>0.2</td>
</tr>
</tbody>
</table>

P-R method for $\nabla^2 u = -\lambda^2 \sin u$, $\lambda = 1$, with initial guess $0.01$.

**Table (6.7.12)**
### Table (6.7.13)

<table>
<thead>
<tr>
<th>$h^{-1}$</th>
<th>Acc. Parameter</th>
<th>No. of Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1.3</td>
<td>12</td>
</tr>
<tr>
<td>10</td>
<td>.69</td>
<td>19</td>
</tr>
<tr>
<td>15</td>
<td>.43</td>
<td>25</td>
</tr>
<tr>
<td>20</td>
<td>.32</td>
<td>32</td>
</tr>
<tr>
<td>30</td>
<td>.20</td>
<td>44</td>
</tr>
<tr>
<td>40</td>
<td>.15</td>
<td>55</td>
</tr>
<tr>
<td>50</td>
<td>.12</td>
<td>66</td>
</tr>
<tr>
<td>60</td>
<td>.10</td>
<td>77</td>
</tr>
<tr>
<td>80</td>
<td>.073</td>
<td>97</td>
</tr>
</tbody>
</table>

D-R method for $V^2 u = u^2$, with initial guess 7.

### Table (6.7.14)

<table>
<thead>
<tr>
<th>$h^{-1}$</th>
<th>Acc. Parameter</th>
<th>No. of Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>.7</td>
<td>13</td>
</tr>
<tr>
<td>10</td>
<td>.35</td>
<td>18</td>
</tr>
<tr>
<td>15</td>
<td>.22</td>
<td>22</td>
</tr>
<tr>
<td>20</td>
<td>.16</td>
<td>27</td>
</tr>
<tr>
<td>30</td>
<td>.11</td>
<td>36</td>
</tr>
<tr>
<td>40</td>
<td>.08</td>
<td>44</td>
</tr>
<tr>
<td>50</td>
<td>.06</td>
<td>51</td>
</tr>
<tr>
<td>60</td>
<td>.05</td>
<td>58</td>
</tr>
<tr>
<td>80</td>
<td>.036</td>
<td>71</td>
</tr>
</tbody>
</table>

D-R method for $V^2 u = u^4$, with initial guess 0.01.
### Table (6.7.15)

<table>
<thead>
<tr>
<th>$h^{-1}$</th>
<th>Acc. Parameter</th>
<th>No. of Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>.55</td>
<td>13</td>
</tr>
<tr>
<td>10</td>
<td>.29</td>
<td>18</td>
</tr>
<tr>
<td>15</td>
<td>.19</td>
<td>23</td>
</tr>
<tr>
<td>20</td>
<td>.15</td>
<td>29</td>
</tr>
<tr>
<td>30</td>
<td>.10</td>
<td>39</td>
</tr>
<tr>
<td>40</td>
<td>.07</td>
<td>46</td>
</tr>
<tr>
<td>50</td>
<td>.055</td>
<td>54</td>
</tr>
<tr>
<td>60</td>
<td>.045</td>
<td>62</td>
</tr>
<tr>
<td>80</td>
<td>.033</td>
<td>76</td>
</tr>
</tbody>
</table>

D-R Method for $V^2u = e^u$, with initial guess $-0.01$.

### Table (6.7.16)

<table>
<thead>
<tr>
<th>$h^{-1}$</th>
<th>Acc. Parameter</th>
<th>No. of Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>.7</td>
<td>14</td>
</tr>
<tr>
<td>10</td>
<td>.32</td>
<td>18</td>
</tr>
<tr>
<td>15</td>
<td>.21</td>
<td>23</td>
</tr>
<tr>
<td>20</td>
<td>.15</td>
<td>28</td>
</tr>
<tr>
<td>30</td>
<td>.103</td>
<td>36</td>
</tr>
<tr>
<td>40</td>
<td>.076</td>
<td>44</td>
</tr>
<tr>
<td>50</td>
<td>.06</td>
<td>52</td>
</tr>
<tr>
<td>60</td>
<td>.05</td>
<td>60</td>
</tr>
<tr>
<td>80</td>
<td>.036</td>
<td>72</td>
</tr>
</tbody>
</table>

D-R method for $V^2u = -\lambda^2 \sin u$, $\lambda = 1$, with initial guess $0.01$. 
<table>
<thead>
<tr>
<th>$\frac{1}{h}$</th>
<th>Acc. Parameters</th>
<th>No. of Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\rho_1$</td>
<td>$\rho_2$</td>
</tr>
<tr>
<td>5</td>
<td>1.2</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>.50</td>
<td>1.2</td>
</tr>
<tr>
<td>15</td>
<td>.25</td>
<td>1.5</td>
</tr>
<tr>
<td>20</td>
<td>.10</td>
<td>1</td>
</tr>
<tr>
<td>30</td>
<td>.05</td>
<td>1</td>
</tr>
<tr>
<td>40</td>
<td>.05</td>
<td>1.2</td>
</tr>
<tr>
<td>50</td>
<td>.03</td>
<td>1.1</td>
</tr>
<tr>
<td>60</td>
<td>.025</td>
<td>1</td>
</tr>
<tr>
<td>80</td>
<td>.01</td>
<td>.6</td>
</tr>
</tbody>
</table>

D-R method for $V^2 u = u^2$, with initial guess 7.

**Table (6.7.17)**

<table>
<thead>
<tr>
<th>$\frac{1}{h}$</th>
<th>Acc. Parameters</th>
<th>No. of Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\rho_1$</td>
<td>$\rho_2$</td>
</tr>
<tr>
<td>5</td>
<td>.3</td>
<td>1.3</td>
</tr>
<tr>
<td>10</td>
<td>.15</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>.11</td>
<td>2</td>
</tr>
<tr>
<td>20</td>
<td>.07</td>
<td>1.5</td>
</tr>
<tr>
<td>30</td>
<td>.05</td>
<td>2</td>
</tr>
<tr>
<td>40</td>
<td>.03</td>
<td>1.7</td>
</tr>
<tr>
<td>50</td>
<td>.02</td>
<td>1</td>
</tr>
<tr>
<td>60</td>
<td>.018</td>
<td>1</td>
</tr>
<tr>
<td>80</td>
<td>.012</td>
<td>1</td>
</tr>
</tbody>
</table>

D-R method for $V^2 u = u^4$, with initial guess 0.01.

**Table (6.7.18)**
<table>
<thead>
<tr>
<th>$h^{-1}$</th>
<th>Acc. Parameters</th>
<th>No. of Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\rho_1$</td>
<td>$\rho_2$</td>
</tr>
<tr>
<td>5</td>
<td>.40</td>
<td>.6</td>
</tr>
<tr>
<td>10</td>
<td>.15</td>
<td>1.5</td>
</tr>
<tr>
<td>15</td>
<td>.08</td>
<td>2</td>
</tr>
<tr>
<td>20</td>
<td>.06</td>
<td>2.1</td>
</tr>
<tr>
<td>30</td>
<td>.03</td>
<td>1.3</td>
</tr>
<tr>
<td>40</td>
<td>.03</td>
<td>2.1</td>
</tr>
<tr>
<td>50</td>
<td>.02</td>
<td>1.6</td>
</tr>
<tr>
<td>60</td>
<td>.02</td>
<td>2.2</td>
</tr>
<tr>
<td>80</td>
<td>.01</td>
<td>1</td>
</tr>
</tbody>
</table>

D-R method for $V^2u = e^u$, with initial guess $-0.01$.

Table (6.7.19)

<table>
<thead>
<tr>
<th>$h^{-1}$</th>
<th>Acc. Parameters</th>
<th>No. of Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\rho_1$</td>
<td>$\rho_2$</td>
</tr>
<tr>
<td>5</td>
<td>.45</td>
<td>1.2</td>
</tr>
<tr>
<td>10</td>
<td>.20</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>.10</td>
<td>2</td>
</tr>
<tr>
<td>20</td>
<td>.05</td>
<td>1.5</td>
</tr>
<tr>
<td>30</td>
<td>.045</td>
<td>2</td>
</tr>
<tr>
<td>40</td>
<td>.036</td>
<td>2</td>
</tr>
<tr>
<td>50</td>
<td>.02</td>
<td>1.1</td>
</tr>
<tr>
<td>60</td>
<td>.01</td>
<td>.70</td>
</tr>
<tr>
<td>80</td>
<td>.01</td>
<td>.70</td>
</tr>
</tbody>
</table>

D-R method for $V^2u = -\lambda^2 \sin u$, $\lambda = 1$, with initial guess $0.01$.

Table (6.7.20)
<table>
<thead>
<tr>
<th>$h^{-1}$</th>
<th>Acc. Parameters</th>
<th>No. of Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\rho_1$</td>
<td>$\rho_2$</td>
</tr>
<tr>
<td>5</td>
<td>.4</td>
<td>1.8</td>
</tr>
<tr>
<td>10</td>
<td>.22</td>
<td>1.5</td>
</tr>
<tr>
<td>15</td>
<td>.15</td>
<td>1.3</td>
</tr>
<tr>
<td>20</td>
<td>.10</td>
<td>1</td>
</tr>
<tr>
<td>30</td>
<td>.03</td>
<td>.30</td>
</tr>
<tr>
<td>40</td>
<td>.03</td>
<td>.40</td>
</tr>
<tr>
<td>50</td>
<td>.03</td>
<td>.40</td>
</tr>
<tr>
<td>60</td>
<td>.015</td>
<td>.20</td>
</tr>
<tr>
<td>80</td>
<td>.01</td>
<td>.18</td>
</tr>
</tbody>
</table>

D-R method for $\nabla^2 u = u^2$, with initial guess 7.

Table (6.7.21)

<table>
<thead>
<tr>
<th>$h^{-1}$</th>
<th>Acc. Parameters</th>
<th>No. of Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\rho_1$</td>
<td>$\rho_2$</td>
</tr>
<tr>
<td>5</td>
<td>.5</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>.2</td>
<td>.3</td>
</tr>
<tr>
<td>15</td>
<td>.14</td>
<td>.2</td>
</tr>
<tr>
<td>20</td>
<td>.1</td>
<td>.18</td>
</tr>
<tr>
<td>30</td>
<td>.1</td>
<td>.08</td>
</tr>
<tr>
<td>40</td>
<td>.07</td>
<td>.06</td>
</tr>
<tr>
<td>50</td>
<td>.04</td>
<td>.05</td>
</tr>
<tr>
<td>60</td>
<td>.04</td>
<td>.05</td>
</tr>
<tr>
<td>80</td>
<td>.02</td>
<td>.05</td>
</tr>
</tbody>
</table>

D-R method for $\nabla^2 u = u^4$, with initial guess 0.01.

Table (6.7.22)
<table>
<thead>
<tr>
<th>No. of h^{-1}</th>
<th>Acc. Parameters</th>
<th>No. of Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\rho_1$</td>
<td>$\rho_2$</td>
</tr>
<tr>
<td>5</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>0.2</td>
<td>0.3</td>
</tr>
<tr>
<td>15</td>
<td>0.05</td>
<td>0.3</td>
</tr>
<tr>
<td>20</td>
<td>0.05</td>
<td>0.2</td>
</tr>
<tr>
<td>30</td>
<td>0.025</td>
<td>0.19</td>
</tr>
<tr>
<td>40</td>
<td>0.015</td>
<td>0.2</td>
</tr>
<tr>
<td>50</td>
<td>0.01</td>
<td>0.2</td>
</tr>
<tr>
<td>60</td>
<td>0.008</td>
<td>0.15</td>
</tr>
<tr>
<td>80</td>
<td>0.003</td>
<td>0.1</td>
</tr>
</tbody>
</table>

D-R method for $V^2 u = e^u$, with initial guess $-0.01$.

Table (6.7.23)

<table>
<thead>
<tr>
<th>No. of h^{-1}</th>
<th>Acc. Parameters</th>
<th>No. of Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\rho_1$</td>
<td>$\rho_2$</td>
</tr>
<tr>
<td>5</td>
<td>0.7</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>0.5</td>
<td>0.1</td>
</tr>
<tr>
<td>15</td>
<td>0.3</td>
<td>2</td>
</tr>
<tr>
<td>20</td>
<td>0.08</td>
<td>0.2</td>
</tr>
<tr>
<td>30</td>
<td>0.07</td>
<td>0.1</td>
</tr>
<tr>
<td>40</td>
<td>0.06</td>
<td>0.07</td>
</tr>
<tr>
<td>50</td>
<td>0.04</td>
<td>0.05</td>
</tr>
<tr>
<td>60</td>
<td>0.04</td>
<td>0.05</td>
</tr>
<tr>
<td>80</td>
<td>0.01</td>
<td>0.3</td>
</tr>
</tbody>
</table>

D-R method for $V^2 u = -\lambda^2 \sin u$. $\lambda = 1$, with initial guess 0.01.

Table (6.7.24)
Problem 1

Legend

- 1-Param.
- 2-Param.
- 3-Param.

Number Of Iterations

\( h^{-1} \) vs. Number of Iterations

P-R method for \( \nabla^2 u = u^2 \)

Figure (6,7,1)

Problem 2

Legend

- 1-Param.
- 2-Param.
- 3-Param.

Number Of Iterations

\( h^{-1} \) vs. Number of Iterations

P-R method for \( \nabla^2 u = u^4 \)

Figure (6,7,2)
Problem 3

Legend
- 1-Param.
- 2-Param.
- 3-Param.

Number of Iterations vs. $h^{-1}$

$P-R$ method for $\nabla^2 u = e^u$

Figure (6,7,3)

Problem 4

Legend
- 1-Param.
- 2-Param.
- 3-Param.

Number of Iterations vs. $h^{-1}$

$P-R$ method for $\nabla^2 u = -\lambda^2 \sin(u), \lambda=1$

Figure (6,7,4)
Problem 1

Legend

- 1-Param.
- 2-Param.
- 3-Param.

D-R method for $\nabla^2 u = u^2$

Figure (6,7,5)

Problem 2

Legend

- 1-Param.
- 2-Param.
- 3-Param.

D-R method for $\nabla^2 u = u^4$

Figure (6,7,6)
Problem 3

Legend

- 1-Param.
- 2-Param.
- 3-Param.

D-R method for $\nabla^2 u = e^u$

Figure (6.7.7)

Problem 4

Legend

- 1-Param.
- 2-Param.
- 3-Param.

