THE INVESTIGATION AND NUMERICAL SOLUTION OF SELECTED BOUNDARY VALUE PROBLEMS

By

RONALD CHARLES WALTERS

Bachelor of Science

Oklahoma State University

Stillwater, Oklahoma

1974

Submitted to the Faculty of the Graduate College
of the Oklahoma State University
in partial fulfillment of the requirements
for the Degree of
MASTER OF SCIENCE
May, 1976

Thesis
1976
W233i
cop. 2

OKLAHOMA
STATE UNIVERSITY
LIBRARY

AUG 26 1976

THE INVESTIGATION AND NUMERICAL SOLUTION OF SELECTED BOUNDARY VALUE PROBLEMS

Thesis Approved:

J. P. Chandler
Thesis Adviser

[illegible]

[illegible]

[illegible]
Dean of Graduate College

947675

PREFACE

This study investigates the performance of various iterative and extrapolation methods when applied to the numerical solution of two elliptic boundary value problems. The methods used can be applied to most boundary value problems in the elliptic class of partial differential equations.

I would like to express my deep appreciation to my adviser, Dr. J.P. Chandler, for his guidance and assistance through all phases of this project. A special thanks is extended to Dr. D.D. Fisher and Dr. G.E. Hedrick, the other two members on my committee. In addition, I would also like to thank all the other members of the faculty and graduate assistants for making these past two years the most enjoyable ones of my college career. Finally, thanks to my typist, Pam Haught, who spent many long hours typing the numerous equations that appear throughout the text.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

The purpose of this thesis is to analyze and solve two selected boundary value problems: the first problem is used as a model problem to help develop numerical techniques for the second problem; the second problem is a practical problem of interest in semiconductor research.

These two boundary value problems are actually two dimensional elliptic partial differential equations. There are many numerical techniques for solving partial differential equations. One of these, which is applicable to both linear and nonlinear equations, is the method of finite differences (5). This method can, in general, be applied to equations in n dimensions. Since the numerical solution of the difference equations resulting from the method of finite differences is a large undertaking, it is almost always done with the aid of automatic digital computers.

The first problem, which is used to develop and test certain iterative methods and acceleration methods, is the Dirichlet problem for Laplace's equation (7). Some of these iterative methods include the following: Jacobi, Gauss-Seidel, successive overrelaxation, simultaneous overrelaxation, and symmetric successive overrelaxation (17,18,19,20). While it is not obvious at the moment, these methods employ a reasonable amount of matrix theory, some of which is presented in sections of this paper where appropriate.

Not as widely known as some of the iterative methods, are various extrapolation techniques designed primarily to accelerate the convergence of particular iterative methods. One of these, modified vector Aitken extrapolation (8,1), will be examined in detail and applied to the iterative methods mentioned above.

The second problem, the numerical solution of the diffused semiconductor resistor, can be formulated using an elliptic partial differential equation in two dimensions, which is more complex than Laplace's equation, with boundary conditions somewhat different from the Dirichlet problem. Physically, the diffused resistor is composed of some material such as silicon with one or more metallic contacts attached to it (9). The author's version will have the contacts on the top left and right sides of a rectangular resistor. Diffused resistors are used extensively in integrated circuits, so this problem is of interest in the field of semiconductor research.

Chapter II presents some of the background of partial differential equations and boundary value problems. The Dirichlet problem is also introduced here. Chapter III contins numerical solutions of the Dirichlet problem, using a model problem developed by the author, and also Young's (17) model problem. These problems are solved by the various iterative methods mentioned earlier along with vector Aitken extrapolation. Chapter IV introduces the diffused semiconductor resistor and derives the appropriate finite difference approximations for it. Chapter V presents numerical solutions of the diffused resistor problem with the methods developed in Chapter III. Finaly, Chapter VI presents some conclusions and areas for further research.

CHAPTER II

BACKGROUND ON BOUNDARY VALUE PROBLEMS IN PARTIAL DIFFERENTIAL EQUATIONS

The general form of a second order partial differential equation in two dimensions, which is linear in the second derivatives only, can be described as follows (7). Given a two-dimensional region R:

$$a(x,y,u,u_x,u_y)u_{xx} + 2b(x,y,u,u_x,u_y)u_{xy} + c(x,y,u,u_x,u_y)u_{yy}$$
$$+ f(x,y,u,u_x,u_y) = 0 \quad , \qquad (2.1)$$

and at each point of R

$$a^2 + b^2 + c^2 \neq 0 \quad .$$

Partial derivatives; such as, $\frac{\partial u}{\partial x}$ and $\frac{\partial^2 u}{\partial x^2}$, will be represented by u_x and u_{xx}, respectively, throughout this paper. Equation (2.1) is linear when $a = a(x,y)$, $b = b(x,y)$, $c = c(x,y)$, and $f = d(x,y)u_x + e(x,y)u_y + g(x,y)u + h(x,y)$. The notation used here indicates that a, b, and c are functions of x and y alone.

Depending upon the values of a, b, and c, equations (2.1) may be in one of three general classes. If $b^2 - ac < 0$ the equation is said to be elliptic. For example, Laplace's equation (2.2), which is sometimes called the potential equation, is an elliptic partial differential equation (7).

$$u_{xx} + u_{yy} = 0 \qquad (2.2)$$

In equation (2.2), if u is time independent, it might represent the electric potential or the temperature. The variables x and y are often spatial coordinates. Equation (2.2) can also be written as:

$$\nabla^2 u = 0 \tag{2.3}$$

or

$$\Delta u = 0 \tag{2.4}$$

If $b^2 - ac = 0$ the equation is said to be parabolic. An example of a parabolic equation is the heat equation (2.5) (7).

$$u_{xx} - u_y = 0 \tag{2.5}$$

In this equation, u might be the temperature, y the time, and x a spatial coordinate.

And finally, if $b^2 - ac > 0$, the equation is in the hyperbolic class. The wave equation (2.6) is hyperbolic (7).

$$u_{xx} - u_{yy} = 0 \tag{2.6}$$

For the wave equation, u is often the amplitude (or pressure) and either x or y is time, with the other one being a spatial coordinate.

This paper will discuss and solve two particular problems in the elliptic class. The methods used will apply to most other problems in this class. To be more specific, these two problems are known as boundary value problems.

In a boundary value problem the solution to the differential equation, partial or ordinary, is desired at many points within a region. This region may be of any shape in general. One of the boundary value problems used by the author to study numerical techniques for the solution of elliptic equations, is the Dirichlet problem.

The Dirichlet problem can be stated as follows (7). Let S be the boundary, which is piecewise continuously differentiable. Let R be the interior region. If f(x,y) is given and continuous on S, then we want to find a function u(x,y) which is:

1) Defined and continuous on R and S.

2) Equal to f(x,y) on S.

3) Can be computed from an approximation of Laplace's equation on R.

This situation is depicted graphically in Figure 1.

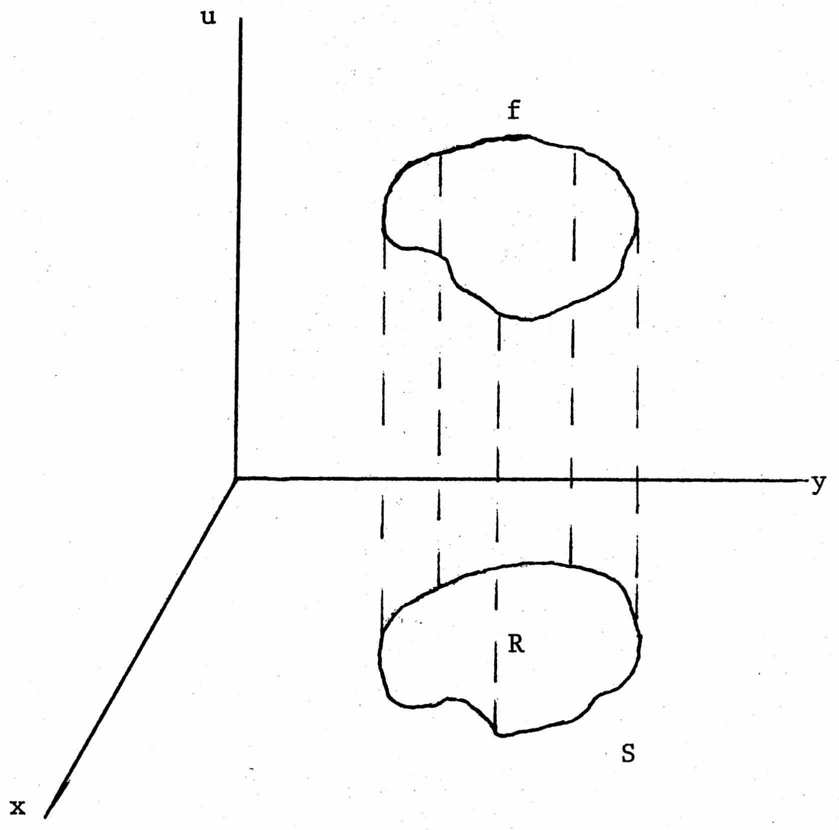

Figure 1. Geometric Description of the Dirichlet Problem

The analytical determination of u(x,y) is nontrivial in most cases, even for the most simple boundary S. If S is a rectangle, circle, or ellipse, the solution may possibly be found by use of a Fourier series or Poisson integral. Even in these cases, however, the integral may be too complex to evaluate analytically in terms of simple functions. And for most other cases there do not seem to exist analytical methods available to determine u(x,y) (7).

A commonly used numerical technique for solving boundary value problems of this nature is the <u>method of finite differences</u>. This method consists of replacing the derivatives of an ordinary differential equation or the partial derivatives of a partial differential equation by approximations derived from Taylor's series, called divided differences.

Divided differences for the first and second derivatives can be derived rather easily by use of a Taylor series expansion and provide the basis for methods to be discussed later in this paper. Finite difference methods can usually be implemented on the computer with ease, which partly accounts for their widespread use.

Usually when solving boundary value problems, a <u>mesh</u> or lattice as it is sometimes called, is used to specify the boundary, S, and the interior region, R. The points within this mesh are called mesh points and can either be a uniform distance,h, apart or more generally a non-uniform distance, h_i, apart.