D-R method for $\nabla^2 u = -\lambda^2\sin(u), \lambda=1$

Figure (6.7.8)
<table>
<thead>
<tr>
<th>Method</th>
<th>P-R Method</th>
<th>D-R Method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>m = 1</td>
<td>m = 2</td>
</tr>
<tr>
<td>h-1</td>
<td>M</td>
<td>A</td>
</tr>
<tr>
<td>5</td>
<td>98 (n^2)</td>
<td>84 (n^2)</td>
</tr>
<tr>
<td>10</td>
<td>182 (n^2)</td>
<td>156 (n^2)</td>
</tr>
<tr>
<td>15</td>
<td>252 (n^2)</td>
<td>216 (n^2)</td>
</tr>
<tr>
<td>20</td>
<td>322 (n^2)</td>
<td>276 (n^2)</td>
</tr>
<tr>
<td>30</td>
<td>462 (n^2)</td>
<td>396 (n^2)</td>
</tr>
<tr>
<td>40</td>
<td>588 (n^2)</td>
<td>504 (n^2)</td>
</tr>
<tr>
<td>50</td>
<td>728 (n^2)</td>
<td>624 (n^2)</td>
</tr>
<tr>
<td>60</td>
<td>854 (n^2)</td>
<td>732 (n^2)</td>
</tr>
<tr>
<td>80</td>
<td>1092 (n^2)</td>
<td>936 (n^2)</td>
</tr>
</tbody>
</table>

Computational Complexity for \(\nabla^2 u = u^2\)

Table (6.7.25)
<table>
<thead>
<tr>
<th>Method</th>
<th>( h^{-1} )</th>
<th>( m = 1 )</th>
<th>( m = 2 )</th>
<th>( m = 3 )</th>
<th>( m = 1 )</th>
<th>( m = 2 )</th>
<th>( m = 3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M</td>
<td>A</td>
<td>M</td>
<td>A</td>
<td>M</td>
<td>A</td>
<td>M</td>
</tr>
<tr>
<td>5</td>
<td>84 ( n^2 )</td>
<td>72 ( n^2 )</td>
<td>42 ( n^2 )</td>
<td>36 ( n^2 )</td>
<td>56 ( n^2 )</td>
<td>48 ( n^2 )</td>
<td>169 ( n^2 )</td>
</tr>
<tr>
<td>10</td>
<td>140 ( n^2 )</td>
<td>120 ( n^2 )</td>
<td>112 ( n^2 )</td>
<td>96 ( n^2 )</td>
<td>56 ( n^2 )</td>
<td>48 ( n^2 )</td>
<td>234 ( n^2 )</td>
</tr>
<tr>
<td>15</td>
<td>196 ( n^2 )</td>
<td>168 ( n^2 )</td>
<td>140 ( n^2 )</td>
<td>120 ( n^2 )</td>
<td>112 ( n^2 )</td>
<td>96 ( n^2 )</td>
<td>286 ( n^2 )</td>
</tr>
<tr>
<td>20</td>
<td>238 ( n^2 )</td>
<td>204 ( n^2 )</td>
<td>140 ( n^2 )</td>
<td>120 ( n^2 )</td>
<td>126 ( n^2 )</td>
<td>108 ( n^2 )</td>
<td>351 ( n^2 )</td>
</tr>
<tr>
<td>30</td>
<td>336 ( n^2 )</td>
<td>288 ( n^2 )</td>
<td>168 ( n^2 )</td>
<td>144 ( n^2 )</td>
<td>140 ( n^2 )</td>
<td>120 ( n^2 )</td>
<td>468 ( n^2 )</td>
</tr>
<tr>
<td>40</td>
<td>420 ( n^2 )</td>
<td>360 ( n^2 )</td>
<td>196 ( n^2 )</td>
<td>168 ( n^2 )</td>
<td>140 ( n^2 )</td>
<td>120 ( n^2 )</td>
<td>572 ( n^2 )</td>
</tr>
<tr>
<td>50</td>
<td>532 ( n^2 )</td>
<td>456 ( n^2 )</td>
<td>224 ( n^2 )</td>
<td>192 ( n^2 )</td>
<td>168 ( n^2 )</td>
<td>144 ( n^2 )</td>
<td>663 ( n^2 )</td>
</tr>
<tr>
<td>60</td>
<td>616 ( n^2 )</td>
<td>528 ( n^2 )</td>
<td>224 ( n^2 )</td>
<td>192 ( n^2 )</td>
<td>168 ( n^2 )</td>
<td>144 ( n^2 )</td>
<td>754 ( n^2 )</td>
</tr>
<tr>
<td>80</td>
<td>756 ( n^2 )</td>
<td>648 ( n^2 )</td>
<td>252 ( n^2 )</td>
<td>216 ( n^2 )</td>
<td>168 ( n^2 )</td>
<td>144 ( n^2 )</td>
<td>923 ( n^2 )</td>
</tr>
</tbody>
</table>

Computational Complexity for \( V^2 u = u^4 \)

Table (6.7.26)
<table>
<thead>
<tr>
<th>Method</th>
<th>P-R Method</th>
<th>D-R Method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>m = 1</td>
<td>m = 2</td>
</tr>
<tr>
<td></td>
<td>M</td>
<td>A</td>
</tr>
<tr>
<td>5</td>
<td>70 n^2</td>
<td>60 n^2</td>
</tr>
<tr>
<td>10</td>
<td>126 n^2</td>
<td>108 n^2</td>
</tr>
<tr>
<td>15</td>
<td>168 n^2</td>
<td>144 n^2</td>
</tr>
<tr>
<td>20</td>
<td>224 n^2</td>
<td>192 n^2</td>
</tr>
<tr>
<td>30</td>
<td>336 n^2</td>
<td>288 n^2</td>
</tr>
<tr>
<td>40</td>
<td>420 n^2</td>
<td>360 n^2</td>
</tr>
<tr>
<td>50</td>
<td>504 n^2</td>
<td>432 n^2</td>
</tr>
<tr>
<td>60</td>
<td>588 n^2</td>
<td>504 n^2</td>
</tr>
<tr>
<td>80</td>
<td>756 n^2</td>
<td>648 n^2</td>
</tr>
</tbody>
</table>

Computational Complexity for $v^2 u = e^u$

Table (6.7.27)
<table>
<thead>
<tr>
<th>Method</th>
<th>P-R Method</th>
<th>D-R Method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>m = 1</td>
<td>m = 2</td>
</tr>
<tr>
<td>h^{-1}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>84n^2</td>
<td>72n^2</td>
</tr>
<tr>
<td></td>
<td>182n^2</td>
<td>168n^2</td>
</tr>
<tr>
<td>10</td>
<td>140n^2</td>
<td>120n^2</td>
</tr>
<tr>
<td></td>
<td>234n^2</td>
<td>216n^2</td>
</tr>
<tr>
<td>15</td>
<td>196n^2</td>
<td>168n^2</td>
</tr>
<tr>
<td></td>
<td>299n^2</td>
<td>276n^2</td>
</tr>
<tr>
<td>20</td>
<td>252n^2</td>
<td>216n^2</td>
</tr>
<tr>
<td></td>
<td>364n^2</td>
<td>336n^2</td>
</tr>
<tr>
<td>30</td>
<td>336n^2</td>
<td>288n^2</td>
</tr>
<tr>
<td></td>
<td>468n^2</td>
<td>432n^2</td>
</tr>
<tr>
<td>40</td>
<td>448n^2</td>
<td>384n^2</td>
</tr>
<tr>
<td></td>
<td>572n^2</td>
<td>528n^2</td>
</tr>
<tr>
<td>50</td>
<td>546n^2</td>
<td>468n^2</td>
</tr>
<tr>
<td></td>
<td>676n^2</td>
<td>624n^2</td>
</tr>
<tr>
<td>60</td>
<td>644n^2</td>
<td>552n^2</td>
</tr>
<tr>
<td></td>
<td>780n^2</td>
<td>720n^2</td>
</tr>
<tr>
<td>80</td>
<td>784n^2</td>
<td>672n^2</td>
</tr>
<tr>
<td></td>
<td>936n^2</td>
<td>864n^2</td>
</tr>
</tbody>
</table>

Computational Complexity for $\nabla^2 u = -\lambda^2 \sin u$, $\lambda = 1$.

Table (6.7.28)
6.7.4 Discussion and Comments.

Since the Peaceman-Rachford parameters and the Wachspress parameters are obtained theoretically, we did not consider them as optimal parameters when we solved the test problems (Birkhoff, et al. (1962) solved their given problems for these parameters). Instead we searched for experimental optimal values, and as noticed from the tables they decrease as the mesh size increases unlike the increasing values of the parameters obtained by using formula (6.7.9).

Now, we examine the graphs (6.7.1) - (6.7.4) for the P-R method which show the slopes of the straight lines as follows:

<table>
<thead>
<tr>
<th>Problem</th>
<th>m = 1</th>
<th>m = 2</th>
<th>m = 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem 1</td>
<td>.87</td>
<td>.55</td>
<td>.38</td>
</tr>
<tr>
<td>Problem 2</td>
<td>.8</td>
<td>.57</td>
<td>.44</td>
</tr>
<tr>
<td>Problem 3</td>
<td>.87</td>
<td>.58</td>
<td>.35</td>
</tr>
<tr>
<td>Problem 4</td>
<td>.83</td>
<td>.57</td>
<td>.34</td>
</tr>
</tbody>
</table>

Table (6.7.29)

From the table above, we notice that for m = 1, although all the points lie apparently along straight lines, the slopes are less than 1 which does not agree exactly with the theory where the slope should be 1. For m = 2, the slopes are nearly $\frac{1}{2}$, while for m = 3 the slopes are nearly $\frac{1}{3}$ except for problem 2 where the corresponding slope is greater than $\frac{1}{3}$. The important
fact is that the slopes decrease when \( m \) increases.

For the D-R method, we have this table of slopes:

<table>
<thead>
<tr>
<th>Problem</th>
<th>( m = 1 )</th>
<th>( m = 2 )</th>
<th>( m = 3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem 1</td>
<td>.8</td>
<td>.56</td>
<td>.32</td>
</tr>
<tr>
<td>Problem 2</td>
<td>.62</td>
<td>.2</td>
<td>.25</td>
</tr>
<tr>
<td>Problem 3</td>
<td>.65</td>
<td>.25</td>
<td>.16</td>
</tr>
<tr>
<td>Problem 4</td>
<td>.61</td>
<td>.192</td>
<td>.194</td>
</tr>
</tbody>
</table>

Table (6.7.30)

The outcome here is different from that of the P-R method where the slopes are much less than \( \frac{1}{m} \) except in the case of problem 1. However, unlike the case of the P-R method, the graphs show that the points approximately lie on a straight line especially for \( m = 1 \). Comparing table (6.7.29) with table (6.7.30) shows that in general the D-R method has slopes less than the slopes of the P-R method but this does not imply that the D-R method converges faster, on the contrary, it is slower since it possesses greater numbers of iterations in comparison with the P-R method.
Chapter Seven

VARIATIONAL METHODS FOR SOLVING NONLINEAR PROBLEMS

7.1 Conjugate Gradient Method
   7.1.1 Introduction
   7.1.2 Conjugate Gradient Method
   7.1.3 Conjugate Gradient Acceleration

7.2 Preconditioning the Conjugate Gradient Method
   7.2.1 Introduction
   7.2.2 Preconditioning by Incomplete Factorization
   7.2.3 Preconditioning by SSOR Method

7.3 Mildly Nonlinear Problems
   7.3.1 Method of Solution
   7.3.2 Experimental Results

7.4 The LU-Decomposition Algorithm for Two Dimensional P.D.E.'s
   7.4.1 Introduction
   7.4.2 The Approximate LU-Decomposition Algorithm (The ALUBOT Algorithm)

7.5 Derivation of Newton-ALUBOT Method

7.6 Iterative Procedures and Experimental Results
7.1 CONJUGATE GRADIENT METHOD.

7.1.1 Introduction

Hestenes and Stiefel (1952) were the first to present the classic conjugate gradient method to solve systems of linear algebraic equations. It is a special case of the conjugate-direction method and can be derived from the steepest descent optimization technique. The conjugate gradient (CG) method is an iterative method which converges to the actual solution of an $N \times N$ system, in the absence of round off error, in at most $N$ iterations.

The CG method, until recently, was not widely considered as an efficient method despite a number of supporting papers which have appeared like those by Daniel (1967), Reid (1971, 1972) and Concus et al. (1976). The CG method is actually a family of methods (Hestenes, 1956). Each is regarded as a technique to accelerate some basic iterative methods (described in Chapter 4). It is known as conjugate gradient acceleration, which can be applied under similar conditions as required for the Chebyshev acceleration.

Attractive features of the CG methods are that no further special matrix properties are required for $A$ and no acceleration parameters are to be estimated. Also the effect of the round off error on actual implementation does not prevent convergence only delays it.

Furthermore, Reid (1972), showed that if the coefficient matrix is two-cyclic, then the work required for the CG method may be approximately halved.
7.1.2 Conjugate Gradient Method

Consider the system of linear equations

$$ A u = b $$

where $A$ is an $n \times n$ symmetric, positive definite matrix, $u$ is an $n \times 1$ unknown vector and $b$ is an $n \times 1$ constant vector.

Let us define the residual vector at the $k^{th}$ iterate as

$$ r(k) = b - A u^{(k)} $$

The system (7.1.1) can be regarded as the gradient of the quadratic function

$$ F(u) = \frac{1}{2} (u, A u) - (b, u) $$

Finding the solution of the system (7.1.1) is equivalent to minimizing the function $F(u)$. The gradient of $F(u)$ is given by

$$ \nabla F(u) = b - A u = r $$

where $r$ is the residual given by (7.1.2).

Before describing the CG method, we derive the steepest descent method to minimize $F(u)$.

To find the minimum of $F(u)$, we move in the direction of

$$ -\nabla F(u^{(k)}) $$

in order to obtain an improved approximation $u^{(k+1)}$ to the solution $u$, i.e.

$$ u^{(k+1)} = u^{(k)} - \alpha_k \nabla F(u^{(k)}) $$

or

$$ u^{(k+1)} = u^{(k)} + \alpha_k r^{(k)} $$

where $\alpha_k$ is chosen to minimize $F(u^{(k+1)})$.

Substituting equation (7.1.5) in equation (7.1.3) and differentiating will result in the following equation to determine
Thus, the steepest descent method is given by the following equations,

\[
\begin{align*}
\mathbf{u}^{(0)} & \text{ is given initial starting vector.} \\
\mathbf{u}^{(k+1)} &= \mathbf{u}^{(k)} + \alpha_k \mathbf{r}^{(k)} , \quad \text{for } k = 0,1,\ldots \\
\end{align*}
\]

where

\[
\mathbf{r}^{(k)} = \mathbf{b} - A \mathbf{u}^{(k)}
\]

and

\[
\alpha_k = \frac{\mathbf{r}^{(k)} \cdot \mathbf{r}^{(k)}}{\mathbf{r}^{(k)} \cdot A \mathbf{r}^{(k)}} .
\]

We modify the steepest descent method by changing the direction vector, which will result in different methods such as the CG method etc., in which \( \alpha_k \) is chosen so that the direction vectors \( p^{(k)} \)'s are \( A \)-conjugate such that \( (p^{(k)}, A p^{(k-1)}) = 0 \).

The CG method is given by:

Let \( k = 0, p^{(0)} = r^{(0)} \), for arbitrary initial vector \( u^{(0)} \).

For \( k = 0,1,2,\ldots \), we compute the vectors \( u^{(k)}, p^{(k)} \) and \( r^{(k)} \) from the relations

\[
\begin{align*}
\mathbf{u}^{(k+1)} &= \mathbf{u}^{(k)} + \alpha_k \mathbf{p}^{(k)} \\
\mathbf{p}^{(k+1)} &= \mathbf{r}^{(k+1)} + \beta_k \mathbf{p}^{(k)} , \quad (7.1.6a) \\
\beta_k &= -\frac{(\mathbf{r}^{(k+1)} \cdot A \mathbf{p}^{(k)})}{(\mathbf{p}^{(k+1)} \cdot A \mathbf{p}^{(k)})} , \\
\mathbf{r}^{(k+1)} &= \mathbf{b} - A \mathbf{u}^{(k+1)} ,
\end{align*}
\]
and
\[ \alpha_k = \frac{\langle p(k), r(k) \rangle}{\langle p(k), A^T p(k) \rangle} \]

The quantities \( \beta_k \), \( r^{(k+1)} \) and \( \alpha_k \) are given equivalently by
\[ \beta_k = \frac{\langle r^{(k+1)}, r^{(k+1)} \rangle}{\langle r^{(k)}, r^{(k)} \rangle} \]
\[ r^{(k+1)} = r^{(k)} - \alpha_k A p^{(k)} \] (7.1.6b)

and
\[ \alpha_k = \frac{\langle r^{(k)}, r^{(k)} \rangle}{\langle p^{(k)}, A^T p^{(k)} \rangle} \]

Hestenes and Stiefel suggested that the first expressions (given in (7.1.6a)) for \( \alpha_k \), \( \beta_k \) are easier to compute, but the second forms of \( \alpha_k \) and \( \beta_k \) give better results.

Hestenes and Stiefel (1952) show that the residual vectors \( r^{(k)} \), \( k = 0, 1, \ldots \) are mutually orthogonal and the direction vectors \( p^{(k)} \), \( k = 0, 1, \ldots \) are mutually \( A \)-conjugate, i.e.
\[ \langle r^{(i)}, r^{(j)} \rangle = 0 \quad \text{for } i \neq j \] (7.1.7)
\[ \langle p^{(i)}, A p^{(j)} \rangle = 0 \quad \text{for } i \neq j \] (7.1.8)
\[ \langle r^{(i)}, A p^{(j)} \rangle = 0 \quad \text{for } i \neq j \text{ and } i \neq j+1. \] (7.1.9)

The CG method in the above form is a first degree iterative method. A second degree iterative method can be derived (Engeli et al. 1959) by eliminating the direction vectors \( p^{(k)} \) from the formulae (7.1.5), and after some rearrangements, we have;
\[ u^{(k+1)} = \omega_{k+1} \left\{ \gamma_{k+1} \frac{r^{(k)}}{r^{(k)}} + u^{(k)} \right\} + (1 - \omega_{k+1}) u^{(k-1)} \]  

(7.1.10)

where

\[ \gamma_{k+1} = \frac{(r^{(k)}, r^{(k)})}{(r^{(k)}, A r^{(k)})} \]  

(7.1.11)

and the parameters \( \omega_k \) are given by

\[ \omega_1 = 1 \]

\[ \omega_{k+1} = \left[ 1 - \frac{(r^{(k)}, r^{(k)})}{(r^{(k-1)}, r^{(k-1)})} \gamma_{k+1} \omega_k \right]^{-1}, \]  

for \( k \geq 1 \) .

(7.1.12)

Apparently, this seems similar to the Chebyshev acceleration procedure except that the acceleration parameters do not involve the largest and the smallest eigenvalues of the iteration matrix of the method.

### 7.1.3 Conjugate Gradient Acceleration

The CG method can be modified to act as a polynomial acceleration procedure in conjunction with the basic iterative methods. To do so, we consider the basic stationary iterative method to solve the system (7.1.1), of the form:

\[ u^{(k+1)} = G u^{(k)} + k \]  

(7.1.13)

where \( G \) is the \( n \times n \) iteration matrix, \( u^{(k+1)}, u^{(k)} \) and \( k \) are \( n \times 1 \) vectors and such that

\[ G = I - Q^{-1}A \] , \hspace{1cm} \[ k = Q^{-1}b \] ,

(7.1.14)

for some nonsingular splitting matrix \( Q \).
Now, we consider the related linear system

\[(I-G) \, u = k\quad , \quad (7.1.15)\]

which has the same solution as the system \((7.1.1)\).

Let \(W\) be a nonsingular symmetrization matrix (see definition 2.4.3) which transforms the system \((7.1.15)\) to

\[W(I-G)\, W^{-1}(W \, u) = W \, k\quad , \quad (7.1.16)\]

such that \(W(I-G)W^{-1}\) is symmetric positive definite. The system \((7.1.16)\) is often called the preconditioned system (explained later). We define the pseudo-residual vector

\[\delta^{(k)} = G \, u^{(k)} + k - u^{(k)}\quad . \quad (7.1.17)\]

The system \((7.1.16)\) can be written in the form

\[\hat{A} \, \hat{u} = \hat{b}\quad , \quad (7.1.18)\]

where \(\hat{A} = W(I-G)W^{-1}\), \(\hat{u} = W \, u\) and \(\hat{b} = W \, k\). Now, the basic CG method (as given in \((7.1.5), (7.1.6)\)) is applied to the system \((7.1.18)\), which will have the form

\[u^{(0)}\text{ is arbitrary, } p^{(0)} = \delta^{(0)}\text{ for } k = 0\] \[u^{(k+1)} \text{ arbitrary, } p^{(k+1)} = \delta^{(k+1)} + \beta_k \, p^{(k)}\quad , \quad (7.1.19)\]

for \(k = 0, 1, 2, \ldots\) compute

\[u^{(k+1)} = u^{(k)} + \alpha_k \, p^{(k)}\]

\[p^{(k+1)} = \delta^{(k+1)} + \beta_k \, p^{(k)}\]

\[\beta_k = \frac{(W \, \delta^{(k+1)}, W \, \delta^{(k+1)})}{(W \, \delta^{(k)}, W \, \delta^{(k)})}\]
and 
\[ \alpha_k = \frac{(W \delta^{(k)}, W \delta^{(k)})}{(W p^{(k)}, W(I-G)p^{(k)})} . \]

Corresponding to (7.1.7) - (7.1.9) (see Hageman and Young, 1981), the pseudo-residual vectors \( \delta^{(k)} \), \( k = 0, 1, \ldots \) are mutually \( W \)-orthogonal, i.e.

\[ (W \delta^{(i)}, W \delta^{(j)}) = 0 \], \( i \neq j \), \hspace{1cm} (7.1.20) \]

and the direction vectors \( p^{(k)} \)'s are such that

\[ (p^{(i)}, W(I-G)p^{(j)}) = 0, \hspace{1cm} i \neq j \]. \hspace{1cm} (7.1.21) \]

Analogous to equation (7.1.10) - (7.1.12), a second degree (three-term recurrence) form of the CG acceleration procedure is given by

\[ u^{(k+1)} = \omega_{k+1} \left\{ \gamma_{k+1} (G u^{(k)} + k) + (1 - \gamma_{k+1}) u^{(k)} \right\} + (1 - \omega_{k+1}) u^{(k-1)} \]

\[ = \omega_{k+1} \left\{ \gamma_{k+1} \delta^{(k)} - u^{(k)} \right\} + (1 - \omega_{k+1}) u^{(k-1)}, \hspace{1cm} (7.1.22) \]

where

\[ \gamma_{k+1} = \left[ 1 - \frac{(W \delta^{(k)}, W \delta^{(k)})}{(W \delta^{(k)}, W \delta^{(k)})} \right]^{-1} \hspace{1cm} (7.1.23) \]

and

\[ \omega_{k+1} = 1 \]

\[ \omega_{k+1} = \left[ 1 - \frac{k+1}{\omega_k \gamma_k} \frac{(W \delta^{(k)}, W \delta^{(k-1)})}{(W \delta^{(k-1)}, W \delta^{(k-1)})} \right]^{-1}, \hspace{1cm} (7.1.24) \]

\[ \text{for } k \geq 1. \]

The above acceleration procedure can be applied to the SSOR method (and others) in a similar way to the Chebyshev acceleration but often requires a smaller number of iterations and the acceleration
parameters do not involve the largest and the smallest eigenvalues. Results of Reid (1972) show that the two-term form in (7.1.19) is somewhat more efficient than the three-term CG acceleration form in (7.1.22) - (7.1.24). It can be shown (Colgan, 1984) that the CG acceleration procedure minimizes the \( \| W (I - G) \| \frac{1}{2} \) norm of the error vector \( e^{(k)} = u^{(k)} - u \), where \( u \) is the exact solution, compared with any polynomial acceleration procedure applied to the same basic iterative method.

7.2 PRECONDITIONING THE CONJUGATE GRADIENT METHOD.

7.2.1 Introduction

For the linear system

\[
A \ u = b
\]  

(7.2.1)

the CG method in its basic form converges slowly when applied to systems resulting from finite difference equations of elliptic boundary value problems. To improve the convergence rate, Evans (1967), (1973) applied the preconditioning technique to the CG method. Since then, a number of authors have studied and described the preconditioning of the CG method to solve the system (7.2.1) (for example: Meijerink and Van der Vorst (1974); Concus et al. (1976)). The simplest procedure to precondition the system (7.2.1) is to find an easily inverted symmetric positive definite matrix \( H \) such that \( H \) is a good approximate of the matrix \( A \) (i.e. \( H^{-1} A \approx I \)), so the system (7.2.1) can be replaced by the system

\[
H^{-1} A \ u = H^{-1} b
\]  

(7.2.2)
Now we apply the basic CG method to the system (7.2.2), we have
the preconditioned CG (PCG) algorithm

\[ u^{(0)} \text{ is arbitrary} \]

\[ r^{(0)} = b - A u^{(0)} \]

and \[ p^{(0)} = \frac{b}{r} \]

for \( k = 0,1,2,\ldots \) compute the vectors \( u^{(k)}, r^{(k)} \) and \( p^{(k)} \) from

\[ u^{(k+1)} = u^{(k)} + \alpha_k p_k \]

\[ r^{(k+1)} = r^{(k)} - \alpha_k A p_k \]

\[ \alpha_k = \frac{(r^{(k)}, H^{-1} r^{(k)})}{(r^{(k+1)}, H^{-1} r^{(k+1)})} \]

with \[ r^{(k+1)} = r^{(k)} - \alpha_k A p_k \]

\[ p^{(k+1)} = H^{-1} r^{(k+1)} + \beta_k p^{(k)} \]

and \[ \beta_k = \frac{(r^{(k+1)}, H^{-1} r^{(k+1)})}{(r^{(k)}, H^{-1} r^{(k)})} \]

For the matrix-vector multiplication \( H^{-1} r^{(k)} \), we solve the auxiliary system:

\[ H z^{(k)} = r^{(k)}, \quad k = 0,1,2,\ldots \quad (7.2.3) \]

for the additional vectors \( z^{(0)}, z^{(1)}, \ldots \)

The efficiency of the method depends upon the requirements

(1) \( H \) is close approximate of \( A \) so that a large number of the eigenvalues of \( H^{-1} A \) are close and nearly equal 1 and \( H^{-1} A \) has spectral condition number smaller than that of \( A \).
(2) The matrix H is easily constructed so that the system (7.2.3) is easy to solve.

The matrix H is referred to as the preconditioning matrix which is required to meet the above two requirements. There are many possible ways of determining H; we will discuss two of them in this chapter, namely preconditioning by incomplete factorization and preconditioning by the SSOR method. The preconditioning by these two methods preserves at most the same sparsity pattern of A to $H^{-1}A$ which means no more storage is needed.

7.2.2 Preconditioning by Incomplete Factorization

Many researchers including Mitchell and Griffiths (1980) have used the incomplete factorization strategy in conjunction with the CG method to provide a powerful composite iterative algorithm.

The symmetric positive definite matrix A of the sparse system (7.2.1) is factorized by the unique factorization (Varga, 1962) of the form

$$A = LDL^T,$$  \hspace{1cm} (7.2.3)

where D is a positive diagonal matrix and L is a unit lower triangular matrix.

However, in order to apply preconditioning to this system we proceed to apply an incomplete factorization which is not a direct factorization in the sense of constructing D and L so that (7.2.3) holds exactly, instead we construct an $L_S$ such that $L_S$ has non-zero entries only in those positions which correspond to the non-zero in the lower triangle of A. The product $L_DS_1LT_S$ is
then obtained and the elements of $D_s$ and $L_s$ are computed by equating only the non-zero elements of $A$ to those of $L_s D_s L_s^T$.

We describe the process through considering the Laplace two-dimensional model problem defined on a unit square region divided into a rectangular mesh of side $h = 1/n$ and having $N = (n-1)^2$ mesh points.

Using the standard five-point difference formula as developed in Chapter 4, produces the coefficient matrix $A$ which has the structure

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

Subsequently, the $k$th component of $A u$ may be written in the form

$$(A u)_k = a_{k,k} u_k + a_{k,k+1} u_{k+1} + a_{k,k+n} u_{k+n} + a_{k-1,k} u_{k-1}$$

$$+ a_{k-n,k} u_{k-n},$$

for $k = 1, 2, \ldots, N$ \hfill (7.2.4)

where $a_{j,k} = 0$ for $j \leq 0$ and $a_{k,j} = 0$ for $j > N$.

The form of (7.2.4) is depicted by the computational molecule given by figure (7.2.1).
Suppose that the matrix $D_s$ has diagonal elements $d_1, d_2, \ldots, d_N$ and $L_s$ is the unit lower triangular matrix such that

$$(L_s u)_k = u_k + \ell_{k-1,k} u_{k-1} + \ell_{k-n,k} u_{k-n}, \quad \text{for } k = 1, 2, \ldots, N$$

(7.2.5)

with $\ell_{j,k} = 0$ for $j < k$.

The structure of $L_s$ and the computational molecule of the form (7.2.5) are illustrated in Figure (7.2.2).

By (7.2.5), we have

$$(L_s^T u)_k = u_k + \ell_{k,k+1} u_{k+1} + \ell_{k,n} u_{k+n}, \quad (7.2.6)$$

and therefore
\[(D_sL_s^T u)_k = d_n (u_k + \ell_{k,k+1} u_{k+1} + \ell_{k,k+n} u_{k+n}) \quad \text{(7.2.7)}\]

Now
\[L_s (D_sL_s^T u)_k = (D_sL_s^T u)_k + \ell_{k-1,k} (D_sL_s^T u)_{k-1} + \ell_{k-n,k} (D_sL_s^T u)_{k-n} \quad \text{(7.2.8)}\]

which becomes
\[(L_sD_sL_s^T u)_k = (d_k + d_{k-1} \ell_{k-1,k}^2 + d_{k-n} \ell_{k-n,k}^2) u_k + d_k (\ell_{k,k+1} u_{k+1} + \ell_{k,k+n} u_{k+n}) + d_{k-1} \ell_{k-1,k} (u_{k-1} + \ell_{k-1,k+n-1} u_{k+n-1}) + d_{k-n} \ell_{k-n,k} (u_{k-n} + \ell_{k-n,k+n+1} u_{k+n+1}) \quad \text{(7.2.9)}\]

The matrix \(L_sD_sL_s^T\) has the following structure

\[
L_sD_sL_s^T = \begin{bmatrix}
\ldots & & & \\
& \ddots & & \\
& & \ddots & \\
& & & 0
\end{bmatrix}
\]

By equating the coefficients of \(u_k, u_{k+1}, u_{k+N}\) in (7.2.4) and (7.2.9), we obtain
\[
d_k = a_{k,k} - d_{k-1} \ell_{k-1,k}^2 - d_{k-n} \ell_{k-n,k}^2, \quad \text{for } k = 1, 2, \ldots, N \quad \text{(7.2.10)}
\]
\[
\ell_{k,k+1} = a_{k,k+1}/d_k, \quad \text{for } k = 1, 2, \ldots, N-1 \quad \text{(7.2.11)}
\]
\[ l_{k,k+n} = \frac{a_{k,k+n}}{d_k}, \text{ for } k = 1, 2, \ldots, N-n \]  

with \[ d_j = \ell_j,k = 0 \text{ for } j < 0 \] and \[ a_k = l_k = 0, \text{ for } j > N. \]

After the factorization is finished, we solve the system by the usual forward-backward substitution processes.

7.2.3 Preconditioning by SSOR Method

Here, we take advantage of the formation of the SSOR method to obtain a preconditioning matrix \( H \).

Let \( A \) have the following partitioning

\[ A = D - L - U \]  

(7.2.13)

where \( D \) is a diagonal matrix and \( L \) and \( U \) are strictly lower and strictly upper triangular matrices. We recall the theory of the SOR and the SSOR methods (section 4.3), as the first half of an SSOR iteration is an SOR iteration, we have from equation (4.3.18):

\[ (D - \omega L)(u^{(k+\frac{1}{2})} - u^k) = \omega r^{(k)} \]  

(7.2.14)

where \[ r^{(k)} = b - A u^{(k)} \].

Similarly, we obtain the second half of the iteration as

\[ (D - \omega U)(u^{(k+1)} - u^{(k+\frac{1}{2})}) = \omega(b - A u^{(k+\frac{1}{2})}) \]  

(7.2.15)

Now, by eliminating \( u^{(k+\frac{1}{2})} \) between (7.2.14) and (7.2.15) we obtain,

\[ (D - \omega L)D^{-1}(D - \omega U)(u^{(k+1)} - u^{(k)}) = \omega(2 - \omega) r^{(k)}. \]  

(7.2.16)

To use the SSOR method in conjunction with the basic CG method, one iteration of the SSOR is interposed between each
 cycle of the basic CG algorithm which is equivalent to pre-
conditioning the CG method by choosing the preconditioning matrix 
H as
\[ H = \frac{1}{\omega(2-\omega)} (D - \omega L)D^{-1}(D - \omega U). \]  (7.2.17)

Here \( H \) is a product of triangular matrices which makes it easy 
to implement the matrix-vector multiplications in order to carry 
out a PCG iteration, i.e. forward and backward substitutions. 
Also by this choice, i.e. preconditioning by the SSOR method, 
we do not have to construct any new entries apart from those in 
\[ \frac{1}{\omega(2-\omega)} D . \]

7.3 MILDLY NONLINEAR PROBLEMS

In this section, we solve the four test problems by applying 
the basic CG method and the preconditioned CG method using the 
strategy of SSOR preconditioning.