To illustrate the method of finite differences we will derive the approximation to Laplace's equation, assuming equal h's. We first expand u(x+h,y) and u(x-h,y) about the point (x,y) using Taylor's series.

$$u(x+h,y) = u(x,y)+((x+h)-x)u_x(x,y) + (y-y)u_y(x,y)$$
$$+ \tfrac{1}{2}[((x+h)-x)^2u_{xx}(x,y) + 2((x+h)-x)\ (y-y)u_{xy}(x,y)$$
$$+ (y-y)^2u_{yy}(x,y)] + 0(h^3)$$
$$u(x+h,y) = u(x,y) + hu_x(x,y) + \tfrac{1}{2}h^2u_{xx}(x,y) + 0(h^3) \qquad (2.7)$$

Similarly,

$$u(x-h,y) = u(x,y)-hu_x(x,y) + \tfrac{1}{2}h^2u_{xx}(x,y) + 0(h^3) \qquad (2.8)$$

Adding (2.7) and (2.8) gives:

$$u_{xx}(x,y) = \frac{1}{h^2}[u(x+h,y)-2u(x,y) + u(x-h,y)] + 0(h^2) \qquad (2.9)$$

Similarly we can derive $u_{yy}(x,y)$ to be:

$$u_{yy}(x,y) = \frac{1}{h^2}[u(x,y+h)-2u(x,y) + u(x,y-h)] + 0(h^2) \qquad (2.10)$$

Therefore, Laplace's equation can be approximated by:

$$u_{xx} + u_{yy} \doteq \frac{1}{h^2}[u(x+h,y) + u(x-h,y) + u(x,y+h) + u(x,y-h) - 4u(x,y)] \doteq 0 \qquad (2.11)$$

It can be seen that equation (2.11) uses five different points and hence is sometimes referred to as the <u>five point rule</u>. In general, the four points (x+h,y), (x,y+h), (x-h,y), and (x,y-h) may not be the same distance, h, from the point (x,y). So, we can define four new points: $(x+h_1,y)$, $(x,y+h_2)$, $(x-h_3,y)$, and $(x,y-h_4)$ where $h_1 \neq h_2 \neq h_3 \neq h_4$. Figure 2 graphically represents the five point rule for this case.

Thus, using the same techniques which were used to derive equation (2.11) we can derive another approximation to Laplace's equation for the general case of unequal h's. For notational purposes, let $u_1 = u(x+h_1,y)$, $u_2 = u(x,y+h_2)$, $u_3 = u(x-h_3,y)$, $u_4 = u(x,y-h_4)$, and $u_0 = u(x,y)$. Also since all derivatives are functions of x and y alone they will be written without the (x,y) part.

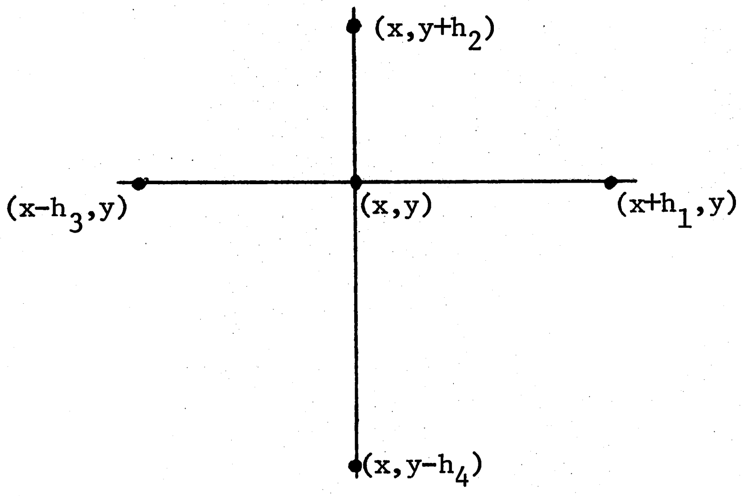

Figure 2. Five Point Rule Using Unequal h's.

Expanding u_1 and u_3 about the point (x,y) using Taylor's series gives:

$$u_1 = u_0 + h_1 u_x + \tfrac{1}{2} h_1^2 u_{xx} + 0(h_1^3) \tag{2.12}$$

$$u_3 = u_0 - h_3 u_x + \tfrac{1}{2} h_3^2 u_{xx} + 0(h_3^3) \tag{2.13}$$

Adding (2.12) and (2.13) and simplifying gives:

$$u_{xx} \doteq \frac{2\,u_1}{h_1(h_1 + h_3)} + \frac{2\,u_3}{h_3(h_1 + h_3)} - \frac{2\,u_0}{h_1 h_3} \tag{2.14}$$

Similarly, expanding u_2 and u_4 about the point (x,y) using Taylor's series gives:

$$u_2 = u_0 + h_2 u_y + \tfrac{1}{2} h_2^2 u_{yy} + 0(h_2^3) \tag{2.15}$$

$$u_4 = u_0 - h_4 u_y + \tfrac{1}{2} h_4^2 u_{yy} + 0(h_4^3) \tag{2.16}$$

Adding (2.15) and (2.16) and simplifying gives:

$$u_{yy} \doteq \frac{2\,u_2}{h_2(h_2 + h_4)} + \frac{2\,u_4}{h_4(h_2 + h_4)} - \frac{2\,u_0}{h_2 h_4} \tag{2.17}$$

The approximation for Laplace's equation for unequal h's follows directly:

$$u_{xx} + u_{yy} = \frac{2\,u_1}{h_1(h_1 + h_3)} + \frac{2\,u_2}{h_2(h_2 + h_4)} + \frac{2\,u_3}{h_3(h_1 + h_3)} + \frac{2\,u_4}{h_4(h_2 + h_4)}$$

$$- 2\left(\frac{1}{h_1 h_3} + \frac{1}{h_2 h_4}\right)u_0 + 0(h^2) = 0 \qquad (2.18)$$

It should be noted that the above equation (2.18) reduces to (2.11) if $h_1 = h_2 = h_3 = h_4 = h$.

Depending upon the boundary, S, either equation (2.11) or (2.18) could be used to solve the Dirichlet probelm. In the Dirichlet problem we want to compute u(x,y) (u_0) for all points in the region R. Therefore we will have, in general, n equations of the form of equation (2.11) or (2.18) which must be solved simultaneously.

Equations (2.11) and 2.18) are both linear and hence may be solved by a direct method such as Gaussian elimination (13). The coefficient matrix for these equations is very sparse, each row consisting of at most five nonzero coefficients (5). If the entire coefficient matrix is stored, n^2 storage locations would be required.

For most practical problems n is very large, so that solution by a direct method becomes impractical. Hence, iterative methods have been developed which can solve these equations using c·n storage locations, where c is usually less than five.

Although the finite difference method has been used extensively. there is another method, the finite element method (21), which should be discussed briefly. Using this method, the solution to the partial differential equation can be computed by finding a function, u, which minimizes an appropriate integral. This integral comes from applying the Euler conditions of the calculus of variations to the partial differential equation, and for Laplace's equation, may be given as (21)

$$I = \tfrac{1}{2} \iint (u_x^2 + u_y^2)\,dxdy \quad . \qquad (2.19)$$

The boundary conditions remain unchanged, and the integration is carried out over the whole region.

The region, R, can be divided up into small elements, such as triangles, as shown in Figure 3.

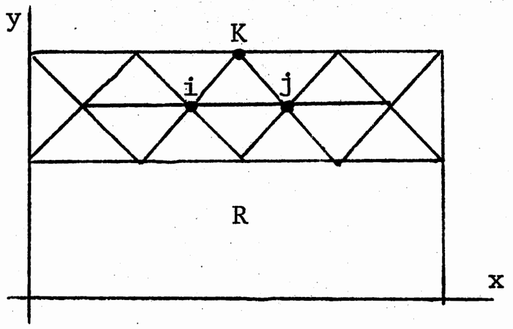

Figure 3. Finite Elements

Zienkiewicz and Cheung (21) then describe how the function u can be uniquely specified within an element whose nodal values are u_i, u_j, and u_k by a linear function of the coordinates x and y. The function varies linearly along the lines connecting the nodal points and therefore must be uniquely defined by two nodal values from any triangular element. In other words, the solution u(x,y) must be the same on the boundary between any two triangular elements, otherwise the resulting discontinuity would cause the integral (2.19) to become infinite.

Geometric shapes other than the triangle could be used for the basic element, such as the parallelogram. These other shapes may result in a better approximation, depending upon the problem, but the basic procedure is not changed greatly. Although not used by the author, the finite element method appears to be a reasonably good, although rather complex method.

CHAPTER III

ITERATIVE AND EXTRAPOLATION METHODS FOR SOLVING ELLIPTIC BOUNDARY VALUE PROBLEMS

When solving elliptic partial differential equations by a finite difference method, one is usually confronted with the problem of solving a large system of equations. If the partial differential equation is linear (as are Laplace's equation and the equation for the diffused resistor), this system is linear and the coefficient matrix is usually very large and very sparse (20).

This chapter examines some of the iterative methods used by Young (17), Varga (14), Wachspress (15), and Smith (12) in an attempt to compare their speed and storage requirements.

In developing these methods, the square mesh in Figure 4 is used to specify the boundary and interior region, using $h = 1/3$. This mesh is similar to the mesh used by Young (17,18,19,20) for his model problem. His model problem used $f(x,y) = 0$ to specify $u(x,y)$ at the boundary. This mesh contains 16 total points, 12 of them boundary points at which the value of $u(x,y)$ is given as $f(x,y)$, and 4 interior points at which the value of $u(x,y)$ can be computed approximately from the difference equations. Reference will be made to $u(x,y)$ throughout this paper and it should be remembered that this is an approximation.

Obviously the value of $u(x,y)$ could be computed directly by elimination methods, but in most problems of this nature such as

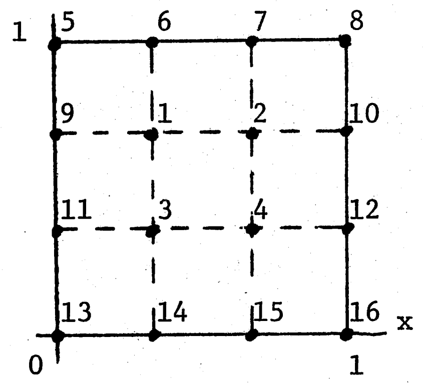

Figure 4. Sample Mesh

neutron diffusion, fluid flow, elasticity, steady-state heat flow, and weather prediction (17), the mesh may contain many points, making direct solution impractical. Some of the iterative methods for solving linear systems are outlined next. They are illustrated using the equations which result from applying the five-point rule to the model problem, but they are not restricted to that case.

The Jacobi Method

The system of difference equations generated from the finite difference method are of the form of equation (2.8), and for Figure 4 are as follows:

$$\begin{aligned} 4u_1 - u_2 - u_6 - u_9 - u_3 &= 0 \\ 4u_2 - u_{10} - u_7 - u_1 - u_4 &= 0 \\ 4u_3 - u_4 - u_1 - u_{11} - u_{14} &= 0 \\ 4u_4 - u_{12} - u_2 - u_3 - u_{15} &= 0 \end{aligned} \tag{3.1}$$

Moving the boundary values to the right hand side, the equations in coefficient form may be written as:

$$\begin{bmatrix} 4 & -1 & -1 & 0 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} u_6 + u_9 \\ u_7 + u_{10} \\ u_{11} + u_{14} \\ u_{12} + u_{15} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} \tag{3.2}$$

In matrix form this may be written as:

$$A\underline{u} = b \quad , \tag{3.3}$$

where A is the coefficient matrix of the original system. Let L and U be the strictly lower and upper triangular matrices shown below.

$$L = \begin{bmatrix} 0 & 0 & 0 & 0 \\ \frac{1}{4} & 0 & 0 & 0 \\ \frac{1}{4} & 0 & 0 & 0 \\ 0 & \frac{1}{4} & \frac{1}{4} & 0 \end{bmatrix}, \quad U = \begin{bmatrix} 0 & \frac{1}{4} & \frac{1}{4} & 0 \\ 0 & 0 & 0 & \frac{1}{4} \\ 0 & 0 & 0 & \frac{1}{4} \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.4}$$

A matrix, M_J, can be defined such that

$$L + U = M_J \quad . \tag{3.5}$$

So, using M_J, the system may also be written in the form:

$$\underline{u} = M_J\underline{u} + \underline{c} \quad , \tag{3.6}$$

where $\underline{c} = \frac{1}{4}\underline{b}$.