7.3.1 Method of Solution

Consider the model problem
\[ \nabla^2 u = f(u) , \]  (7.3.1)

which can be replaced by finite difference approximations as 
outlined in Chapter 4 to yield the following equation,
\[ -\frac{1}{4} (u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}) + u_{ij} = -\frac{h^2}{4} f(u_{ij}), \]
for \( i,j = 1,2,\ldots,n \), (7.3.2)

which can be written in the compact form
\[ A \mathbf{u} = \mathbf{b} \quad (7.3.3) \]

where \( \mathbf{b} \) is the r.h.s. vector, and \( A \) can be represented by the computational five-point molecule.

\[
\begin{array}{ccc}
-\frac{1}{4} & i,j+1 \\

i-1,j & -\frac{1}{4} & i,j \\

i,j & -\frac{1}{4} & i+1,j \\

-\frac{1}{4} & i,j-1 \\
\end{array}
\]

Figure (7.3.1)

It is evident that a convenient splitting of the matrix \( A \) in (7.3.3) is given by

\[ A = I - L - L^T \quad , \]

with \( L \) having the form,

\[
\begin{bmatrix}
0 & & & & \\
-\frac{1}{4} & 0 & & & \\
-\frac{1}{4} & -\frac{1}{4} & 0 & & \\
-\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & 0 & \\
& & & & \end{bmatrix}
\]
Now, we recall the preconditioning matrix given by (7.2.17) which was derived from the SSOR method, i.e.

\[ H = \frac{1}{\omega(2-\omega)} (D - \omega L)D^{-1}(D - \omega L^T) \] \hspace{1cm} (7.3.4)

From equation (7.3.2) we observe that the diagonal matrix \( D \) has its diagonal elements unity, i.e. \( D = I \), so that the matrix \( H \) in (7.3.4) effectively reduces to

\[ H = \frac{1}{\omega(2-\omega)} (I - \omega L)(I - \omega L^T) \] \hspace{1cm} (7.3.5)

Subsequently,

\[ H^{-1} = \omega(2-\omega)(I - \omega L^T)^{-1}(I - \omega L)^{-1} \] \hspace{1cm} (7.3.6)

To implement the matrix-vector multiplication when \( H^{-1} \) is the involved matrix, we use the forward and backward substitution concept as given earlier in section 2.6.

Suppose we want to compute \( H^{-1}x \), i.e.

\[ \omega(2-\omega)(I - \omega L^T)^{-1}(I - \omega L)^{-1}x \] \hspace{1cm} (7.3.7)

we proceed by working through the following steps:

1. **Forward substitution**

\[ (I - \omega L)^{-1}x = y \]

or \( (I - \omega L)y = x \)

or \( y_{i,j} = x_{i,j} - \frac{\omega}{4}(y_{i-1,j} + y_{i,j-1}) \)

2. **Backward substitution**

\[ (I - \omega L^T)^{-1}y = u \]
or \((I - \omega L^T)u = y\)

or \(u_{i,j} = y_{i,j} - \frac{\omega}{4} (u_{i+1,j} + u_{i,j+1})\).

Finally, we multiply the vector \(u\) by scalar quantity \(\omega(2-\omega)\).

From the two steps above we have the following computational molecules.

We sum up that for any matrix-vector multiplication involving the matrix \(A\) we use directly the molecule in figure (7.3.1), and for the matrices \((I - \omega L)^{-1}\) and \((I - \omega L^T)^{-1}\) we use the molecules in figure (7.3.2).

7.3.2 Experimental Results

Since the preconditioning parameter \(\omega\) lies in the interval \((1,2)\) (see section 4.6 for optimal values) then to obtain the optimal values we use the usual technique of searching for the optimal value by solving the problem for different values of \(\omega\) and choosing the value which produces the lowest number of iterations. For the stopping criterion, we use the mixed...
convergence test with a tolerance of $\varepsilon = 5 \times 10^{-6}$. Tables (7.3.1) to (7.3.4) contain the results of applying the CG and the PCG methods to the test problems. It is clear that the number of iterations for the PCG method is less than those for the CG method and the difference is substantial for larger values of $h^{-1}$. We observe that the number of iterations for Problems 2 to 3 are close for the CG method and exactly the same for the PCG method.

The results in tables (7.3.1) to (7.3.4) are plotted in logarithmic-scale graphs as shown by figures (7.3.1) - (7.3.4) which illustrate that the slope of the regression lines for the CG methods are constant and approximately 1 while those for the PCG methods are approximately $\frac{1}{2}$. The slopes of the regression lines are given in the following table.

<table>
<thead>
<tr>
<th>Problem</th>
<th>CG</th>
<th>PCG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem 1</td>
<td>1.19</td>
<td>.47</td>
</tr>
<tr>
<td>Problem 2</td>
<td>1.1</td>
<td>.43</td>
</tr>
<tr>
<td>Problem 3</td>
<td>1.03</td>
<td>.5</td>
</tr>
<tr>
<td>Problem 4</td>
<td>1.1</td>
<td>.44</td>
</tr>
<tr>
<td>$h^{-1}$</td>
<td>CG method</td>
<td>PCG method</td>
</tr>
<tr>
<td>---------</td>
<td>-----------</td>
<td>------------</td>
</tr>
<tr>
<td></td>
<td>No. of iterations</td>
<td>Preconditioning Parameter</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>1.3</td>
</tr>
<tr>
<td>10</td>
<td>14</td>
<td>1.4</td>
</tr>
<tr>
<td>20</td>
<td>32</td>
<td>1.5</td>
</tr>
<tr>
<td>30</td>
<td>47</td>
<td>1.7</td>
</tr>
<tr>
<td>40</td>
<td>63</td>
<td>1.75</td>
</tr>
<tr>
<td>50</td>
<td>78</td>
<td>1.8</td>
</tr>
<tr>
<td>60</td>
<td>93</td>
<td>1.85</td>
</tr>
<tr>
<td>80</td>
<td>124</td>
<td>1.92</td>
</tr>
</tbody>
</table>

Table (7.3.1): $\nabla^2 u = u^2$, with initial guess 7.
<table>
<thead>
<tr>
<th>$h^{-1}$</th>
<th>CG method</th>
<th>PCG method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No. of iterations</td>
<td>Preconditioning Parameter</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>1.3</td>
</tr>
<tr>
<td>10</td>
<td>13</td>
<td>1.4</td>
</tr>
<tr>
<td>20</td>
<td>25</td>
<td>1.6</td>
</tr>
<tr>
<td>30</td>
<td>37</td>
<td>1.8</td>
</tr>
<tr>
<td>40</td>
<td>50</td>
<td>1.85</td>
</tr>
<tr>
<td>50</td>
<td>62</td>
<td>1.85</td>
</tr>
<tr>
<td>60</td>
<td>73</td>
<td>1.9</td>
</tr>
<tr>
<td>80</td>
<td>96</td>
<td>1.9</td>
</tr>
</tbody>
</table>

Table (7.3.2): \( \nabla^2 u = u^4 \), with initial guess 0.01.
<table>
<thead>
<tr>
<th>$h^{-1}$</th>
<th>CG method</th>
<th>PCG method</th>
<th>Preconditioning Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No. of iterations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>1.35</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>12</td>
<td>1.4</td>
<td>6</td>
</tr>
<tr>
<td>20</td>
<td>23</td>
<td>1.5</td>
<td>9</td>
</tr>
<tr>
<td>30</td>
<td>32</td>
<td>1.7</td>
<td>10</td>
</tr>
<tr>
<td>40</td>
<td>41</td>
<td>1.75</td>
<td>12</td>
</tr>
<tr>
<td>50</td>
<td>51</td>
<td>1.8</td>
<td>13</td>
</tr>
<tr>
<td>60</td>
<td>61</td>
<td>1.85</td>
<td>14</td>
</tr>
<tr>
<td>80</td>
<td>80</td>
<td>1.9</td>
<td>16</td>
</tr>
</tbody>
</table>

Table (7.3.3): $\nabla^2 u = e^u$, with initial guess $-0.01$. 
<table>
<thead>
<tr>
<th>Method</th>
<th>CG Method</th>
<th>PCG Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h^{-1}$</td>
<td>No. of Iterations</td>
<td>Preconditioning Parameter</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>1.3</td>
</tr>
<tr>
<td>10</td>
<td>13</td>
<td>1.4</td>
</tr>
<tr>
<td>20</td>
<td>25</td>
<td>1.7</td>
</tr>
<tr>
<td>30</td>
<td>37</td>
<td>1.8</td>
</tr>
<tr>
<td>40</td>
<td>50</td>
<td>1.82</td>
</tr>
<tr>
<td>50</td>
<td>62</td>
<td>1.85</td>
</tr>
<tr>
<td>60</td>
<td>73</td>
<td>1.85</td>
</tr>
<tr>
<td>80</td>
<td>96</td>
<td>1.9</td>
</tr>
</tbody>
</table>

Table (7.3.4): $\nabla^2 u = -\lambda^2 \sin u$, $\lambda = 1$, with initial guess 0.01.
Problem 1

Legend
- CG method
- PCG method

Problem 2

Legend
- CG method
- PCG method

Figure (7.3.1): $\nabla^2 u = u^2$

Figure (7.3.2): $\nabla^2 u = u^4$
Problem 3

Legend
- CG method
- PCG method

Figure (7.3.3): \( \nabla^2 u = e^u \)

Problem 4

Legend
- CG method
- PCG method

Figure (7.3.4): \( \nabla^2 u = -\lambda^2 \sin(u), \lambda = 1 \)
Computational Complexity

From the definition of the algorithms of the CG and the PCG methods, we notice that the r.h.s. is only evaluated once at the start of the procedures, so the r.h.s. unit will not be included in the computational complexity in each iteration.

The computational complexity per iteration is

1) For the CG method

\[ 6n^2 \text{ multiplications} + 9n^2 \text{ additions}. \]

2) For the PCG method

\[ 9n^2 \text{ multiplications} + 13n^2 \text{ additions}. \]

Tables (7.3.5) - (7.3.8) contain the computational complexity corresponding to the results given by tables (7.3.1) - (7.3.4) respectively. Clearly, we notice from the tables that the PCG method is much better than the CG method.
<table>
<thead>
<tr>
<th>h⁻¹</th>
<th>CG method</th>
<th></th>
<th>PCG method</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M</td>
<td>A</td>
<td>M</td>
<td>A</td>
</tr>
<tr>
<td>5</td>
<td>24 n²</td>
<td>36 n²</td>
<td>63 n²</td>
<td>91 n²</td>
</tr>
<tr>
<td>10</td>
<td>84 n²</td>
<td>126 n²</td>
<td>90 n²</td>
<td>130 n²</td>
</tr>
<tr>
<td>20</td>
<td>192 n²</td>
<td>288 n²</td>
<td>126 n²</td>
<td>182 n²</td>
</tr>
<tr>
<td>30</td>
<td>282 n²</td>
<td>423 n²</td>
<td>153 n²</td>
<td>221 n²</td>
</tr>
<tr>
<td>40</td>
<td>378 n²</td>
<td>567 n²</td>
<td>171 n²</td>
<td>247 n²</td>
</tr>
<tr>
<td>50</td>
<td>468 n²</td>
<td>702 n²</td>
<td>189 n²</td>
<td>273 n²</td>
</tr>
<tr>
<td>60</td>
<td>558 n²</td>
<td>837 n²</td>
<td>207 n²</td>
<td>298 n²</td>
</tr>
<tr>
<td>80</td>
<td>744 n²</td>
<td>1116 n²</td>
<td>225 n²</td>
<td>325 n²</td>
</tr>
</tbody>
</table>

Computational Complexity for $\nabla^2 u = u^2$ corresponding to table (7.3.1).

Table (7.3.5)
<table>
<thead>
<tr>
<th>$h^{-1}$</th>
<th>CG method</th>
<th></th>
<th>PCG method</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M</td>
<td>A</td>
<td>M</td>
<td>A</td>
</tr>
<tr>
<td>5</td>
<td>$24n^2$</td>
<td>$36n^2$</td>
<td>$45n^2$</td>
<td>$65n^2$</td>
</tr>
<tr>
<td>10</td>
<td>$78n^2$</td>
<td>$117n^2$</td>
<td>$54n^2$</td>
<td>$78n^2$</td>
</tr>
<tr>
<td>20</td>
<td>$150n^2$</td>
<td>$225n^2$</td>
<td>$81n^2$</td>
<td>$117n^2$</td>
</tr>
<tr>
<td>30</td>
<td>$222n^2$</td>
<td>$333n^2$</td>
<td>$90n^2$</td>
<td>$130n^2$</td>
</tr>
<tr>
<td>40</td>
<td>$300n^2$</td>
<td>$450n^2$</td>
<td>$108n^2$</td>
<td>$156n^2$</td>
</tr>
<tr>
<td>50</td>
<td>$372n^2$</td>
<td>$558n^2$</td>
<td>$117n^2$</td>
<td>$169n^2$</td>
</tr>
<tr>
<td>60</td>
<td>$438n^2$</td>
<td>$657n^2$</td>
<td>$126n^2$</td>
<td>$182n^2$</td>
</tr>
<tr>
<td>80</td>
<td>$576n^2$</td>
<td>$864n^2$</td>
<td>$144n^2$</td>
<td>$208n^2$</td>
</tr>
</tbody>
</table>

Computational Complexity for $\nabla^2 u = u^4$ corresponding to table (7.3.2).

Table (7.3.6)
<table>
<thead>
<tr>
<th>method $h^{-1}$</th>
<th>CG method</th>
<th></th>
<th>PCG method</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$M$</td>
<td>$A$</td>
<td>$M$</td>
<td>$A$</td>
</tr>
<tr>
<td>5</td>
<td>$24n^2$</td>
<td>$36n^2$</td>
<td>$36n^2$</td>
<td>$52n^2$</td>
</tr>
<tr>
<td>10</td>
<td>$72n^2$</td>
<td>$108n^2$</td>
<td>$54n^2$</td>
<td>$78n^2$</td>
</tr>
<tr>
<td>20</td>
<td>$138n^2$</td>
<td>$207n^2$</td>
<td>$81n^2$</td>
<td>$117n^2$</td>
</tr>
<tr>
<td>30</td>
<td>$192n^2$</td>
<td>$288n^2$</td>
<td>$90n^2$</td>
<td>$130n^2$</td>
</tr>
<tr>
<td>40</td>
<td>$246n^2$</td>
<td>$369n^2$</td>
<td>$108n^2$</td>
<td>$156n^2$</td>
</tr>
<tr>
<td>50</td>
<td>$306n^2$</td>
<td>$459n^2$</td>
<td>$117n^2$</td>
<td>$169n^2$</td>
</tr>
<tr>
<td>60</td>
<td>$366n^2$</td>
<td>$549n^2$</td>
<td>$126n^2$</td>
<td>$182n^2$</td>
</tr>
<tr>
<td>80</td>
<td>$480n^2$</td>
<td>$720n^2$</td>
<td>$144n^2$</td>
<td>$208n^2$</td>
</tr>
</tbody>
</table>

Computational Complexity for $V^2u = e^u$ corresponding to table (7.3.3).