This form becomes the basis for the first iterative method to be considered, the Jacobi method.

$$\begin{aligned} u_1^{(i+1)} &= \qquad \tfrac{1}{4}u_2^{(i)} + \tfrac{1}{4}u_3^{(i)} \qquad + c_1 \\ u_2^{(i+1)} &= \tfrac{1}{4}u_1^{(i)} \qquad\qquad + \tfrac{1}{4}u_4^{(i)} + c_2 \\ u_3^{(i+1)} &= \tfrac{1}{4}u_1^{(i)} \qquad\qquad + \tfrac{1}{4}u_4^{(i)} + c_3 \\ u_4^{(i+1)} &= \qquad \tfrac{1}{4}u_2^{(i)} + \tfrac{1}{4}u_3^{(i)} \qquad + c_4 \end{aligned} \tag{3.7}$$

Here $u^{(i)}$ and $u^{(i+1)}$ denote the values of a particular u after the ith and (i+1)th iterations, respectively. The choice of $u^{(0)}$ is arbitrary.

Hence, the approximate value of u(x,y) at the interior points of the mesh may be computed by (3.8).

$$u^{(i+1)}(x,y) = \tfrac{1}{4}[u^{(i)}(x+h,y) + u^{(i)}(x,y+h) + u^{(i)}(x-h,y) + u^{(i)}(x,y-h)] \tag{3.8}$$

where each $u^{(i+1)}$ is computed from the u's of the i^{th} iteration, and can be computed in any order (12).

The Jacobi method (also called the method of simultaneous displacements) has been shown to be slow in convergence and not effective for meshes containing a large number of points (12,14,17). However, it is of theoretical value and provides a starting point for developing other methods.

The Gauss-Seidel Method

The system (3.7) can obviously be altered slightly to the follwoing:

$$\begin{aligned}
u_1^{(i+1)} &= \phantom{\tfrac{1}{4}u_1^{(i+1)}} \tfrac{1}{4}u_2^{(i)} + \tfrac{1}{4}u_3^{(i)} + c_1 \\
u_2^{(i+1)} &= \tfrac{1}{4}u_1^{(i+1)} + \tfrac{1}{4}u_4^{(i)} + c_2 \\
u_3^{(i+1)} &= \tfrac{1}{4}u_1^{(i+1)} + \tfrac{1}{4}u_4^{(i)} + c_3 \\
u_4^{(i+1)} &= \tfrac{1}{4}u_2^{(i+1)} + \tfrac{1}{4}u_3^{(i+1)} + c_4
\end{aligned} \tag{3.9}$$

This system may be written in matrix form

$$\underline{u}^{(i+1)} = L\underline{u}^{(i+1)} + U\underline{u}^{(i)} + \underline{c} \quad , \tag{3.10}$$

which is equivalent to (18)

$$\underline{u}^{(i+1)} = M_{GS}u^{(i)} + (I-L)^{-1}\underline{c} \quad , \tag{3.11}$$

where $M_{GS} = (I-L)^{-1}U.$ (3.12)

In this case we are using the latest iterative values when computing $u_2^{(i+1)}$, $u_3^{(i+1)}$, and $u_4^{(i+1)}$. This variation is called the <u>Gauss-Seidel</u> method. Equation (3.13) describes the iterative process for computing the value of u(x,y) at the interior points (12).

$$u^{(+1)}(x,y) = \tfrac{1}{4}[u^{(i)}(x+h,y) + u^{(i+1)}(x,y+h) + u^{(i+1)}(x-h,y) + u^{(i)}(x,y-h)] \qquad (3.13)$$

In order to use the (i+1) values in the (i+1)th iteration they must have been computed previously. The order which is implied by the system (3.9) is sometimes called reading order; that is, from left to right and top to bottom of the mesh.

An implementation advantage of this method is that the u's from the previous iteration do not have to be stored, thus reducing storage requirements when compared to the Jacobi method. Smith (12) and others (14,15,17) have shown that the Gauss-Seidel method converges about twice as fast as the Jacobi method.

The Successive Overrelaxation Method

The Gauss-Seidel iteration may also be written as (12):

$$\begin{aligned} u^{(i+1)}(x,y) &= u^{(i)}(x,y) + \tfrac{1}{4}[u^{(i)}(x+h,y) + u^{(i+1)}(x,y+h) + u^{(i+1)}(x-h,y) \\ &\quad + u^{(i)}(x,y-h) - 4u^{(i)}(x,y)] \\ &= u^{(i)}(x,y) + R^{(i)}(x,y) \quad , \end{aligned} \qquad (3.14)$$

so that $R^{(i)}(x,y)$ is the change in the value of u(x,y) for one Gauss-Seidel iteration. The new method, called <u>successive overrelaxation</u> (SOR), uses a larger change in $u^{(i)}(x,y)$. So, the value of u(x,y) at the interior points may be computed by (3.15)

$$\begin{aligned} u^{(i+1)}(x,y) &= u^{(i)}(x,y) + wR^{(i)}(x,y) \\ u^{(i+1)}(x,y) &= u^{(i)}(x,y) + w[\tfrac{1}{4}(u^{(i)}(x+h,y) + u^{(i+1)}(x,y+h) \\ &\quad + u^{(i+1)}(x-h,y) + u^{(i)}(x,y-h)) - u^{(i)}(x,y)] \end{aligned} \qquad (3.15)$$

In matrix form, this may be written as:

$$\underline{u}^{(i+1)} = w(L\underline{u}^{(i+1)} + U\underline{u}^{(i)} + c) + (1-w)\underline{u}^{(i)} \quad , \qquad (3.16)$$

which is equivalent to (18)

$$u^{(i+1)} = M_{SOR}\underline{u}^{(i)} + (I-wL)^{-1}w\underline{c} \quad , \tag{3.17}$$

where $M_{SOR} = (I-wL)^{-1}[wU+(1-w)I]$. (3.18)

This w is termed the relaxation factor. Note that when w is equal to one, the SOR iteration is the same as Gauss-Seidel iteration. Young (17) shows that if A is a symmetric matrix with positive diagonal elements then the SOR method converges if and only if A is positive definite and $0 < w < 2$. Positive definite means that A is nonsingular, with positive diagonal elements, and that the eigenvalues of A are real and positive (19).

Young (17) and others (5,12,14,15) describe methods for determining the optimum w for SOR. Using the matrix M_J defined by equation (3.5), the optimum w may be determined by

$$w_b = \frac{2}{1+[1-S^2(M_J)]^{\frac{1}{2}}} \tag{3.19}$$

where $S(M_J)$ is the spectral radius of M_J. The spectral radius of a matrix is the magnitude of the largest eigenvalue.

Computing the spectral radius of M_J would not be a trivial task and was not done by the author. $S(M_J)$ is known for Young's model problem, and is given later.

Forsythe and Wasow (5) describe a slightly different method for computing w_b which is given below.

$$w_b = \frac{2}{1+(1-d)^{\frac{1}{2}}} \quad , \tag{3.20}$$

where $d = \lim_{i\to\infty} \frac{||\underline{u}^{(i+1)}-\underline{u}^{(i)}||_1}{||\underline{u}^{(i)}-\underline{u}^{(i-1)}||_1}$ (3.21)

The constant d must be computed from the $\underline{u}$'s using the Gauss-Seidel method. The L_1 norm of $(\underline{u}^{(i+1)}-\underline{u}^{(i)})$ can be computed during an iteration of Gauss-Seidel, so d does not require much additional computation.

The Simultaneous Overrelaxation Method

A method similar to the SOR method, but based on the Jacobi method, is the simultaneous overrelaxation (JOR) method (17). Equation (3.22) describes the iterative process for computing u(x,y) at the interior points.

$$u^{(i+1)}(x,y) = u^{(i)}(x,y) + w[\tfrac{1}{4}(u^{(i)}(x+h,y) + u^{(i)}(x,y+h) + u^{(i)}(x-h,y) + u^{(i)}(x,y-h)) - u^{(i)}(x,y)]. \quad (3.22)$$

Here, the u's may be computed in any order, as in the Jacobi method. Using matrix notation, the JOR method may be expressed as

$$\underline{u}^{(i+1)} = w(M_J\underline{u}^{(i)}+\underline{c}) + (1-w)\underline{u}^{(i)} \quad , \qquad (3.23)$$

whish is equivalent to (17)

$$\underline{u}^{(i+1)} = M_{JOR}\underline{u}^{(i)} + w\underline{c} \quad , \qquad (3.24)$$

where

$$M_{JOR} = wM_J + (1-w)I \quad . \qquad (3.25)$$

Young (17) shows that if the Jacobi method converges then the JOR method will converge for $0 < w \leq 1$.

The matrices M_J, M_{GS}, M_{SOR}, and M_{JOR} are usually called the iteration matrices of the Jacobi, Gauss-Seidel, SOR, and JOR methods, respectively (14). They are not computed explicitly during the application of one of these methods, but their eigenvalues are of theoretical interest which will be mentioned later.

Other Rules

Although the iterative methods above were illustrated by the linear system from the basic five-point rule for Laplace's equation with fixed h's, other rules do exist and the respective Jacobi, Gauss-Seidel, SOR, and JOR methods can be applied to any of them. Two of these rules are described below.

An alternative five point rule for Laplace's equation with equal h's can be given as (10)

$$u_0 = \tfrac{1}{4}(u_1+u_2+u_3+u_4) + O(h^2) \quad , \tag{3.26}$$

which is the same as before, except that the five points used are those shown in Figure 5.

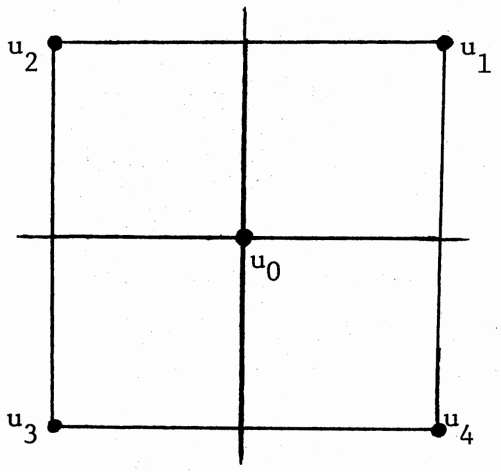

Figure 5. Alternate Five-Point Rule

Another rule, the nine-point rule, is described by the following equation, using fixed h's (10)

$$u_0 = \frac{1}{5}(u_1+u_2+u_3+u_4) + \frac{1}{20}(u_5+u_6+u_7+u_8) + O(h^6) \qquad (3.27)$$

This rule uses the nine points shown in Figure 6.

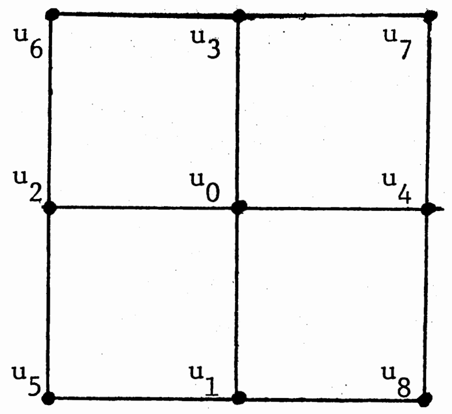

Figure 6. Nine-Point Rule

The truncation error is of order h^6 as shown by Forsythe and Wasow (5) for Laplace's equation. In all other instances the truncation error would be $O(h^4)$. Even so, this is higher order accuracy than the $O(h^2)$ of the five-point rule.

Numerical Solution of the Dirichlet Problem

The Dirichlet problem was solved numerically using the basic five-point rule with fixed h's equal to one. The mesh of Figure 7, containing a total of 121 points, was used to specify the boundary and interior region.

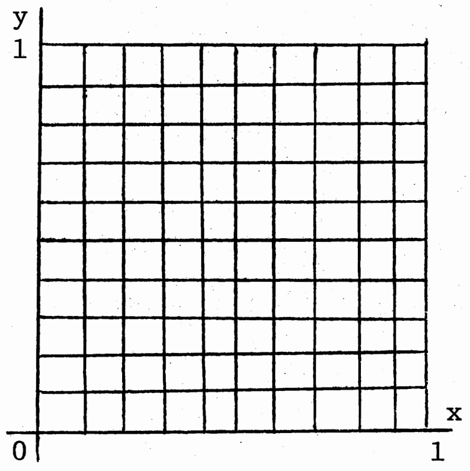

Figure 7. Author's Model Mesh, $h = \frac{1}{10}$

The function $f(x,y) = 5(x+y)$ was used to specify $u(x,y)$ on the boundary. This leaves a total of 81 interior points at which the value of $u(x,y)$ must be computed from the difference equations, using an appropriate iterative method. The initial value of $u(x,y)$ at these interior points was arbitrarily chosen to be zero. Remember that there will be 81 equations to solve. A program was developed which incorporates the methods discussed previously and was used to solve the Dirichlet problem.

Before analyzing the results, the convergence criterion should be discussed briefly. Two different indicators of convergence have been used,

$$P = \log_{10}\left[\sum_{k=1}^{n}(R^{(k)}(x,y))^2\right]^{\frac{1}{2}} \qquad (3.28)$$

and

$$G = \sum_{k=1}^{n}\left|R^{(k)}(x,y)\right| \quad , \qquad (3.29)$$

where n is the number of difference equations. The value of $R^{(k)}(x,y)$, from equation (3.14), is available directly in each iteration of all the methods, so little additional computation is required to compute either P or G. P is actually the $\log_{10}$ of the L_2 norm of the $\Delta \underline{u}$ vector, when using either Gauss-Seidel or Jacobi. This value should theoretically approach minus infinity. G is the L_1 norm of the $\Delta \underline{u}$ vector when using Gauss-Seidel or Jacobi. This value, which should approach zero, is used as the stopping criterion for all of the methods.

Numerical solution of the Dirichlet problem was done with the Jacobi, Gauss-Seidel, and SOR methods initially. The results are shown in Table I.

TABLE I

RESULTS OF THE JACOBI, GAUSS-SEIDEL, AND SOR METHODS

Method	P_{50}	$-\frac{1}{m}$
Jacobi	-0.7774	45.58
Gauss-Seidel	-1.5263	23.04
SOR, w=1.5348	-10.8067	3.94

The m-value in Table I is the slope of the curve P vs Iter which is computed as follows:

$$m = \frac{P_{50}-P_{25}}{50-25} \quad , \tag{3.30}$$

where P_{50} and P_{25} are the values of P after 50 and 25 iterations respectively. So, the value of $-\frac{1}{m}$ is the number of iterations required to reduce the norm of the residual vector, $\underline{R}(x,y)$, by a factor of ten, or to add one more digit of accuracy to $\underline{u}$.

The value of $-\frac{1}{m}$ appears to be a reasonably good indicator of how a method is converging. For example, SOR produces one digit of accuracy after about 4 iterations, Gauss-Seidel requires about 23 iterations, and Jacobi requires about 46 iterations. Thus, Gauss-Seidel does indeed converge about twice as fast as Jacobi, but still considerably slower than SOR. This is consistent with Young's (17) theory and his work with the model problem.

The value of the optimum w, w_b, was found to be about 1.5348 using equations (3.20) and (3.21). Recall that when using these equations, the value of the optimum w is computed during each iteration of Gauss-Seidel method. The value of the optimum w stated above came from the 17th iteration of Gauss-Seidel. When the SOR method was applied with other w's, the values of $-\frac{1}{m}$ were larger, which means that these w's were not optimum. Figure 8 displays graphically P vs w after 20 iterations of SOR, using various values of w. Figure 9 illustrates the performance of Jacobi, Gauss-Seidel, and SOR with w_b.

The graph of Figure 8 is very similar to Young's (17) graph of $S(M_{SOR})$ vs w for his model problem. He shows that as the value of w approached w_b, the slope of $S(M_{SOR})$ approached minus infinity. Figure 8 also shows this square root dependence on the left side of the