Table (7.3.7)
<table>
<thead>
<tr>
<th>method</th>
<th>( h^{-1} )</th>
<th></th>
<th></th>
<th></th>
<th>CG method</th>
<th>( M )</th>
<th>( A )</th>
<th>PCG method</th>
<th>( M )</th>
<th>( A )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5</td>
<td>24</td>
<td>36</td>
<td></td>
<td>45</td>
<td>65</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>78</td>
<td>117</td>
<td></td>
<td>54</td>
<td>78</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>150</td>
<td>225</td>
<td></td>
<td>81</td>
<td>117</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>222</td>
<td>333</td>
<td></td>
<td>90</td>
<td>130</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>300</td>
<td>450</td>
<td></td>
<td>108</td>
<td>156</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>372</td>
<td>558</td>
<td></td>
<td>117</td>
<td>169</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>438</td>
<td>657</td>
<td></td>
<td>126</td>
<td>182</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>567</td>
<td>864</td>
<td></td>
<td>144</td>
<td>208</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Computational Complexity for \( \nabla^2 u = -\lambda^2 \sin u \), \( \lambda = 1 \) corresponding to table (7.3.4).

Table (7.3.8)
7.4 THE LU-DECOMPOSITION ALGORITHM FOR TWO DIMENSIONAL P.D.E.'s.

7.4.1 Introduction

Consider the system of equations which arise from approximating the elliptic p.d.e.'s by using finite-differences approximation on a rectangular grid on a 2-dimensional region, i.e.

\[ A \mathbf{u} = \mathbf{b} \quad , \tag{7.4.1} \]

where \( A \) is an \( n \times n \) symmetric, positive definite matrix, \( \mathbf{u} \) and \( \mathbf{b} \) are \( n \times 1 \) unknown and constant vectors respectively. The matrix \( A \) has the following form

\[
\begin{bmatrix}
    b_1 & c_1 \\
    a_2 & b_2 & c_2 \\
    a_3 & b_3 & c_3 \\
    & \ddots & \ddots & \ddots \\
    & & b_{m-2} & c_{m-2} \\
    & & & \ddots & \ddots & \ddots \\
    & & & & b_m & c_m \\
    & & & & & b_{m+1} & c_{m+1} \\
    & & & & & & b_n \\
    & & & & & & & & & & & & b_{n-1} & c_{n-1} \\
    & & & & & & & & & & & & & b_n \\
    & & & & & & & & & & & & & & & b_{n-1} & c_{n-1} \\
    & & & & & & & & & & & & & & & & b_n \\
    & & & & & & & & & & & & & & & & & & b_{n-1} & c_{n-1} \\
    & & & & & & & & & & & & & & & & & & & b_n \\
    & & & & & & & & & & & & & & & & & & & & b_{n-1} & c_{n-1} \\
\end{bmatrix}
\]

The matrix \( A \) can be factorized as

\[ A = L U \quad , \tag{7.4.2} \]

where \( L \) and \( U \) are lower and upper triangular matrices respectively, which have the forms:
\[
L = \begin{bmatrix}
\begin{array}{cccc}
\beta_1 & \omega_1 \\
\beta_2 & \omega_2 & & 0 \\
& \omega_3 & & \\
& & \ddots & \\
& & & 0 \\
\end{array}
\end{bmatrix}
= \begin{bmatrix}
L_{11} & 0 \\
L_{21} & L_{22}
\end{bmatrix}
\]

(7.4.4)
\[ U = \begin{bmatrix} 1 & (g_{m-1} + h_{m-1,1}) & h_{m-2,2} & h_{3,n-n+1} \\ 1 & (g_m + h_{m-1,2}) & h_{m-2,3} & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ 1 & (g_{n-1} + h_{m-1,n-m}) & \vdots & 1 \end{bmatrix} \]

\[ \equiv \begin{bmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{bmatrix} \]

(7.4.5)
As can be seen from the structure of the triangular matrices \( L \) and \( U \), the storage requirements and the computational work are prohibitive and consequently it will be considerably expensive to factorize \( A \) into the above matrices \( L \) and \( U \), also their entries \( (y_{i,j}, h_{i,j}) \) generated in the band are monotonically decreasing in magnitude and hence after the \( r \) outermost off-diagonal entries they can be replaced by zero which will lead to the approximate factorization technique. This idea was first introduced by Buleev (1960) and Oliphant (1962) and proposes an approximate factorization of \( A \) instead of the complete factorization, i.e.

\[
A = L_s U_s ,
\]

where \( L_s \) and \( U_s \) are sparse forms of the lower and upper triangular matrices respectively.
7.4.2 The Approximate LU-Decomposition Algorithm
(The ALUBOT Algorithm)

Here, we describe the approximate factorization of the coefficient matrix $A$ (given by (7.4.2), where $L_s$ and $U_s$ are easily determined and have only $r$ off-diagonal entries.

\[
L_s = \begin{bmatrix}
\tilde{\omega} \\
\tilde{\beta}_1 & \tilde{\omega}_2 \\
\tilde{\beta}_2 & \tilde{\omega}_3 & \ddots \\
\vdots & \ddots & \ddots & \ddots \\
\tilde{\gamma}_{1,1} & \tilde{\gamma}_{1,2} & \cdots & \tilde{\gamma}_{1,r} & \tilde{\beta}_{m-2} & \tilde{\omega}_{m-1} \\
\tilde{\gamma}_{2,1} & \tilde{\gamma}_{2,2} & \cdots & \tilde{\gamma}_{2,r} & \tilde{\beta}_{m-1} & \tilde{\omega}_m \\
\vdots & \vdots & \ddots & \vdots & \ddots & \ddots \\
\tilde{\gamma}_{r,1} & \cdots & \tilde{\gamma}_{r,r} & \tilde{\beta}_m & \tilde{\omega}_{m+1} \\
\end{bmatrix} \tag{7.4.7}
\]

\[
U_s = \begin{bmatrix}
1 & \tilde{g}_1 & \tilde{h}_{1,1} & & \ddots & \ddots & \ddots & \ddots & \ddots \\
1 & \tilde{g}_2 & \tilde{h}_{1,2} & \tilde{h}_{2,1} & \tilde{h}_{2,2} & & \ddots & \ddots & \ddots \\
1 & \tilde{g}_3 & \tilde{h}_{1,3} & \tilde{h}_{2,2} & \tilde{h}_{2,3} & \ddots & \ddots & \ddots & \ddots \\
\vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
1 & \tilde{g}_{m-2} & \tilde{h}_{m-2,1} & \tilde{h}_{m-2,2} & \tilde{h}_{m-2,3} & \ddots & \ddots & \ddots & \ddots \\
1 & \tilde{g}_{m-1} & \tilde{h}_{m-1,1} & \tilde{h}_{m-1,2} & \tilde{h}_{m-1,3} & \ddots & \ddots & \ddots & \ddots \\
1 & \tilde{g}_m & \tilde{h}_m & \tilde{h}_m & \tilde{h}_m & \ddots & \ddots & \ddots & \ddots \\
1 & \tilde{g}_{m+1} & \tilde{h}_{m+1} & \tilde{h}_{m+1} & \tilde{h}_{m+1} & \ddots & \ddots & \ddots & \ddots \\
1 & \tilde{g}_{n-1} & \tilde{h}_{n-1} & \tilde{h}_{n-1} & \tilde{h}_{n-1} & \ddots & \ddots & \ddots & \ddots \\
1 & & \tilde{g}_n & \tilde{h}_n & \tilde{h}_n & \ddots & \ddots & \ddots & \ddots \\
\end{bmatrix} \tag{7.4.8}
\]
To obtain the matrices $L_S$ and $U_S$ we follow the following algorithm (ALUBOT Algorithm, see Lipitakis, 1978) which retains the $r$ outermost off-diagonal entries

\[ \tilde{\omega}_1 = b_1; \quad \tilde{\beta}_1 = a_2; \quad \tilde{\gamma}_1 = c_1/\tilde{\omega}_1 \]  
(7.4.9)

for $i = 2, 3, \ldots, m-2$

\[ \tilde{\omega}_i = b_i - \tilde{\beta}_{i-1} \tilde{\gamma}_{i-1}; \quad \tilde{\beta}_i = a_{i+1}; \quad \tilde{\gamma}_i = c_i/\tilde{\omega}_i, \]  
(7.4.10)

and \[ \tilde{\omega}_{m-1} = b_{m-1} - \tilde{\beta}_{m-2} \tilde{\gamma}_{m-2} \]  
(7.4.11)

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

\[ \tilde{h}_{1,j} = e_j/\tilde{\omega}_j; \quad \tilde{\gamma}_{1,j} = d_{j+m-1} \]  
(7.4.12)

and \[ \tilde{\gamma}_{m+j-2} = c_{m+j-2}/\tilde{\omega}_{m+j-2}; \quad \tilde{\beta}_{m+j-2} = a_{m+j-1} \]  
(7.4.13)

for $j \leq r-1$

\[ \tilde{h}_{i,j} = - \tilde{\gamma}_{i+j-2} \tilde{h}_{i-1,j}/\tilde{\omega}_{i+j-1}; \quad \text{for } i = 2, 3, \ldots, r-j+1 \]  
(7.4.14)

and \[ \tilde{\gamma}_{i,j} = - \tilde{\gamma}_{i+j-2} \tilde{\gamma}_{i-1,j}; \quad \text{for } i = 2, 3, \ldots, r-j+1 \]  
(7.4.15)

for $j > 1$ and $r > 1$

\[ \tilde{\gamma}_{i,j} = - \tilde{\gamma}_{i+j-2} \tilde{\gamma}_{i-1,j} - \sum_{k=1}^{i-1} \tilde{\gamma}_{k,j} \tilde{h}_{k-i+r+1, i+j-r-1} \]  
(7.4.16)

and \[ \tilde{h}_{i,j} = \left(- \tilde{\beta}_{i+j-2} \tilde{h}_{i-1,j} - \sum_{k=1}^{i-1} \tilde{\gamma}_{k-i+r+1, i+j-r-1} \tilde{h}_{k,j}/\tilde{\omega}_{i+j-1} \right) \]  
(7.4.17)

for either $i = (r-j+2), (r-j+3), \ldots, r$ for all $j \leq r$

or $i = 2, 3, \ldots, r$ for $j > r$
for $i = r$

$$\tilde{\omega}_{m+j-1} = b_{m+j-1} - \tilde{\beta}_{i+j-1} \tilde{h}_{i,j} - g_{i+j-1} \tilde{y}_{i,j} - \tilde{\beta}_{i+j-1} g_{i+j-1} - \sum_{k=1}^{i} \tilde{y}_{k,j} \tilde{h}_{k,j},$$

(7.4.18)

Now, the system (7.4.1) can be written as

$$L_s U_s u = b,$$  

(7.4.19)

which can be solved by the forward-backward substitution process (explained earlier in section 2.6). Using the notation of the entries of $L_s$ and $U_s$, the forward substitution process by which we obtain the auxiliary vector $\tilde{y}$ is given by

$$\tilde{y}_1 = b_1 / \tilde{\omega}_1,$$

$$\tilde{y}_i = (b_i - \tilde{\beta}_{i-1} \tilde{y}_{i-1}) / \tilde{\omega}_i, \quad \text{for } i = 2, 3, \ldots, m-1 \quad (7.4.20)$$

and

$$\tilde{y}_i = (b_i - \tilde{\beta}_{i-1} \tilde{y}_{i-1} - \sum_{k=i-m+1}^{i-m+r} \tilde{y}_{k} / \tilde{\omega}_i, \quad \text{for } i = m, m+1, \ldots, n \quad (7.4.21)$$

and the backward substitution process which yields the solution vector $u$ in terms of the auxiliary vector $\tilde{y}$ is given by:

$$u_n = \tilde{y}_n,$$

$$u_i = \tilde{y}_i - g_i u_{i+1} - \sum_{j=p}^{q} \tilde{h}_{i-j+m,j-m+1} \tilde{y}_j, \quad \text{for } i = n-1, n-2, \ldots, 1 \quad (7.4.22)$$
The quantities p and q are determined as

if \( m > (\frac{n}{2} + 1) \)

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

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

then

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

else

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

N.B. Note the notation of the solution vector where we used one subscript for convenience. It can be easily transformed to two-subscript notation.
The ALUBOT algorithm can be used in conjunction with Newton's method to obtain what is called the Newton-ALUBOT composite iterative method where the outer iteration is carried out by Newton's method and the inner iteration by the ALUBOT algorithm. Suppose we have the following problem to solve:

\[ \nabla^2 u = f(u) \quad , \tag{7.5.1} \]

defined in a closed region \( R \) with boundary \( C \) and \( f(u) \) is a nonlinear function of \( u \).

Now, by recalling the theory of the Newton-SOR method (section 5.6), we can rewrite the problem (7.5.1) using the Newton linearization technique in the form

\[ L u^{(k+1)} - f'(u^{(k)}) u^{(k+1)} = f(u^{(k)}) - u^{(k)} f'(u^{(k)}), \tag{7.5.2} \]

where \( L u^{(k+1)} \) is the Laplacian linear operator.

The system (7.5.2) can be written in the compact form

\[ A u = b \quad . \tag{7.5.3} \]

Now, we divide the closed region \( R \) into a grid of mesh lines of size \( h \) with \( u_{ij} \) is defined at the mesh point \( i,j \) for \( i = 1,2,\ldots,n \).

The \( n^2 \times n^2 \) matrix \( A \) can be constructed from the computational molecule given in figure (7.5.1),
and the right hand side is given by

\[ b(u_{i,j}) = \frac{-h^2 [ f(u_{i,j}) - u_{i,j} f'(u_{i,j}) ]}{[4+h^2 f'(u_{i,j})]} \]  

(7.5.4)

Now, we factorize the matrix \( A \) by applying the ALUBOT algorithm, i.e. to obtain

\[ A = L_s U_s \]  

(7.5.5)

Then the solution is obtained by solving

\[ L_s U_s u_{(k+1)}^{(k+1)} = L_s U_s u^{(k)} + \omega (b(u^{(k)}) - \Lambda u^{(k)}) \]  

(7.5.6)

or

\[ L_s U_s (u_{(k+1)}^{(k+1)} - u^{(k)}) = \omega (b(u^{(k)}) - \Lambda u^{(k)}) \]  

(7.5.7)

where \( \omega \) is an acceleration parameter and the L.H.S. is obtained as before by using a forward-backward substitution process using the sparse forms \( L_s \) and \( U_s \).
7.6 ITERATIVE PROCEDURE AND EXPERIMENTAL RESULTS

By introducing a pre-determined acceleration parameter \( \omega \), we can obtain from the system (7.5.6) the following composite iterative scheme

\[
L_s U_s (u^{(k+1)} - u^{(k)}) = \omega \, r^{(k)},
\]

where

\[
r^{(k)} = b - A \, u^{(k)}
\]

and the inner iteration is carried out by the ALUBOT algorithm.

Since \( r \) (the number of off-diagonal entries) is variable, hence \( L_s \) and \( U_s \) are not the complete triangular factors of \( A \), Lipitakis (1978) updates \( L_s \) and \( U_s \) at each inner iteration until the convergence to specific tolerance has been achieved. Here we apply the ALUBOT algorithm only once in each outer iteration since the right solution is not obtained and hence no convergence test is necessary for the inner iteration. This has proved to provide faster convergence and saves much of computational work.

In addition to the Newton-ALUBOT method we apply the Picard-ALUBOT method in which the system

\[
\tilde{A} \, u^{(k+1)} = \tilde{b}(u^{(k)})
\]

is solved where \( \tilde{A} \) is obtained by the computational molecule given by figure (7.6.1):
Again the system (7.6.3) is solved by solving the iterative system given in (7.6.1) and (7.6.2). We use the mixed test of convergence with tolerance $5 \times 10^{-6}$ to terminate the outer iteration. The optimal acceleration parameter is determined experimentally using the procedure explained in earlier chapters.

It is noticeable that for the Picard-ALUBOT method, the coefficient matrix of the system (7.6.3) is factorized only once since it has a fixed form, while for the Newton-ALUBOT method, the coefficient matrix is factorized at each iteration since its diagonal entries are variables. It is possible to reduce this number by factorizing it only after a certain number of iterations.