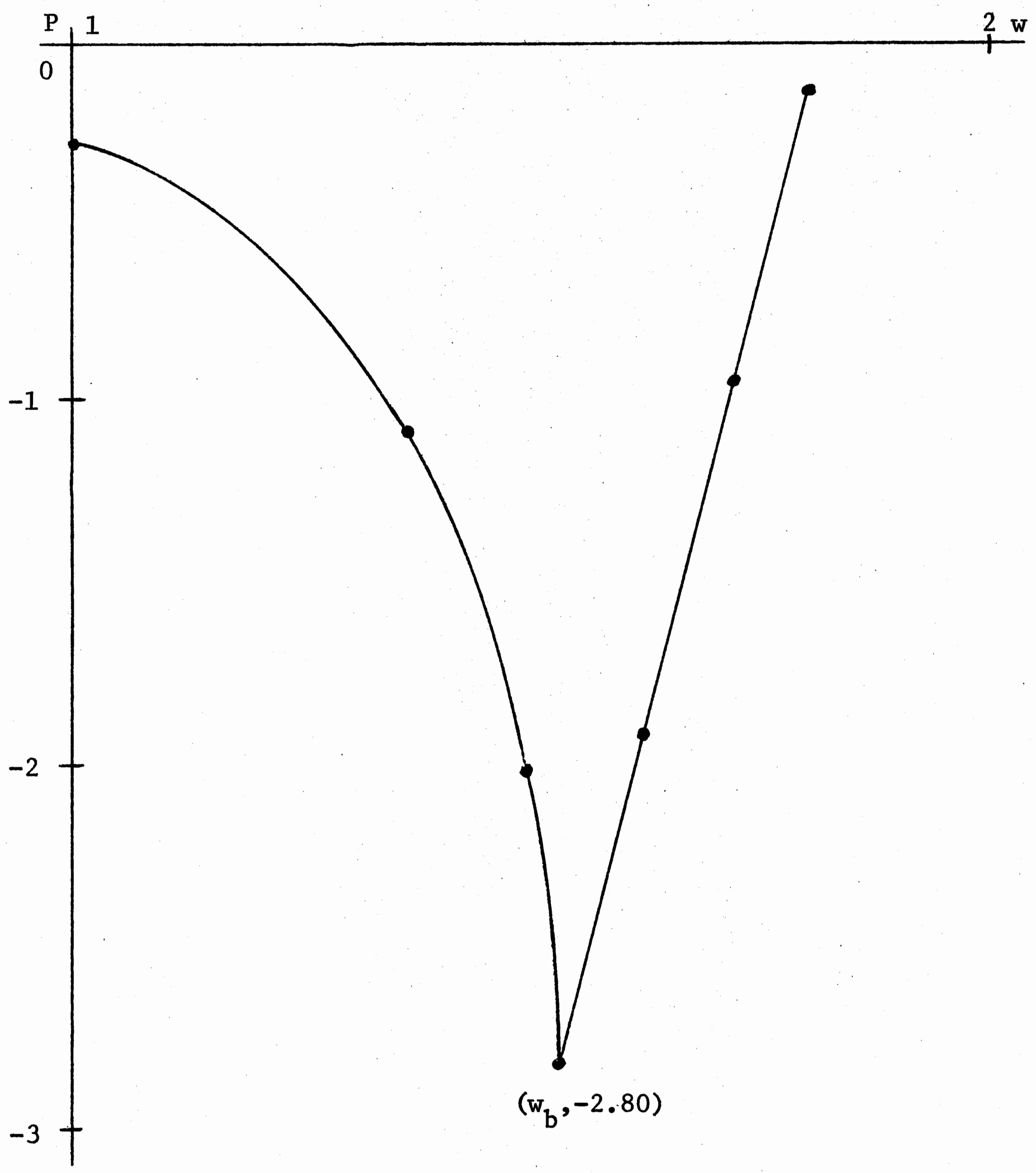

Figure 8. P vs w for SOR

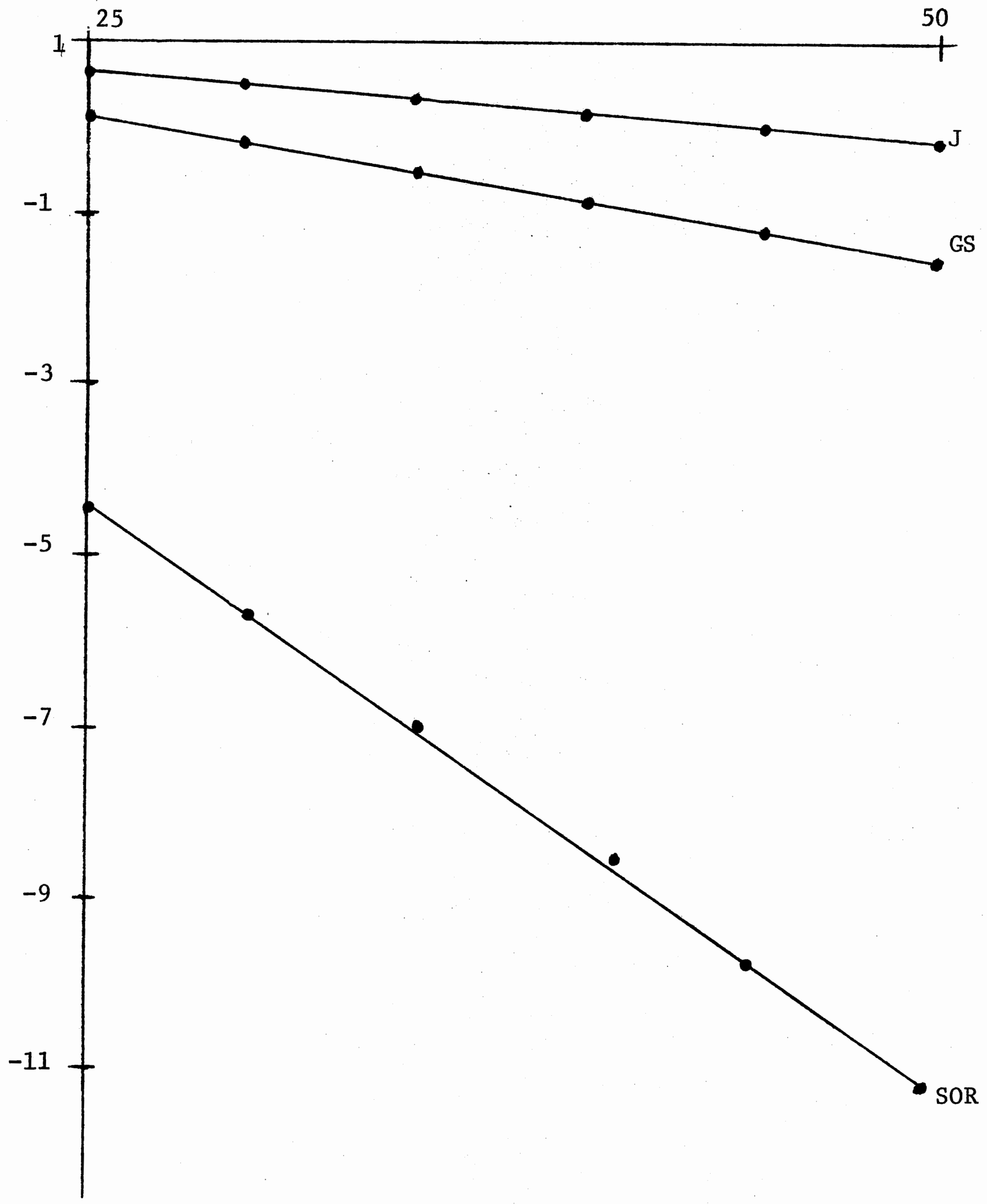

Figure 9. Comparison of Jacobi, Gauss-Seidel, and SOR

minimum (w_b). On the right side of w_b, the graph is linear. Thus, small decreases in the value of w result in a much larger relative change in the rate of convergence than corresponding increases in w.

Summarizing, Young (18) determined the following values for his model problem.

$$S(M_J) = \cos \pi h \tag{3.31}$$

$$S(M_{GS}) = \cos^2 \pi h \tag{3.32}$$

$$S(M_{SOR}) = \frac{1-\sin \pi h}{1+\sin \pi h} \tag{3.33}$$

and

$$w_b = \frac{2}{1+\sin \pi h} \tag{3.34}$$

Hence, SOR does appear to be one of the best methods available to solve the Dirichlet problem. Its major advantages are its fast convergence, and low storage requirements (only one $\underline{u}$ vector). Its major disadvantage is the difficulty in determining w_b. This problem was not of major concern for the author's model problem, but Forsythe and Wasow (5) state a major disadvantage of computing w_b from equations (3.20) and (3.21): A prohibitive number of iterations of Gauss-Seidel would probably be required to compute w_b for very fine meshes. This comes from the fact that d is computed using a limit. So, numerical analysts have developed other methods which compete favorably with SOR. One of these is the next topic of this chapter.

Modified Vector Aitken Extrapolation

Scalar Aitken extrapolation replaces a linearly converging sequence x_0, x_1, x_2,...,x by a faster converging sequence $x_1^{(1)}$, $x_2^{(1)}$, ..., x where x is the desired limiting value (4).

If for a convergent sequence the limit

$$\lim_{k \to \infty} \frac{|x_k - x|}{|x_{k-1} - x|^q} \equiv C \tag{3.35}$$

exists, and C is not zero, then the sequence is said to converge to x with order q. We will consider the case of linear convergence, for which q = 1. Thus, we have

$$x_k - x \doteq C(x_{k-1} - x) \quad , \tag{3.36}$$

where x and C are unknown. To solve for x, we need another equation of the form (3.36), such as

$$x_{k+1} - x \doteq C(x_k - x) \quad . \tag{3.37}$$

Solving equations (3.36) and (3.37) will give an approximation to x which we will call $x_k^{(1)}$.

$$x_k^{(1)} = x_{k+1} + s(x_{k+1} - x_k) \quad , \tag{3.38}$$

where

$$s = \frac{x_k - x_{k+1}}{x_{k+1} - 2x_k + x_{k-1}} \tag{3.39}$$

Since the u's in the Dirichlet problem actually form a vector $\underline{u}$, it would be reasonable to believe that component-by-component scalar extrapolation of $\underline{u}$ would work well. However, Aitken has attempted this, and has found that it does not work well. A few components of $\underline{u}$ may be extrapolated a long way, or in the wrong direction. So, a different method of vector extrapolation must be devised.

One could replace the scalars of equations (3.38) and (3.39) by the $\underline{u}$ vectors and then extrapolate the three vectors $\underline{u}_{k+1}$, $\underline{u}_k$, and $\underline{u}_{k-1}$, except that division by a vector is undefined. To eliminate this problem, we can premultiply the denominator and numerator of s by the transpose of an appropriate vector, which will reduce s to scalar form. Equations (3.40) and (3.41) describe this modification, which will be called <u>modified vector Aitken extrapolation</u> (1,8).

$$\underline{u} = \underline{u}^{(i+1)} + s\cdot(\underline{u}^{(i+1)}-\underline{u}^{(i)}) \quad , \qquad (3.40)$$

where
$$s = \frac{\underline{z}^T\cdot(\underline{u}^{(i)}-\underline{u}^{(i+1)})}{\underline{z}^T\cdot(\underline{u}^{(i+1)}-2\underline{u}^{(i)}+\underline{u}^{(i-1)})} \qquad (3.41)$$

Jennings (8) calls the vector $\underline{z}$ a weight vector. This vector may have two natural forms, which are discussed next. Suppose that, in analogy to equation (3.36)

$$\underline{u}^{(i)}-\underline{u} = M(\underline{u}^{(i-1)}-\underline{u}) \quad , \qquad (3.42)$$

where $\underline{u}$ is the desired limiting vector and M is the iteration matrix. If M is symmetric, $\underline{z}$ can be given as

$$\underline{z} = \underline{u}^{(i-1)}-\underline{u}^{(i)} \quad . \qquad (3.43)$$

However, if M is unsymmetric, Jennings (8) states that there is the possibility of cancellation in the denominator of s in equation (3.41) if (3.43) is used. In this case, equation (3.44) should be used for $\underline{z}$.

$$\underline{z} = \underline{u}^{(i+1)}-2\underline{u}^{(i)}+\underline{u}^{(i-1)} \qquad (3.44)$$

Equations (3.43) and (3.44) are sometimes called the first <u>difference modulation</u> (FDM) and the <u>second difference modulation</u> (SDM) (8). Note that the SDM form is also valid when the iteration matrix is symmetric.

Additional references on vector extrapolation include separate articles by Wynn (16) and Gekeler (6), and an article by Brezinski and Rieu (2).

Referring to equation (3.41), the value of s indicates the amount of change the extrapolation will cause as a multiple of the last iteration (8). Boyle and Jennings (1), during their elastic-plastic stress analysis, had s-values ranging from one half to five.

It was found that extrapolation did not always improve convergence of certain iterative methods as much as expected. Examining the $\Delta\underline{u}$ vector computed by these iterative methods showed that it was changing in sign or value in a rhythmic pattern. That is, it was zigzagging. The type of zigzagging observed is depicted in Figure 10.

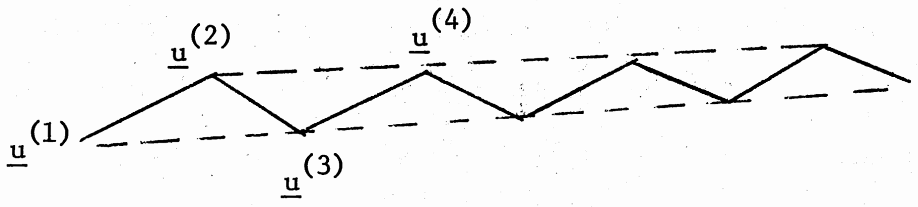

Figure 10. $\Delta\underline{u}$ Zigzags

It can be seen from Figure 10 that extrapolating $\underline{u}^{(1)}$, $\underline{u}^{(2)}$, and $\underline{u}^{(3)}$ will probably not help convergence much. However if every other $\underline{u}^{(i)}$ is discarded, and the remaining $\underline{u}$'s extrapolated, convergence may be improved. This would correspond to extrapolating along the peaks or valleys of Figure 10.

Jennings (8) shows that when extrapolating an iterative method whose iteration matrix is symmetric with eigenvalues between -1 and 1, convergence was not improved very much, which is similar to the results observed when the $\Delta\underline{u}$ vector zigzags. So, it is worthwhile to reexamine some of the iteration matrices for the methods we have developed.

Table II outlines some of the aspects of the iteration matrices for the iterative methods discussed so far (8,17). The matrix, M_{SSOR}, comes from an iterative method which will be discussed later.

TABLE II

ITERATION MATRICES

Iteration Matrix	Type	Eigenvalues
M_J	symmetric	real, -1 to 1
M_{JOR}	symmetric	real, -1 to 1
M_{GS}	unsymmetric	mostly real, 0 to 1
M_{SOR}	unsymmetric	complex, when $w=w_b$
M_{SSOR}	unsymmetric	real, 0 to 1

The results in Table II are true if A, the coefficient matrix, is positive definite.

Table II shows that when $\underline{u}^{(i-1)}$, $\underline{u}^{(i)}$, and $\underline{u}^{(i+1)}$ are extrapolated using the Jacobi method (or the JOR method) convergence will not improve much. Jennings (8) suggested that it is better to extrapolate $\underline{u}^{(i-1)}$, $\underline{u}^{(i+1)}$, and $\underline{u}^{(i+3)}$, which has the effect of squaring the iteration matrix, giving only positive eigenvalues. This same conclusion was reached when observing the $\Delta\underline{u}$ vector earlier. Thus, in actual practice it is not necessary to compute the eigenvalues to determine which $\underline{u}$ vectors to extrapolate. The terms period one and period two will be used to indicate if extrapolation was performed on the vectors $\underline{u}^{(i-1)}$, $\underline{u}^{(i)}$, and $\underline{u}^{(i+1)}$, or $\underline{u}^{(i-1)}$, $\underline{u}^{(i+1)}$, and $\underline{u}^{(i+3)}$, respectively.

Repeated Aitken Extrapolation

The scalar Aitken extrapolation described by equation (3.38) may be repeated, producing an even faster converging sequence. That is, the extrapolated values $x_1^{(1)}$, $x_2^{(1)}$,...,x may be extrapolated producing the sequence $x_2^{(2)}$, $x_3^{(2)}$,..., x. This process may be repeated until only one or two values remain. Figure 11 gives an example of this.

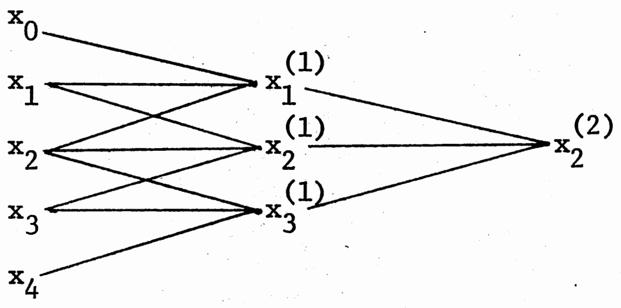

Figure 11. Repeated Aitken

The superscriptes indicate the number of times extrapolation has been applied. Here, the results of the previous extrapolation, the $x^{(i)}$ values must be stored before they may be extrapolated producing the $x^{(i+1)}$ values.

Jennings (8) describes a version of repeated Aitken extrapolation, called <u>double extrapolation</u>. Double extrapolation, using FDM, is done using vector iterates in the same manner the scalar iterates are extrapolated in Figure 11. The rate of convergence was found to be about 7.9

times faster than the convergence rate without extrapolation (8).

Another form of repeated vector Aitken extrapolation is similar in technique to Steffensen iteration for scalars (4). This form, which will be called super extrapolation, is illustrated in Figure 12.

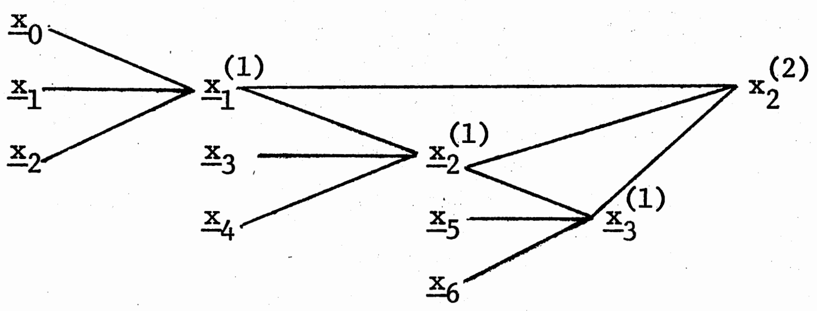

Figure 12. Super Extrapolation

Using this technique only three vectors need to be stored when extrapolation is applied once, and only five vectors need to be stored when it is repeated one time.

The actual motivation for super extrapolation came from observing the s-values after the original extrapolations were performed. In most cases, these s-values oscillated (zigzagged) in a high/low pattern similar to the way the $\Delta\underline{u}$ vector zigzagged when extrapolating the Jacobi method. So, it was found that extrapolating $\underline{v}^{(k-1)}$, $\underline{v}^{(k+1)}$, and $\underline{v}^{(k+3)}$, where the $\underline{v}$'s are the results from extrapolating the $\underline{u}$'s, produced large improvements in convergence. We will again use the terms: period two and period one, to indicate whether or not the s-values are oscillating.

An Implementation and Results of Modified Vector Aitken Extrapolation

Chanlder (3) has written a modified vector Aitken extrapolation subroutine, VAITK, which incorporates many of the features of extrapolation discussed previously. Extrapolation and super extrapolation are both available, using either FDM or SDM. The technique of Steffensen iteration, discussed previously, is used to reduce storage requirements.

Boyle and Jennings (1) found that, for their stress analysis problem, when zigzagging is not occurring, it is best to extrapolate as soon as the $\underline{u}$ vectors are available; that is, after every two iterations. This was found not to be true in general. At times it may be desirable to perform one or two preparatory iterations and then begin saving the $\underline{u}$ vectors for extrapolation. Intuitively, this gives the $\underline{u}$ vector time to "settle down" after an extrapolation, before extrapolating again. Shortly and Weller (11) speak of allowing the error in $\underline{u}^{(i)}$ to smooth out and assume a "pillow-shaped" form. This feature of allowing preparatory iterations is also incorporated into VAITK.

The results of extrapolation and super extrapolation, using SDM when applied to Gauss-Seidel are shown in Table III. The period of the iterations is indicated in the table. When super extrapolation is applied, an extrapolation period of two is used. The abbreviation "Prep." means preparatory. So, "No. of Prep. Iter." means the number of preparatory iterations done before the $\underline{u}$ vectors are saved for an extrapolation. Similarily. "No. of Prep. Extrap." means the number of preparatory extrapolations performed before the $\underline{v}$ vectors are saved for a super extrapolation. NA means "not applicable", and indicates that super extrapolation was not used in that particular instance.

TABLE III

GAUSS-SEIDEL WITH SDM EXTRAPOLATION

No. of Prep. Iter.	Period Of Iter.	No. of Prep. Extrap.	$-\frac{1}{m}$
0	1	NA	4.94
1	1	NA	6.90
2	1	NA	7.26
1	2	NA	8.11
0	1	2	4.93
2	1	1	4.73
1	2	1	6.57

The slopes in Table III, and the ones which follow, are computed from equation (3.45) below.

$$m = \frac{P_b - P_a}{b-a} \quad , \tag{3.45}$$

where a and b are chosen to be iterations near 25 and 50 respectively. For $-\frac{1}{m}$ to indicate reliably how the iterations were converging, a and b were chosen to be iterations where an extrapolation had just occurred. This is illustrated in Figure 13.

It should be mentioned again that in the author's implementation, the error, P, is computed during an iteration. So, the slope in Figure 13 would be computed from two points on the dashed line.

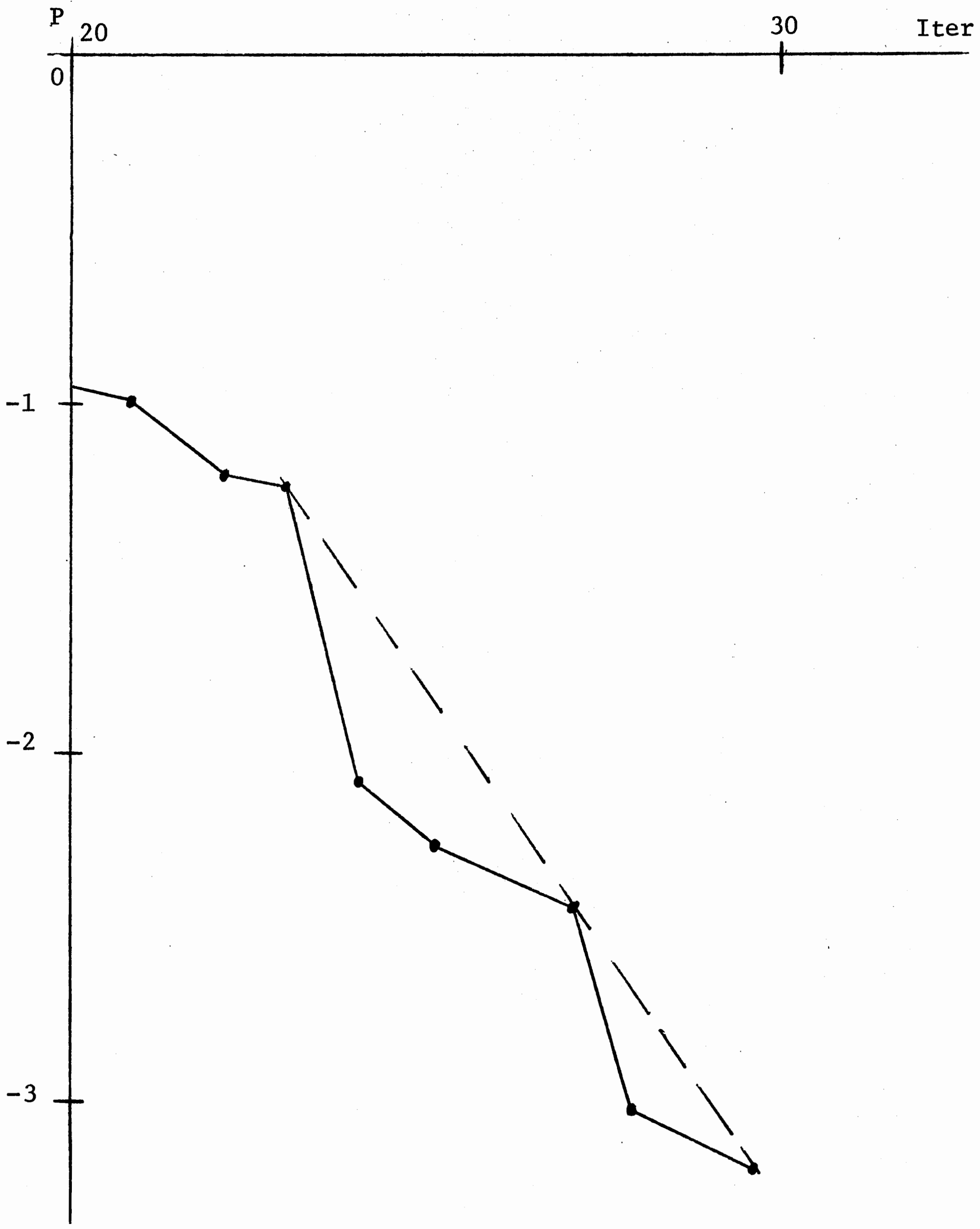

Figure 13. Gauss-Seidel, SDM Extrapolation After Every 2 Iterations

Referring to Table III again, SDM extrapolation after every two iterations improved accuracy by one digit about every five iterations. However, performing an extrapolation every four iterations worked best when super extrapolation was applied. The reason is that a clear extrapolation period of two was established in this case, thus making super extrapolation more effective.

Hence it appears that super extrapolation does indeed provide an alternative to SOR. Table I showed that SOR produced one digit of accuracy in slightly under four iterations, and Table III shows that super extrapolation requires a little over four iterations to do the same.

Table IV outlines the results using the other version of extrapolation, FDM, when applied to Gauss-Seidel. An iteration period of one and an extrapolation period of two are used.

TABLE IV

GAUSS-SEIDEL WITH FDM EXTRAPOLATION

No. of Prep. Iter.	No. of Prep. Extrap.	$-\frac{1}{m}$
0	NA	5.90
1	NA	6.21
2	NA	7.18
2	2	5.11

Table IV shows that the FDM extrapolation performed fairly well. However, when observing the error criterion, P, after some extrapolations were performed, it could be seen that the results were erratic. At times, an extrapolation with a large s-value would degrade the convergence, which is contrary to the previous results and the results to be shown later. The reason for this behavior comes from the fact that the iteration matrix for Gauss-Seidel is unsymmetric and Jennings (8) states that FDM extrapolation should only work well for symmetric iteration matrices. The $-\frac{1}{m}$ values in Table IV are thus perhaps slightly low in this case.

Table V outlines the results of applying SDM extrapolation to the Jacobi method. The iteration and extrapolation periods are given in the table.