The Picard-ALUBOT and the Newton-ALUBOT methods have been applied to the four test problems and the following tables (7.6.1) to (7.6.4) contain the results for various values of $h^{-1}$. We notice that the number of iterations of the Newton-ALUBOT method is slightly less than those of the Picard-ALUBOT method only for the first problem, and then are the same for the remaining problems. This proves that the Picard-ALUBOT method is superior to the Newton-ALUBOT method considering the difference in the computational work required by each method. The results are plotted in logarithmic-scale graphs depicted in figures (7.6.1) to (7.6.4). The slopes of the regression lines are given by the following table.

\[
\tilde{b}(u^{(k)}) = -\frac{h^2}{4} f(u^{(k)}) \quad (7.6.4)
\]
<table>
<thead>
<tr>
<th>Problem</th>
<th>Picard-ALUBOT</th>
<th>Newton-ALUBOT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem 1</td>
<td>0.78</td>
<td>0.75</td>
</tr>
<tr>
<td>Problem 2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Problem 3</td>
<td>0.93</td>
<td>0.84</td>
</tr>
<tr>
<td>Problem 4</td>
<td>1.04</td>
<td>0.93</td>
</tr>
</tbody>
</table>

**Computational Complexity:**

We consider only the number of multiplications since the number of additions are very small compared with the number of multiplications in the ALUBOT algorithm and the forward-backward substitution process.

The number of multiplications for the ALUBOT algorithm is \((r^2 + 3r + 2)n^2\) and for the forward-backward substitution process it is \((r^2 + 5r + 5)n^2\), where \(n^2\) is the number of mesh points in the grid (Evans and Lipitakis, 1979).

**The Picard-ALUBOT method**

Since in this case the ALUBOT algorithm is applied only once, then its number of multiplications will be added to the overall number of multiplications. First we have the number of multiplications for each mesh point, i.e.

\[(r^2 + 5r + 9) + 1 \text{ r.h.s. unit.}\]

So the total number of multiplications per each mesh point is

\[(\text{number of iterations}) \times (r^2 + 5r + 9) + (r^2 + 3r + 2).\]
The Newton–ALUBOT method

The total number of multiplications at each mesh point per iteration is

\[(2r^2 + 8r + 15) + 3 \text{ r.h.s. unit.}\]

Tables (7.6.5) to (7.6.8) contain the number of multiplications corresponding to the results given by tables (7.6.1) to (7.6.4). They are estimated with margin error as we have not included the r.h.s. unit.

Finally, the ALUBOT algorithm can be used as a preconditioner in exactly the same manner as shown earlier by the SSOR iteration matrix to provide a powerful preconditioned variational method.
| Method \( h^{-1} \) | Picard-ALUBOT | | Newton-ALUBOT | | |
|---|---|---|---|---|
| | Acceleration Parameter | Number of Iterations | Acceleration Parameter | Number of Iterations |
| 5 | .7 | 9 | 1 | 6 |
| 10 | .78 | 10 | .95 | 9 |
| 15 | .73 | 12 | .9 | 10 |
| 20 | .64 | 15 | .85 | 12 |
| 30 | .5 | 22 | .68 | 18 |
| 40 | .4 | 31 | .55 | 24 |
| 50 | .32 | 40 | .47 | 29 |
| 60 | .27 | 48 | .41 | 34 |
| 80 | .2 | 67 | .31 | 46 |

\( \psi^2 u = u^2 \) with initial guess 7 and \( r = 3 \).
<table>
<thead>
<tr>
<th>$h^{-1}$</th>
<th>Picard-ALUBOT</th>
<th>Newton-ALUBOT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Acceleration Parameter</td>
<td>Number of Iterations</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>.99</td>
<td>9</td>
</tr>
<tr>
<td>15</td>
<td>.98</td>
<td>10</td>
</tr>
<tr>
<td>20</td>
<td>.95</td>
<td>12</td>
</tr>
<tr>
<td>30</td>
<td>.82</td>
<td>15</td>
</tr>
<tr>
<td>40</td>
<td>.65</td>
<td>21</td>
</tr>
<tr>
<td>50</td>
<td>.54</td>
<td>27</td>
</tr>
<tr>
<td>60</td>
<td>.44</td>
<td>35</td>
</tr>
<tr>
<td>80</td>
<td>.333</td>
<td>50</td>
</tr>
</tbody>
</table>

Table (7.6.2): $\nu^2 u = u^4$ with initial guess 0.01 and $r=4$. 

<table>
<thead>
<tr>
<th>Method</th>
<th>Picard-ALUBOT</th>
<th></th>
<th>Newton-ALUBOT</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Acceleration Parameter</td>
<td>Number of Iterations</td>
<td>Acceleration Parameter</td>
<td>Number of Iterations</td>
</tr>
<tr>
<td>$h^{-1}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>10</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>15</td>
<td>1.1</td>
<td>11</td>
<td>.97</td>
<td>12</td>
</tr>
<tr>
<td>20</td>
<td>.95</td>
<td>13</td>
<td>.95</td>
<td>14</td>
</tr>
<tr>
<td>30</td>
<td>.78</td>
<td>17</td>
<td>.78</td>
<td>17</td>
</tr>
<tr>
<td>40</td>
<td>.63</td>
<td>24</td>
<td>.62</td>
<td>25</td>
</tr>
<tr>
<td>50</td>
<td>.52</td>
<td>31</td>
<td>.53</td>
<td>31</td>
</tr>
<tr>
<td>60</td>
<td>.44</td>
<td>39</td>
<td>.45</td>
<td>39</td>
</tr>
<tr>
<td>80</td>
<td>.33</td>
<td>56</td>
<td>.34</td>
<td>55</td>
</tr>
</tbody>
</table>

Table (7.6.3): $\nu^2 u = e^u$, with initial guess $-0.01$ and $r=4$. 
<table>
<thead>
<tr>
<th>Method</th>
<th>Picard-ALUBOT</th>
<th>Newton-ALUBOT</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\frac{h}{1}$</td>
<td>Acceleration Parameter</td>
<td>Number of Iterations</td>
</tr>
<tr>
<td>5</td>
<td>1.04</td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td>1.02</td>
<td>9</td>
</tr>
<tr>
<td>15</td>
<td>.95</td>
<td>11</td>
</tr>
<tr>
<td>20</td>
<td>.9</td>
<td>12</td>
</tr>
<tr>
<td>30</td>
<td>.8</td>
<td>16</td>
</tr>
<tr>
<td>40</td>
<td>.66</td>
<td>21</td>
</tr>
<tr>
<td>50</td>
<td>.54</td>
<td>28</td>
</tr>
<tr>
<td>60</td>
<td>.45</td>
<td>36</td>
</tr>
<tr>
<td>80</td>
<td>.33</td>
<td>52</td>
</tr>
</tbody>
</table>

Table (7.6.4): $\nabla^2 u = -\lambda^2 \sin u, \lambda = 1$, with initial guess 0.01 and $r = 4$. 
Problem 1

Legend
- Picard-ALUBOT
- Newton-ALUBOT

Figure (7.6.1) : $\nabla^2 u = u^2$

Problem 2

Legend
- Picard-ALUBOT
- Newton-ALUBOT

Figure (7.6.2) : $\nabla^2 u = u^4$
Problem 3

Legend
- Picard-ALUBOT
- Newton-ALUBOT

Figure (7.6.3): $\nabla^2 u = e^u$

Problem 4

Legend
- Picard-ALUBOT
- Newton-ALUBOT

Figure (7.6.4): $\nabla^2 u = -\lambda^2 \sin(u), \lambda=1$
<table>
<thead>
<tr>
<th>method \ h^{-1}</th>
<th>Picard-ALUBOT</th>
<th>Newton-ALUBOT</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>$317 , n^2$</td>
<td>$342 , n^2$</td>
</tr>
<tr>
<td>10</td>
<td>$350 , n^2$</td>
<td>$513 , n^2$</td>
</tr>
<tr>
<td>15</td>
<td>$416 , n^2$</td>
<td>$570 , n^2$</td>
</tr>
<tr>
<td>20</td>
<td>$515 , n^2$</td>
<td>$684 , n^2$</td>
</tr>
<tr>
<td>30</td>
<td>$746 , n^2$</td>
<td>$1026 , n^2$</td>
</tr>
<tr>
<td>40</td>
<td>$1043 , n^2$</td>
<td>$1368 , n^2$</td>
</tr>
<tr>
<td>50</td>
<td>$1340 , n^2$</td>
<td>$1653 , n^2$</td>
</tr>
<tr>
<td>60</td>
<td>$1604 , n^2$</td>
<td>$1938 , n^2$</td>
</tr>
<tr>
<td>80</td>
<td>$2231 , n^2$</td>
<td>$2622 , n^2$</td>
</tr>
</tbody>
</table>

Computational complexity for $V^2u = u^2$.

Table (7.6.5)

<table>
<thead>
<tr>
<th>method \ h^{-1}</th>
<th>Picard-ALUBOT</th>
<th>Newton-ALUBOT</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>$120 , n^2$</td>
<td>$158 , n^2$</td>
</tr>
<tr>
<td>10</td>
<td>$435 , n^2$</td>
<td>$711 , n^2$</td>
</tr>
<tr>
<td>15</td>
<td>$480 , n^2$</td>
<td>$790 , n^2$</td>
</tr>
<tr>
<td>20</td>
<td>$570 , n^2$</td>
<td>$948 , n^2$</td>
</tr>
<tr>
<td>30</td>
<td>$705 , n^2$</td>
<td>$1185 , n^2$</td>
</tr>
<tr>
<td>40</td>
<td>$975 , n^2$</td>
<td>$1659 , n^2$</td>
</tr>
<tr>
<td>50</td>
<td>$1245 , n^2$</td>
<td>$2212 , n^2$</td>
</tr>
<tr>
<td>60</td>
<td>$1605 , n^2$</td>
<td>$2765 , n^2$</td>
</tr>
<tr>
<td>80</td>
<td>$2280 , n^2$</td>
<td>$4029 , n^2$</td>
</tr>
</tbody>
</table>

Computational complexity for $V^2u = u^4$

Table (7.6.6)
<table>
<thead>
<tr>
<th>method $h^{-1}$</th>
<th>Picard-ALUBOT $M$</th>
<th>Newton-ALUBOT $M$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>$165 n^2$</td>
<td>$237 n^2$</td>
</tr>
<tr>
<td>10</td>
<td>$480 n^2$</td>
<td>$869 n^2$</td>
</tr>
<tr>
<td>15</td>
<td>$525 n^2$</td>
<td>$948 n^2$</td>
</tr>
<tr>
<td>20</td>
<td>$615 n^2$</td>
<td>$1106 n^2$</td>
</tr>
<tr>
<td>30</td>
<td>$795 n^2$</td>
<td>$1343 n^2$</td>
</tr>
<tr>
<td>40</td>
<td>$1110 n^2$</td>
<td>$1155 n^2$</td>
</tr>
<tr>
<td>50</td>
<td>$1425 n^2$</td>
<td>$2449 n^2$</td>
</tr>
<tr>
<td>60</td>
<td>$1785 n^2$</td>
<td>$3081 n^2$</td>
</tr>
<tr>
<td>80</td>
<td>$2550 n^2$</td>
<td>$4345 n^2$</td>
</tr>
</tbody>
</table>

Computational complexity for $\nabla^2 u = e^u$

Table (7.6.7)

<table>
<thead>
<tr>
<th>method $h^{-1}$</th>
<th>Picard-ALUBOT $M$</th>
<th>Newton-ALUBOT $M$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>$165 n^2$</td>
<td>$158 n^2$</td>
</tr>
<tr>
<td>10</td>
<td>$435 n^2$</td>
<td>$711 n^2$</td>
</tr>
<tr>
<td>15</td>
<td>$525 n^2$</td>
<td>$869 n^2$</td>
</tr>
<tr>
<td>20</td>
<td>$570 n^2$</td>
<td>$948 n^2$</td>
</tr>
<tr>
<td>30</td>
<td>$750 n^2$</td>
<td>$1264 n^2$</td>
</tr>
<tr>
<td>40</td>
<td>$975 n^2$</td>
<td>$1738 n^2$</td>
</tr>
<tr>
<td>50</td>
<td>$1290 n^2$</td>
<td>$2291 n^2$</td>
</tr>
<tr>
<td>60</td>
<td>$1650 n^2$</td>
<td>$2844 n^2$</td>
</tr>
<tr>
<td>80</td>
<td>$2370 n^2$</td>
<td>$4187 n^2$</td>
</tr>
</tbody>
</table>

Computational complexity for $\nabla^2 u = -\lambda^2 \sin u$, $\lambda = 1$.  

Table (7.6.8)
Chapter Eight

CONCLUSIONS AND FINAL COMMENTS
Many iterative methods have been formulated to solve large scale linear systems of equations, especially those sparse systems resulting from approximating elliptic p.d.e.'s by finite-difference approximations over a rectangular grid. In this thesis we dealt with classes of iterative methods namely: the basic iterative methods including the GS, SOR methods etc., the accelerated SOR methods, i.e. ESOR, AOR and Two-parameter methods, the Alternating direction implicit methods in particular the Peaceman-Rachford method and the Douglas-Rachford method and some variational methods including the conjugate gradient method, the preconditioned conjugate gradient method and finally an approximate LU-decomposition algorithm as an alternative preconditioning strategy.

In Chapters 4 to 7 we have applied the methods mentioned above to the following four mildly nonlinear test problems which are defined on a unit square region $\mathbb{R}$ with closed boundary $\mathbb{C}$,

1) $\nabla^2 u = u^2$
   with $u = 10$ on $\mathbb{C}$ and initial guess 7

2) $\nabla^2 u = u^4$
   with $u = 0$ on $\mathbb{C}$ and initial guess 0.01

3) $\nabla^2 u = e^u$
   with $u = 0$ on $\mathbb{C}$ and initial guess -0.01

4) $\nabla^2 u = -\lambda^2 \sin u$, $\lambda = 1$
   with $u = 0$ on $\mathbb{C}$ and initial guess 0.01

In each case we have estimated the computational complexity
in terms of multiplication and addition operations of each method. Here we compare the efficiency of the various methods in terms of number of multiplications required by each method for a specific mesh size. The following tables (8.1) to (8.4) contain the numbers of multiplications of all the methods for the four problems. These tables are represented by the graphs in figures (8.1) to (8.4). From the graphs we can discern that Problem 1 appears to be much harder than Problems 2-4 and we can conclude with the following remarks:

1) For Problem 1: the PCG, SOR-Newton and the Newton-SOR methods display a good performance index.

   In addition the multi-parameter ADI methods present good results (mainly because of suitability of the given region). The model problem is easily solved on this class of methods too. The results for the GS method shows a large variation since it is better than some methods for small sized problems but worse for larger problems. Also it is clearly seen that the Picard-ALUBOT and the Newton-ALUBOT algorithms are too costly to consider for this class of mildly nonlinear problems where the coefficient matrix is sparse and symmetric.

2) For Problems 2-4: the class of overrelaxation methods involve little work due to the sparseness of the coefficient matrix and the rate of convergence achieved is powerful enough to cope with this class of problems giving a good performance index. Also, the multi-parameter ADI methods display good results. The GS method gives poor results because of its slow convergence rate.
Also, it can be noticed that the Picard-ALUBOT and Newton-ALUBOT methods involve a great deal of computational work and are essentially too expensive for this class of problems as confirmed by the computational complexity results in tables (8.2) - (8.4). Also it can be noticed that the $1$ parameter ADI results are poor for this class of problems because of the amount of work involved for a convergence rate comparable to the SOR method.