TABLE V

JACOBI WITH SDM EXTRAPOLATION

No. of Prep. Iter.	Period of Iter.	No. of Prep. Extrap.	Period of Extrap.	$-\frac{1}{m}$
1	1	NA	NA	45.64
0	2	NA	NA	15.36
1	2	NA	NA	14.70
0	2	2	2	8.75
1	2	1	2	6.60
1	2	1	1	10.39

Table V clearly shows that the iteration and extrapolation periods for Jacobi must be two. This was expected since M_J is symmetric with eigenvalues in the range -1 to 1. Super extrapolation performed well, requiring slightly over six iterations to produce one digit of accuracy.

Table VI illustrates the results of applying the FDM extrapolation to the Jacobi method. The iteration period and extrapolation period were both set to two.

TABLE VI

JACOBI WITH FDM EXTRAPOLATION

No. of Prep. Iter.	No. of Prep. Extrap.	$-\frac{1}{m}$
0	NA	15.27
1	NA	13.76
2	NA	13.47
3	NA	14.20
4	NA	14.75
0	2	9.18
1	1	11.58
1	2	8.12
2	1	9.79
2	2	7.07

Table VI shows that FDM extrapolation performed better than SDM extrapolation when super extrapolation was not used. The reason here is probably the fact that FDM extrapolation works particularly well when applied to iterative methods which have symmetric iteration matrices (8). However, Table V shows that super extrapolation using SDM when applied to Jacobi produced slightly more accurate results than super extrapolation using FDM. It is thus the author's opinion that extrapolation with SDM should be used in most cases.

Hence, extrapolation does perform effectively on the Dirichlet problem. When super extrapolation is used convergence is almost equal to that of SOR. Chandler (3) in some work with linear least squares problems found an even greater improvement in convergence when using super extrapolation.

Each of these methods has its own disadvantages. Using SOR, the optimum w must be computed. Using super extrapolation, the period of the iterations, and the period of the original extrapolations must be determined. Also, additional storage is used. However, it is usually easier to extrapolate correctly than to compute the best w. For example, we know that if Jacobi is extrapolated the iteration period is probably two and if Gauss-Seidel is extrapolated the iteration period is probably one. An when super extrapolation is applied, the extrapolation period is probably two. In other words, the parameters for extrapolation are independent of n (the number of equations) and the optimum w for SOR is not. This point is shown conclusively in Chapter V.

One might wonder if it is possible to extrapolate the SOR method, accelerating convergence further. Unfortunately, when $w > 1$, the eigenvalues of M_{SOR} become complex and when $w = w_b$ all of the eigenvalues are complex (14,17). When this is the case, a zigzagging pattern cannot be determined in fewer than n iterations, and hence extrapolation becomes ineffective. Experimentally, extrapolation of SOR using the optimum w produces negative s-values and did not improve convergence.

Using SOR with w not equal to w_b, extrapolation may possibly be done (17), although the extrapolations will cause less improvement because of the small positive s-values. Even in this case, some of the s-values produced may be negative which means that, in our case, convergence would not be improved. Table VII outlines the results obtained when extrapolating SOR, using an iteration period of one. Super extrapolation, when used, is applied using two preparatory extrapolations and a period of two.

As w approached w_b, extrapolation of SOR degraded to unextrapolated SOR as expected. Super extrapolation was ineffective in all cases. One interesting result in Table VII is that when w = 1.2, SOR with extrapolation slightly outperformed unextrapolated SOR with w_b. It is not expected that this would always be the case, although little theory is available in this area.

TABLE VII

SOR WITH SDM EXTRAPOLATION

w	No. of Prep. Iter.	Super Extrap.	$-\frac{1}{m}$
1.1	1	NO	5.78
	2	NO	5.85
	2	YES	4.47
1.2	1	NO	3.30
	2	NO	3.09
1.3	1	NO	4.52
	2	NO	3.61
1.4	1	NO	4.84
	2	NO	4.28
1.5	1	NO	3.86
	2	NO	3.56

The Symmetric Successive Overrelaxation Method

A less widely known method recently reexamined by Young (18,19,20) is the symmetric successive overrelaxation (SSOR) method. The SSOR method can be considred as two half iterations. The first half iteration is the same as SOR. The second half iteration is the SOR method with the equations evaluated in reverse order, or backwards reading order, from right to left and bottom to top of the mesh.

The major advantage of the SSOR method is that it may be accelerated by extrapolation even when the optimum w is used. In fact, if the coefficient matrix, A, is positive definite and if $0 < w < 2$, then the eigenvalues of the iteration matrix, M_{SSOR}, are real, nonnegative, and less than one, thus making extrapolation possible (18).

The iterative form for computing u(x,y) at the interior points in the mesh is given below.

$$u^{(i+\frac{1}{2})}(x,y) = u^{(i)}(x,y) + w[\tfrac{1}{4}(u^{(i)}(x+h,y) + u^{(i+\frac{1}{2})}(x,y+h) + u^{(i+\frac{1}{2})}(x-h,y) + u^{(i)}(x,y-h)) - u^{(i)}(x,y)] \tag{3.46}$$

and

$$u^{(i+1)}(x,y) = u^{(i+\frac{1}{2})}(x,y) + w[\tfrac{1}{4}(u^{(i+1)}(x+h,y) + u^{(i+\frac{1}{2})}(x,y+h) + u^{(i+\frac{1}{2})}(x-h,y) + u^{(i+1)}(x,y-h)) - u^{(i+\frac{1}{2})}(x,y)] \tag{3.47}$$

In matrix form, we have (19)

$$\begin{aligned} \underline{u}^{(i+\frac{1}{2})} &= w(L\underline{u}^{(i+\frac{1}{2})} + U\underline{u}^{(i)} + \underline{c}) + (1-w)\underline{u}^{(i)} \\ \underline{u}^{(i+1)} &= w(L\underline{u}^{(i+\frac{1}{2})} + U\underline{u}^{(i+1)} + \underline{c})\ (1-w)\underline{u}^{(i+\frac{1}{2})} \end{aligned} \tag{3.48}$$

Simplifying this may be reduced to

$$\begin{aligned} \underline{u}^{(i+\frac{1}{2})} &= (I-wL)^{-1}[(wU + (1-w)I)\underline{u}^{(i)} + w\underline{c}] \\ \underline{u}^{(i+1)} &= (I-wU)^{-1}[(wL + (1-w)I)\underline{u}^{(i+\frac{1}{2})} + w\underline{c}] \end{aligned} \tag{3.49}$$

Eliminating $\underline{u}^{(i+\frac{1}{2})}$ gives

$$\underline{u}^{(i+1)} = M_{SSOR}\underline{u}^{(i)} + \underline{k}\ , \tag{3.50}$$

where

$$M_{SSOR} = (I-wU)^{-1}(wL + (1-w)I)(I-wL)^{-1}(wU + (1-w)I) \tag{3.51}$$

and

$$\underline{k} = w(2-w)(I-wU)^{-1}(I-wL)^{-1}\underline{c}$$

The method for choosing the optimum w for SSOR is similar to the method used for SOR. The method described by equations (3.20) and (3.21) was used by the author, but the value of the computed w was found not to be the optimum. Experimentally, the optimum w was found to be about 1.6, slightly larger than w_b for SOR. This is similar to some of Young's (17) results.

The results obtained using SSOR are displayed in Table VIII, when using SDM extrapolation. It should be noted that since one iteration of SSOR is almost as time consuming as two iterations of SOR, the results shown by the author count each sweep of the mesh as a full iteration. Therefore, reasonable comparisons can be made with the other methods discussed. When extrapolation is applied, the iteration period is one.

TABLE VIII

SSOR, w = 1.6, WITH SDM EXTRAPOLATION

No. of Prep. Iter.	No. of Prep. Extrap.	Period of Extrap.	$-\frac{1}{m}$
NA	NA	NA	10.66
0	NA	NA	4.88
1	NA	NA	4.96
0	0	1	4.26
0	1	1	4.22
1	1	2	4.22

The slopes, when using the SSOR method, are computed from the values of P, in the same manner as described earlier. Table VIII shows that using unextrapolated SSOR, with $w = w_b$, required over ten iterations to produce one digit of accuracy. This is not as good as SOR with its optimum w, but Young (17) describes similar results.

The value of SSOR becomes apparent when it is extrapolated. Using SDM extrapolation, SSOR required only about four iterations to produce one digit of accuracy. Figure 14 displays graphically P vs Iter for SSOR with and without super extrapolation. It can be seen that SSOR with super extrapolation (SSOR-SE) converges much faster than unextrapolated SSOR.

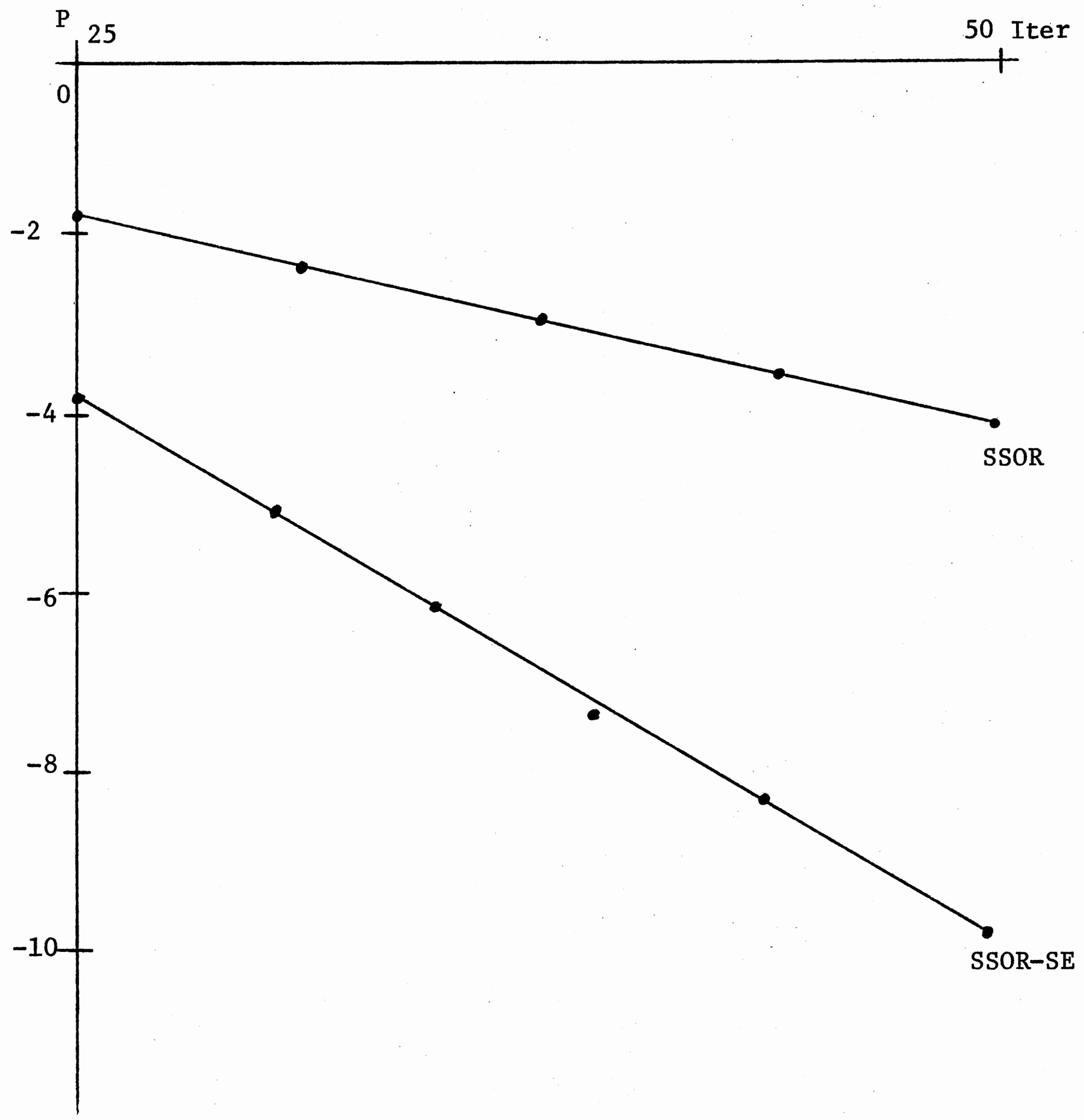

Figure 14. P vs Iter for SSOR

Table IX outlines the results obtained when applying FDM extrapolation to SSOR. An iteration period of one is used.