Generally, the PCG method gave the best results and is to be recommended while the Picard-ALUBOT and the Newton-ALUBOT gave the worst computational complexity results though for smaller sized problems ($h^{-1} = 5 - 10$) some gains in efficiency were obtained by choosing a smaller fill in parameter, i.e. $r = 1$. However, we would expect these methods to improve for more severe nonlinear problems.
<table>
<thead>
<tr>
<th>$h^{-1}$-method</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>80</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS</td>
<td>290</td>
<td>521</td>
<td>560</td>
<td>751</td>
<td>986</td>
<td>706</td>
<td>1835</td>
</tr>
<tr>
<td>SOR</td>
<td>362</td>
<td>568</td>
<td>724</td>
<td>850</td>
<td>1028</td>
<td>1035</td>
<td>1164</td>
</tr>
<tr>
<td>SSOR</td>
<td>508</td>
<td>724</td>
<td>924</td>
<td>1072</td>
<td>1184</td>
<td>1324</td>
<td>1492</td>
</tr>
<tr>
<td>SSOR &amp; Cheby.</td>
<td>1010</td>
<td>1290</td>
<td>1440</td>
<td>1680</td>
<td>1790</td>
<td>1990</td>
<td>2340</td>
</tr>
<tr>
<td>Newton-SOR</td>
<td>72</td>
<td>136</td>
<td>198</td>
<td>258</td>
<td>306</td>
<td>374</td>
<td>468</td>
</tr>
<tr>
<td>SOR-Newton</td>
<td>44</td>
<td>88</td>
<td>130</td>
<td>172</td>
<td>212</td>
<td>252</td>
<td>336</td>
</tr>
<tr>
<td>ESOR</td>
<td>564</td>
<td>852</td>
<td>1080</td>
<td>1275</td>
<td>1440</td>
<td>1587</td>
<td>1815</td>
</tr>
<tr>
<td>AOR</td>
<td>543</td>
<td>846</td>
<td>1206</td>
<td>1263</td>
<td>1518</td>
<td>1545</td>
<td>1830</td>
</tr>
<tr>
<td>Two-Param.</td>
<td>576</td>
<td>942</td>
<td>1206</td>
<td>1401</td>
<td>1554</td>
<td>1692</td>
<td>1989</td>
</tr>
<tr>
<td>P-R Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$m=1$</td>
<td>182</td>
<td>322</td>
<td>462</td>
<td>588</td>
<td>728</td>
<td>854</td>
<td>1092</td>
</tr>
<tr>
<td>$m=2$</td>
<td>112</td>
<td>154</td>
<td>182</td>
<td>224</td>
<td>252</td>
<td>308</td>
<td>366</td>
</tr>
<tr>
<td>$m=3$</td>
<td>112</td>
<td>132</td>
<td>168</td>
<td>168</td>
<td>180</td>
<td>216</td>
<td>252</td>
</tr>
<tr>
<td>D-R Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$m=1$</td>
<td>247</td>
<td>416</td>
<td>572</td>
<td>715</td>
<td>858</td>
<td>1001</td>
<td>1261</td>
</tr>
<tr>
<td>$m=2$</td>
<td>234</td>
<td>299</td>
<td>325</td>
<td>364</td>
<td>390</td>
<td>416</td>
<td>455</td>
</tr>
<tr>
<td>$m=3$</td>
<td>208</td>
<td>273</td>
<td>325</td>
<td>351</td>
<td>351</td>
<td>377</td>
<td>377</td>
</tr>
<tr>
<td>CG</td>
<td>84</td>
<td>192</td>
<td>282</td>
<td>378</td>
<td>468</td>
<td>558</td>
<td>744</td>
</tr>
<tr>
<td>PCG</td>
<td>90</td>
<td>126</td>
<td>153</td>
<td>171</td>
<td>189</td>
<td>207</td>
<td>225</td>
</tr>
<tr>
<td>Picard-ALUBOT</td>
<td>350</td>
<td>515</td>
<td>746</td>
<td>1043</td>
<td>1340</td>
<td>1604</td>
<td>2231</td>
</tr>
<tr>
<td>Newton-ALUBOT</td>
<td>513</td>
<td>684</td>
<td>1026</td>
<td>1368</td>
<td>1653</td>
<td>1938</td>
<td>2622</td>
</tr>
</tbody>
</table>

Number of multiplications (times $n^2$) for $V^2u = u^2$.

Table (8.1)
<table>
<thead>
<tr>
<th>Method</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>80</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS</td>
<td>59</td>
<td>178</td>
<td>327</td>
<td>487</td>
<td>647</td>
<td>798</td>
<td>1041</td>
</tr>
<tr>
<td>SOR</td>
<td>40</td>
<td>80</td>
<td>120</td>
<td>160</td>
<td>200</td>
<td>240</td>
<td>310</td>
</tr>
<tr>
<td>SSOR</td>
<td>68</td>
<td>116</td>
<td>164</td>
<td>208</td>
<td>248</td>
<td>284</td>
<td>356</td>
</tr>
<tr>
<td>SSOR &amp; Cheby.</td>
<td>100</td>
<td>150</td>
<td>200</td>
<td>240</td>
<td>230</td>
<td>230</td>
<td>340</td>
</tr>
<tr>
<td>Newton-SOR</td>
<td>40</td>
<td>80</td>
<td>120</td>
<td>160</td>
<td>170</td>
<td>200</td>
<td>258</td>
</tr>
<tr>
<td>SOR-Newton</td>
<td>40</td>
<td>80</td>
<td>120</td>
<td>160</td>
<td>200</td>
<td>242</td>
<td>320</td>
</tr>
<tr>
<td>ESOR</td>
<td>54</td>
<td>114</td>
<td>168</td>
<td>213</td>
<td>261</td>
<td>309</td>
<td>423</td>
</tr>
<tr>
<td>AOR</td>
<td>57</td>
<td>117</td>
<td>168</td>
<td>240</td>
<td>294</td>
<td>354</td>
<td>405</td>
</tr>
<tr>
<td>Two-Param.</td>
<td>57</td>
<td>123</td>
<td>177</td>
<td>240</td>
<td>297</td>
<td>357</td>
<td>462</td>
</tr>
<tr>
<td>P-R m=1</td>
<td>140</td>
<td>238</td>
<td>336</td>
<td>420</td>
<td>532</td>
<td>616</td>
<td>756</td>
</tr>
<tr>
<td>P-R m=2</td>
<td>112</td>
<td>140</td>
<td>168</td>
<td>196</td>
<td>224</td>
<td>224</td>
<td>252</td>
</tr>
<tr>
<td>P-R m=3</td>
<td>56</td>
<td>126</td>
<td>140</td>
<td>140</td>
<td>168</td>
<td>168</td>
<td>168</td>
</tr>
<tr>
<td>D-R m=1</td>
<td>234</td>
<td>351</td>
<td>468</td>
<td>572</td>
<td>663</td>
<td>754</td>
<td>923</td>
</tr>
<tr>
<td>D-R m=2</td>
<td>208</td>
<td>234</td>
<td>260</td>
<td>286</td>
<td>286</td>
<td>286</td>
<td>286</td>
</tr>
<tr>
<td>D-R m=3</td>
<td>234</td>
<td>273</td>
<td>312</td>
<td>213</td>
<td>351</td>
<td>351</td>
<td>351</td>
</tr>
<tr>
<td>CG</td>
<td>78</td>
<td>150</td>
<td>222</td>
<td>300</td>
<td>372</td>
<td>438</td>
<td>576</td>
</tr>
<tr>
<td>FCG</td>
<td>54</td>
<td>81</td>
<td>90</td>
<td>108</td>
<td>117</td>
<td>126</td>
<td>144</td>
</tr>
<tr>
<td>Picard-ALUBOT</td>
<td>435</td>
<td>570</td>
<td>705</td>
<td>975</td>
<td>1245</td>
<td>1605</td>
<td>2280</td>
</tr>
<tr>
<td>Newton-ALUBOT</td>
<td>711</td>
<td>948</td>
<td>1185</td>
<td>1659</td>
<td>2212</td>
<td>2765</td>
<td>4029</td>
</tr>
</tbody>
</table>

Number of multiplications (times n²) for \( v^2u = u^4 \).  
Table (8.2)
<table>
<thead>
<tr>
<th>Method</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>80</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS</td>
<td>67</td>
<td>223</td>
<td>432</td>
<td>679</td>
<td>953</td>
<td>1244</td>
<td>1856</td>
</tr>
<tr>
<td>SOR</td>
<td>50</td>
<td>86</td>
<td>120</td>
<td>160</td>
<td>202</td>
<td>242</td>
<td>320</td>
</tr>
<tr>
<td>SSOR</td>
<td>68</td>
<td>124</td>
<td>196</td>
<td>256</td>
<td>316</td>
<td>368</td>
<td>476</td>
</tr>
<tr>
<td>SSOR &amp; Cheby.</td>
<td>120</td>
<td>160</td>
<td>200</td>
<td>210</td>
<td>280</td>
<td>320</td>
<td>340</td>
</tr>
<tr>
<td>Newton-SOR</td>
<td>42</td>
<td>78</td>
<td>116</td>
<td>152</td>
<td>190</td>
<td>228</td>
<td>288</td>
</tr>
<tr>
<td>SOR-Newton</td>
<td>40</td>
<td>80</td>
<td>120</td>
<td>160</td>
<td>202</td>
<td>242</td>
<td>320</td>
</tr>
<tr>
<td>ESOR</td>
<td>75</td>
<td>132</td>
<td>156</td>
<td>216</td>
<td>270</td>
<td>324</td>
<td>408</td>
</tr>
<tr>
<td>AOR</td>
<td>75</td>
<td>117</td>
<td>156</td>
<td>237</td>
<td>279</td>
<td>327</td>
<td>441</td>
</tr>
<tr>
<td>Two-Param.</td>
<td>81</td>
<td>129</td>
<td>171</td>
<td>252</td>
<td>300</td>
<td>375</td>
<td>495</td>
</tr>
<tr>
<td>P-R Method m=1</td>
<td>126</td>
<td>224</td>
<td>336</td>
<td>420</td>
<td>504</td>
<td>588</td>
<td>756</td>
</tr>
<tr>
<td></td>
<td>84</td>
<td>126</td>
<td>140</td>
<td>168</td>
<td>196</td>
<td>252</td>
<td>308</td>
</tr>
<tr>
<td></td>
<td>70</td>
<td>98</td>
<td>126</td>
<td>126</td>
<td>126</td>
<td>168</td>
<td>168</td>
</tr>
<tr>
<td>D-R Method m=1</td>
<td>234</td>
<td>377</td>
<td>507</td>
<td>598</td>
<td>702</td>
<td>806</td>
<td>988</td>
</tr>
<tr>
<td></td>
<td>234</td>
<td>260</td>
<td>260</td>
<td>312</td>
<td>312</td>
<td>364</td>
<td>338</td>
</tr>
<tr>
<td></td>
<td>216</td>
<td>252</td>
<td>252</td>
<td>252</td>
<td>252</td>
<td>252</td>
<td>312</td>
</tr>
<tr>
<td>CG</td>
<td>72</td>
<td>138</td>
<td>192</td>
<td>246</td>
<td>306</td>
<td>366</td>
<td>480</td>
</tr>
<tr>
<td>PCG</td>
<td>54</td>
<td>81</td>
<td>90</td>
<td>108</td>
<td>117</td>
<td>126</td>
<td>144</td>
</tr>
<tr>
<td>Picard-ALUBOT</td>
<td>480</td>
<td>615</td>
<td>795</td>
<td>1110</td>
<td>1425</td>
<td>1785</td>
<td>2550</td>
</tr>
<tr>
<td>Newton-ALUBOT</td>
<td>869</td>
<td>1106</td>
<td>1343</td>
<td>1155</td>
<td>2449</td>
<td>3081</td>
<td>4345</td>
</tr>
</tbody>
</table>

Number of multiplications (times $n^2$) for $\nabla^2 u = e^u$.

Table (8.3)
<table>
<thead>
<tr>
<th>Method</th>
<th>h⁻¹ = 10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>80</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS</td>
<td>62</td>
<td>186</td>
<td>339</td>
<td>504</td>
<td>668</td>
<td>821</td>
<td>1062</td>
</tr>
<tr>
<td>SOR</td>
<td>40</td>
<td>82</td>
<td>122</td>
<td>162</td>
<td>200</td>
<td>240</td>
<td>320</td>
</tr>
<tr>
<td>SSOR</td>
<td>72</td>
<td>124</td>
<td>172</td>
<td>216</td>
<td>256</td>
<td>296</td>
<td>372</td>
</tr>
<tr>
<td>SSOR &amp; Cheby.</td>
<td>110</td>
<td>160</td>
<td>170</td>
<td>210</td>
<td>240</td>
<td>290</td>
<td>330</td>
</tr>
<tr>
<td>Newton-SOR</td>
<td>40</td>
<td>82</td>
<td>120</td>
<td>162</td>
<td>176</td>
<td>208</td>
<td>262</td>
</tr>
<tr>
<td>SOR-Newton</td>
<td>40</td>
<td>82</td>
<td>120</td>
<td>162</td>
<td>200</td>
<td>240</td>
<td>320</td>
</tr>
<tr>
<td>ESOR</td>
<td>63</td>
<td>111</td>
<td>174</td>
<td>213</td>
<td>267</td>
<td>315</td>
<td>441</td>
</tr>
<tr>
<td>AOR</td>
<td>60</td>
<td>120</td>
<td>180</td>
<td>234</td>
<td>264</td>
<td>354</td>
<td>429</td>
</tr>
<tr>
<td>Two-Param.</td>
<td>57</td>
<td>117</td>
<td>180</td>
<td>243</td>
<td>300</td>
<td>363</td>
<td>471</td>
</tr>
<tr>
<td>P-R Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m=1</td>
<td>140</td>
<td>252</td>
<td>336</td>
<td>448</td>
<td>546</td>
<td>644</td>
<td>784</td>
</tr>
<tr>
<td>m=2</td>
<td>98</td>
<td>140</td>
<td>140</td>
<td>168</td>
<td>196</td>
<td>224</td>
<td>252</td>
</tr>
<tr>
<td>m=3</td>
<td>84</td>
<td>126</td>
<td>126</td>
<td>168</td>
<td>168</td>
<td>168</td>
<td>168</td>
</tr>
<tr>
<td>D-R Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m=1</td>
<td>234</td>
<td>364</td>
<td>468</td>
<td>572</td>
<td>676</td>
<td>780</td>
<td>936</td>
</tr>
<tr>
<td>m=2</td>
<td>234</td>
<td>234</td>
<td>286</td>
<td>286</td>
<td>386</td>
<td>312</td>
<td>312</td>
</tr>
<tr>
<td>m=3</td>
<td>221</td>
<td>273</td>
<td>312</td>
<td>351</td>
<td>351</td>
<td>273</td>
<td></td>
</tr>
<tr>
<td>CG</td>
<td>78</td>
<td>150</td>
<td>222</td>
<td>300</td>
<td>372</td>
<td>438</td>
<td>576</td>
</tr>
<tr>
<td>PCG</td>
<td>54</td>
<td>81</td>
<td>90</td>
<td>108</td>
<td>117</td>
<td>126</td>
<td>144</td>
</tr>
<tr>
<td>Picard-ALUBOT</td>
<td>435</td>
<td>570</td>
<td>750</td>
<td>975</td>
<td>1290</td>
<td>1650</td>
<td>2370</td>
</tr>
<tr>
<td>Newton-ALUBOT</td>
<td>711</td>
<td>948</td>
<td>1264</td>
<td>1738</td>
<td>2291</td>
<td>2844</td>
<td>4187</td>
</tr>
</tbody>
</table>

Number of multiplications (times n²) for \( \nabla^2 = -\lambda^2 \sin u, \lambda = 1 \).  

*Table (8.4)*
Problem 1

Figure (8.1): $v^2 u = u^2$
Problem 2

Legend

- GS
- SOR
- SSOR
- SSOR&Cheby
- Newton-SOR
- SOR-Newton
- ESOR
- AOR
- Two-Param
- P-R:m=1
- P-R:m=2
- P-R:m=3
- D-R:m=1
- D-R:m=2
- D-R:m=3
- CG
- PCG
- Picard-ALUBOT
- Newton-ALUBOT

Figure (8.2): $v^2 u = u^4$
Problem 3

Legend

- GS
- SOR
- SSOR
- SSOR&Cheby
- Newton-SOR
- SOR-Newton
- ESOR
- AOR
- Two-Param
  - P-R:m=1
  - P-R:m=2
  - P-R:m=3
  - D-R:m=1
  - D-R:m=2
  - D-R:m=3
- CG
- PCG
- Picard-ALUBOT
- Newton-ALUBOT

Figure (8.3): $v^2 u = \delta^u$
Problem 4

Figure (8.4): $\nabla^2 u = -\lambda^2 \sin u, \lambda = 1$
REFERENCES


GOTTFRIED, B.S. and WEISMAN, J. (1973), Introduction to Optimization Theory, Prentice-Hall.


† See Evans and Lipitakis (1979) at the end of the list.


VARGA, R.S. (1962), Matrix Iterative Analysis, Prentice-Hall, Inc.


This program is to solve the mildly non-linear problems (Lu=u**2) by the SSOR method accelerated by the Chebyshev acceleration.

The file dd contains the data (input) and the file res contains the results (output).

f, f0, f1, f2 are 2-dimensional arrays to carry out the iteration.

g, error: arrays for the convergence tests.

hbb: the r.h.s. array.

nx: number of grid points on the x-axis.

ny: number of grid points on the y-axis.

om: the acceleration parameter.

h: the mesh-size.

open(6, file='dd', form='formatted', status='old')
open(5, file='res', form='formatted', status='new')
dimension f(41, 41), f0(41, 41), g(41, 41), hbb(41, 41)
dimension error(41, 41), f1(41, 41), f2(41, 41)

read(6,*) nx, ny, om
h=1./nx
eps=1e-3
epss=5e-6
write(5, 110)
110 format(100(Ih*,1I20x.1R))

METHOD WITH CHEBYSHEV

write(5, 130) nx, ny, h, om, eps, epss
130 format(' NUMBER OF X INTERVALS=', i3, 
' NUMBER OF Y INTERVALS=', i3, 
' INTERVAL LENGTH=', f6.4, 
' OMEGA=', f6.4, 
' INNER LOOP TOLERANCE=', f9.7, 
' OUTER LOOP TOLERANCE=', f9.7, 
' INITIAL GUESS U(I,J)=7')

hh=h*h
om1=om-1
om25=om*.25

dimension of the parameters required to calculate.

em=0
pi=3.142847
esin=sin(pi*h*0.5)
emm=(1-esin)/(1+esin)
gama=2./(2.-emm-em)
gama1=1-gama
sigma=(emm-em)/(2.-emm-em)
sigma2=sigma*sigma

Setting the boundary conditions for the given problem.

do 1 i=1, nx+1
   f(i,1)=10
1 f(i,ny+1)=10
do 2 j=1, ny+1
   f(1,j)=10
2 f(nx+1,j)=10

The initial starting solution.

do 3 i=2, nx
   do 3 j=2, ny
   f(i,j)=7
3 itn=0
180 itn=0
Calculating the r.h.s.

```fortran
180  do 4 i=2,nx
    do 4 j=2,ny
      g(i,j)=f(i,j)
4   hhhb(i,j)= hh*f(i,j)**2
    itnn=itnn+1
    omega=1
190  itn=itn+1
    do 5 i=2,nx
      do 5 j=2,ny
        f1(i,j)=f(i,j)
        f'(i,j)=om25*(f(i-1,j)+f(i+1,j)+f(i,j-1)+f(i,j+1))
* -hhb(i,j))=oom1*f(i,j)
5  continue
    do 6 i=nx,2,-1
      do 6 j=ny,2,-1
        f(i,j)=0
        f0(i,j)=f1(i,j)
    endif
6 continue
    if(itn.eq.1) goto 200
    if(itn.eq.2) then
      omega=1/((1.-.5*sigma2)
    else
      omega=1/(1.-.25*sigma2* omega)
    endif
190  omega1=1-omega
    do 7 i=2,nx
      do 7 j=2,ny
        if(abs(error(i,j))>(1.+abs(f(i,j))).gt.eps) goto 190
      endif
7 error(i,j)=f2(i,j)-f1(i,j)
    do 8 i=2,nx
      do 8 j=2,ny
        f(i,j)=f2(i,j)
8  continue
  c .... The inner loop test of convergence.
    do 9 i=2,nx
      do 9 j=2,ny
        if(abs(error(i,j))/(1.+abs(f(i,j))).gt.eps) goto 190
    endif
9 continue
  c .... The outer loop test of convergence.
    do 10 i=2,nx
      do 10 j=2,ny
10 if(abs(g(i,j)-f(i,j))/(1.+abs(f(i,j))).gt.eps) goto 180
 write(5,150)
150 format( ' THE SOLUTION IS AS FOLLOWING, IN ORDER'/ )
  c .... Printing the final solution.
    do 11 i=nx+1,1,-1
      write(5,160)(f(i,j),j=1,ny+1)
11 format(22e14.6)
    write(5,170)itn, itnn
160 format(//3x,'Total Number of Inner Iterations=',i4/3x,'Total Number of Outer Iterations=',i4/)
 stop
end
```
This program is to solve the mildly nonlinear problem
\( (L_u = \exp(u)) \) by the Peaceman-Rachford method with

---

cycles of 3 parameters in turns.

The file dd contains the data (input) and the file
res contains the results (output).

---

f, ff, f1 are 2-dimensional arrays to carry out the
iteration.

a, b, c are the entries of the tridiagonal matrix.

r; the r.h.s. elements.

t; the solution vector.

nx : number of grid points on the x-axis.

ny : number of grid points on the y-axis.

beta1, beta2, beta3 : the acceleration parameters.

h : mesh-size.

---

subroutine tridiag : to solve the tridiagonal system.

---

Set up the initial guess.

---

Set the boundary conditions.

---

Set up the initial guess.
fl(1,j)=0
fl(nxpl,j)=0
ff(1,j)=0
ff(nxpl,j)=0

c .... mitn : counter for the cycle of parameters.
c .... itn : counter for the number of iterations.
mitn=0
itn=0
c .... Test if no convergence obtained until specific number
c .... of iterations, then exit.
if(itn.eq.100) then
  write(6,100)
goto 98
end if
mitn=mitn+1
itn=itn+1
if(mitn.eq.1) then
  betat=beta1-2.
else if(mitn.eq.2) then
  betat=beta2-2.
else
  betat=beta3-2.
 mitn=0
end if
do 500 jj=1,nxpl
b(jj)=betat+4.
500
if(b(jj).lt.epss) b(jj)=epss
do 30 i=1,nxpl
do 30 j=1,ny+1
f(i,j)=fl(i,j)
do 40 j=2,ny
r(1)=f(2,j-1)-hh*exp(f(2,j))+betat*f(2,j)
c  +f(2,j+1)+f(1,j)
do 35 i=2,nxm2
35  r(i)=f(i+1,j-1)-hh*exp(f(i+1,j))+betat*f(i+1,j)
  *+f(i+1,j+1)
r(nxm1)=f(nx,j-1)-hh*exp(f(nx,j))+betat*f(nx,j)
  *+f(nx,j+1)+f(nxpl,j)
call tridiag(nxm1,a,b,c,t,r)
do 40 i=1,nxm1
40  ff(i+1,j)=t(i)
c .... y sweep. mirror image of the x sweep.
do 50 i=2,nx
r(i)=ff(i-1,2)-hh*exp(ff(i,2))+betat*ff(i,2)
  *+ff(i+1,2)+ff(i,1)
do 45 j=2,ny-2
45  r(j)=ff(i-1,j+1)-hh*exp(ff(i,j+1))+betat*ff(i,j+1)
  *+ff(i+1,j+1)
r(ny-1)=ff(i-1,ny)-hh*exp(ff(i,ny))+betat*ff(i,ny)
  *+ff(i+1,ny)+ff(i,ny+1)
call tridiag(ny-1,a,b,c,t,r)
do 50 j=1,ny-1
50  fl(i,j+1)=t(i)
c .... Test of convergence.
do 52 i=2,nx
do 52 j=2,ny
  if(abs(fl(i,j)-f(i,j))/(1.0+abs(f(i,j))).gt.epss) then
    goto 26
  end if

c .... NOW PRINT THE OUTPUT.
write(6,99)itn
do 55 j=ny+1,1,-1
write(6,180)(fl(i,j),i=1,nx+1)
write(6,181)
format(4x,'NO. OF ITERATIONS=',i4//)
format(2x,44(e14.6))
format(//88(1h*))
goto 111
end

--- Subroutine to solve tridiagonal system of linear equations.---

subroutine tridiag(n,a,b,c,x,g)
dimension a(81),b(81),c(81),x(81),g(81),bb(81)
do 1 i=1,n
  bb(i)=b(i)
do 2 i=2,n
t=a(i)/bb(i-1)
  bb(i)=bb(i)-c(i-1)*t
2  g(i)=g(i)-g(i-1)*t
  c---- BACK SUBSTITUTION.
x(n)=g(n)/bb(n)
do 3 i=1,n-1
  j=n-i
3  x(j)=(g(j)-c(j)*x(j+1))/bb(j)
return
end
This program is to solve the mildly non-linear problem \((Lu=-\sin (u))\) by the Preconditioned Conjugate Gradients method using the SSOR iteration matrix as the preconditioning matrix.

The file \textit{dd} contains the data (input) and the file \textit{res} contains the results (output).

\text{f,ff,p,ap,rmr2} :are 2-dimensional arrays used in the iteration.

\text{r} :the residual array.

\text{b} :the r.h.s. array.

\text{nx} :number of grid points on the x-axis.

\text{ny} :number of grid points on the y-axis.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.

\text{omega} :the preconditioning parameter.

\text{h} :mesh-size.
itn : count for the number of iterations.
c .... compt the initial calculations in the algorithm
    call ax(r,f)
do 30 i=2,nx
do 30 j=2,ny
    b(i,j)=hh*.25*sin(f(i,j))
30 r(i,j)=b(i,j)-r(i,j)
call mr(p)
call vv(rmr,r,p)
40 itn=itn+1
    if(itn.eq.300) then
        write(7,90)
goto 70
    end if
do 42 i=1,nx+1
do 42 j=1,ny+1
42 f(i,j)=f(i,j)
call ax(ap,p)
rmr1=rmr
call vv(pap,p,ap)
    if(abs(pap).lt.epss) pap=epss
    alfa=rmr1/pap
do 50 i=2,nx
do 50 j=2,ny
    r(i,j)=r(i,j)-alfa*ap(i,j)
50 f(i,j)=f(i,j)+alfa*p(i,j)
call mr(rmr2)
call vv(rmr,r,rmr2)
    if(abs(rmr1).lt.epss) rmr1=epss
    beta=rmr/rmr1
do 60 i=2,nx
do 60 j=2,ny
60 p(i,j)=rmr2(i,j)+beta*p(i,j)
do 65 i=2,nx
do 65 j=2,ny
65 if(abs(ff(i,j)-f(i,j))/(1.+abs(f(i,j))).gt.eps) goto 40
     c .... print the solution.
c ....
70 write(7,80)nx,omega,itn
11 do 100 j=ny+1,1,-1
100 write(7,110)(f(i,j),i=1,nx+1)
110 format(3x,11(e14.6))
80 format(//110(1h*)//3x,'No of intervals=',i3/3x,'Omega=',f7.4//3x,'No. of iterations=',i4//}}
90 format(3x,'the max. no. of iter. has been exceeded')
stop

c .... subroutine to do matrix-vector multiplication
    subroutine ax(y,u)
dimension y(81,81),u(81,81)
common/nxy/nx,ny
do 10 i=2,nx
do 10 j=2,ny
10 y(i,j)=u(i,j)-.25*(u(i+1,j)+u(i-1,j)+u(i,j-1)+u(i,j+1))
return
end

c .......
subroutine to calculate matrix-vector multiplications.
c .... the matrix is the preconditioning matrix
c .......
subroutine mr(y)
dimension z(81,81),zz(81,81),y(81,81)
common/rr/r(81,81)
common/nxy/nx,ny/omega
do 10 j=1,nx+1
do 10 i=1,ny+1
zz(i,j)=0
y(i,j)=0
10 z(i,j)=1*r(i,j)
do 20 j=2,nx
do 20 i=2,ny
20 zz(i,j)=z(i,j)+omega*(zz(i-1,j)+zz(i,j-1))/4
do 30 j=ny,2,-1
do 30 i=nx,2,-1
30 y(i,j)=zz(i,j)+omega*(y(i,j+1)+y(i+1,j))/4
do 50 j=2,nx
do 50 i=2,ny
50 y(i,j)=y(i,j)*omega*(2-omega)
return
end

c .......
subroutine vv(y,z,zz)
dimension z(81,81),zz(81,81)
common/nxy/nx,ny
y=0
do 10 j=2,nx
do 10 i=2,ny
10 y=y+z(i,j)*zz(i,j)
return
end
This program is to solve the mildly non-linear problem (Lu = u**4) with specific boundary conditions by the Extrapolated SOR method.

The file dd contains the data (input) and the file res contains the results (output).

ny : number of the grid points on the y-axis.
com : the overrelaxation parameter.
tau : the extrapolation parameter.
h : the mesh-size.
f, ff, t, temp are 2-dimensional arrays required to carry out the iteration.
error is the error array to be used in the inner iteration test of convergence.
g is to be used in the outer iteration test of convergence.

open(o, file='dd', form='formatted', status='old')
open(5, file='res', form='formatted', status='new')
dimension f(81,81), ff(81,81), g(81,81), hhb(81,81)
dimension error(81,81), temp(81,81), t(81,81)
read(6,*) nx, ny, om, tau
h = 1./nx
hh = h*h
eps = 1e-3
epss = 5e-6
write(5,110)
110 format(100(1h*)1120x,'E.S.O.R. METHOD APPLIED TO'11)
write(5,130) nx, ny, h, om, eps, epss, tau
130 format(' NUMBER OF X INTERVALS='13/' NUMBER OF Y INTERVALS =',13/
' INTERVAL LENGTH =',f6.4/' OMEGA =',f6.4/' INNER LOOP TOLERANCE =
OUTER LOOP TOLERANCE =',f9.7/' INITIAL GUESS U(I,J)=0.01')

Setting the boundary conditions.
do 1 i=1,nx+1
1 f(i,1)=0
f(i,ny+1)=0
do 7 j=1,ny+1
f(1,j)=0
f(nx+1,j)=0
c . . . . The initial starting solution.
do 8 i=2,nx
do 8 j=2,ny
8 f(i,j)=.01
tau1=1-tau
om2=om*.25
tau2=tau*.25
do 11 i=1,nx+1
do 11 j=1,ny+1
11 temp(i,j)=f(i,j)
c . . . . Calculating the r.h.s.

... Setting the boundary conditions.
... The initial starting solution.
... Calculating the r.h.s...
220 do 2 i=2,nx
   do 2 j=2,ny
     g(i,j)=f(i,j)
   2 hhb(i,j)=hh*f(i,j)**4
      itnn=itnn+1
230 do 3 i=2,nx
   do 3 j=2,ny
     f(i,j)=tau25*(f(i-1,j)+f(i+1,j)+f(i,j-1)+f(i,j+1)-hhb(i,j))
     t(i,j)=om25*(temp(i-1,j)+temp(i,j-1)-f(i-1,j)-f(i,j-1))
     error(i,j)=temp(i,j)-f(i,j)
     itn=itn+1
   3 do 4 i=2,nx
     do 4 j=2,ny
     4 f(i,j)=temp(i,j)
        c .... The inner iteration test of convergence.
        do 10 k=2,nx
        do 10 j=2,ny
          10 if(abs(error(k,j))/1.+(abs(f(k,j))).gt.eps) goto 230
        c .... The outer iteration test of convergence.
        do 5 i=2,nx
        do 5 j=2,ny
          5 if(abs(f(i,j)-g(i,j))/(1.+abs(f(i,j))).gt.epss) then
             goto 220
        end if
        c .... Printing the output.
        write(5,150)
        150 format(' THE SOLUTION IS AS FOLLOWING, IN ORDER')
        do 6 i=nx+1,1,-1
        6 write(5,160)(f(i,j),j=1,ny+1)
        160 format(22e14.6)
        write(5,170)itn,itnn
        170 format(//2x,'Total number of inner iterations=',i4,'//')
        * i4/2x,'Total number of outer iterations=',i4//'')
        stop
  end