TABLE IX

SSOR, w = 1.6, WITH FDM EXTRAPOLATION

No. of Prep. Iter.	No. of Prep. Extrap.	Period of Extrap.	$-\frac{1}{m}$
0	NA	NA	4.10
1	NA	NA	4.92
2	NA	NA	4.62
0	2	2	4.22
1	1	1	8.00
2	1	1	6.02

Table IX shows that FDM extrapolation of SSOR also performed reasonably well, usually requiring about four iterations to produce one digit of accuracy.

Thus, SSOR does provide an accurate method for solving the Dirichlet problem. However, since it has the advantages of both SOR and extrapolation, it also has the disadvantages of both methods. To use SSOR effectively, the optimum w or near optimum w must be determined, along with the iteration and extrapolation periods when super extrapolation is used. Super extrapolation would probably not be required when using SSOR on meshes containing fewer than 100 interior points, since regular extrapolation accelerates convergence almost as fast.

Before leaving the Dirichlet problem it is worth mentioning a variation of vector extrapolation tried by the author. Instead of using s for our extrapolation factor we may use a value called S_{min} which is computed using <u>quadratic interpolation</u>. Quadratic interpolation is used to fit a quadratic equation through three points. The minimum of that quadratic then may be found analytically.

In our case, we want to determine the equation of a parabola which passes through the three points $(-1,Q)$, $(0,Q_2)$ and (s,Q_s), where the Q's are computed from equation (3.53) below.

$$Q = \sum_{k=1}^{n} (R^{(k)}(x,y))^2 \quad , \tag{3.53}$$

where n is the number of equations, and $R^{(k)}$ is the residual. Q_1 and Q_2 are computed after an appropriate iteration, and Q_s is computed after a normal extrapolation. The minimum of this parabola will be some point (S_{min}, Q_{smin}), where Q_{smin} is less than Q_s. That is, Q_{smin} is smaller at S_{min} than anywhere else along the direction of extrapolation $(u + s\cdot\Delta\underline{u})$.

Let equation (3.54) be the equation of our parabola.

$$ax^2 + bx + c = Q \tag{3.54}$$

To find the minimum of (3.54) we take the derivative and set it equal to zero, which gives

$$S_{min} = \frac{-b}{2a} \quad . \tag{3.55}$$

Substituting the values -1, 0, and s for x into equation (3.54) and solving the resulting system gives the following values for a and b.

$$a = \frac{s \cdot Q_1 + Q_s - s \cdot Q_2 - Q_2}{s^2 + s} \tag{3.56}$$

$$b = a - Q_1 + Q_2 \tag{3.57}$$

Thus, S_{min} may be determined from (3.56) and (3.57). In actual practice the values of s and S_{min} differed only slightly, and hence this method was not considered further. Although this method was not used, it does indicate that vector Aitken extrapolation is probably performing nearly as well as possible, in the sense of minimizing Q along the direction of extrapolation.

Before concluding this chapter, the validity of our error estimates, and a direct comparison of the results of the author's model problem and Young's model problem need to be shown. Recall that Young's (17) model problem used the square mesh in Figure 4, at the beginning of this chapter, with various h values. f(x,y) = 0 was used to define u(x,y) on the boundary, and the u values in the interior of the mesh were initialized to one. Since the boundary is set to zero, the actual solution will be $\underline{u} = \underline{0}$. Thus, the $\underline{u}$ vector will not only be the solution vector, but also the actual error vector.

So, to show that the $-\frac{1}{m}$ values given previously are valid convergence indicators, Young's model problem was implemented using $h = \frac{1}{5}$ and $h = \frac{1}{10}$. When $h = \frac{1}{5}$, the mesh contained a total of 16 interior points, and when $h = \frac{1}{10}$, the mesh contained 81 interior points. Since the $\underline{u}$ vector in this case is the actual error vector, we may compute the value of P for it as we did for the residual vector. Equation (3.58) is the equation for this value which we will call P^u.

$$P^u = \log_{10} \left(\sum_{k=1}^{n} U_k^2 \right)^{\frac{1}{2}} , \tag{3.58}$$

where n is the number of equations. Thus P^u is the $\log_{10}$ of the L_2 norm of u. We may compute the slope, m_u, of the line P^u vs Iter in the same way m was computed earlier.

$$m_u = \frac{P^u_{50} - P^u_{25}}{50 - 25} \tag{3.59}$$

Table X give the results of the author's implementation of Young's model problem for $h = \frac{1}{5}$ and $h = \frac{1}{10}$ using Jacobi and Gauss-Seidel.

TABLE X

YOUNG'S MODEL PROBLEM

Method	h	$-\frac{1}{m}$	$-\frac{1}{m_u}$
Jacobi	$\frac{1}{5}$	10.86	10.86
Jacobi	$\frac{1}{10}$	45.88	45.88
Gauss-Seidel	$\frac{1}{5}$	5.43	5.43
Gauss-Seidel	$\frac{1}{10}$	23.04	22.98

The values given for $-\frac{1}{m}$ and $-\frac{1}{m_u}$ in Table X are very close indeed. This indicates that computing $-\frac{1}{m}$ from the residual vector, using P, gives a very good estimate of convergence. It should also be noted that the $-\frac{1}{m}$ values for Jacobi and Gauss-Seidel with $h = \frac{1}{10}$ are very close to the $-\frac{1}{m}$ values for these methods in Table I with $h = \frac{1}{10}$. The reason for this is that the behavior of the error is independent of the boundary values as long as the boundary value function, f(x,y), is well-behaved.

Also, it can be seen in Table X that when h is halved, from $\frac{1}{5}$ to $\frac{1}{10}$, the number of iterations required for convergence using either Gauss-Seidel or Jacobi increases by a factor of about 4. In other words, the number of iterations required for convergence is proportional to h^{-2}. Table XI summarizes these results for Young's model problem (19).

TABLE XI

POWERS OF h FOR YOUNGS'S MODEL PROBLEM

Method	Order of the No. of iter. required
Jacobi	h^{-2}
Jacobi-SI	h^{-1}
Gauss-Seidel	h^{-2}
Gauss-Seidel-SI	h^{-1}
SOR	h^{-1}
SSOR	h^{-1}
SSOR-SI	$h^{-\frac{1}{2}}$

SI: semi-iterative extrapolation

Table XI showed that the h^{-2} value for Jacobi and Gauss-Seidel increases from 25 to 100 as h decreases from $\frac{1}{5}$ to $\frac{1}{10}$. The acceleration techniques mentioned in Table XI are various semi-iterative methods used by Young (17).

The results of the author's model problem should compare reasonably with Young's results in Table XI. To begin, we know that the following is ture.

$$N = M\cdot(-\frac{1}{m}) \quad , \tag{3.60}$$

where N is the number of iterations required to produce M digits of accuracy, and $-\frac{1}{m}$ is the number of iterations required to produce one digit of accuracy. N may also be given by:

$$N \doteq C\cdot h^{a} \quad , \tag{3.61}$$

where $a < 0$, as in Table XI. Thus, using $h = \frac{1}{10}$ and $h = \frac{1}{20}$, we may use equations (3.60) and (3.61) to form the following system.

$$\begin{aligned} M\cdot[-\frac{1}{m}(h=\frac{1}{10})] &= C\cdot(\frac{1}{10})^{a} \\ M\cdot[-\frac{1}{m}(h=\frac{1}{20})] &= C\cdot(\frac{1}{20})^{a} \end{aligned} \tag{3.62}$$

Solving for a gives

$$a = \log_2 \frac{-\frac{1}{m}(h=\frac{1}{10})}{-\frac{1}{m}(h=\frac{1}{20})} \tag{3.63}$$

So, to compare the results of our model problem with Young's results in Table XI, we need to compute a using the $-\frac{1}{m}$ values when $h = \frac{1}{10}$ and $h = \frac{1}{20}$. Table XII summarizes the results for all of our methods, including SDM extrapolation, for $h = \frac{1}{10}$ and $h = \frac{1}{20}$. The results shown were obtained using the optimum parameters. The optimum w for SOR was determined from equations (3.20) and (3.21) to be about 1.730249, and the optimum w for SSOR was experimentally determined to be about 1.75, when $h = \frac{1}{20}$.

The $-\frac{1}{m}(h = \frac{1}{10})$ values in Table XII were obtained from the previous tables for the various methods when $h = \frac{1}{10}$. The $-\frac{1}{m}(h = \frac{1}{20})$ values were computed in the same manner, with the exception of the Jacobi method. When $h = \frac{1}{20}$, our mesh contained 361 interior points where the value of u(x,y) was initially set to zero. Since the Jacobi method is so slow, additional iterations are required to replace these zeros by appropriate approximations. So, the values of P at iterations 75 and 100 were used to compute $-\frac{1}{m}(h = \frac{1}{20})$ in this case.

TABLE XII

SUMMARY OF THE DIRICHLET PROBLEM

Method	Extrap.	Super Extrap.	$-\frac{1}{m}(h=\frac{1}{10})$	$-\frac{1}{m}(h=\frac{1}{20})$	a
Jacobi	NO	NO	45.58	177.04	-1.96
Jacobi	YES	NO	14.70	49.75	-1.76
Jacobi	YES	YES	6.60	14.27	-1.11
Gauss-Seidel	NO	NO	23.04	84.20	-1.87
Gauss-Seidel	YES	NO	4.94	13.98	-1.50
Gauss-Seidel	YES	YES	4.73	11.30	-1.26
SOR	NO	NO	3.94	7.66	- .96
SSOR	NO	NO	10.66	21.90	-1.04
SSOR	YES	NO	4.88	7.98	- .71
SSOR	YES	YES	4.16	5.65	- .44

Hence, most of the values of a in Table XII compare favorably with the exponents of h in Table XI. Negative a-values nearest to zero indicate the fastest convergence. For example, when h is reduced from $\frac{1}{10}$ to $\frac{1}{20}$, SOR requires about 2 times as many iterations to converge. Similarly, SSOR with super extrapolation requires only about $\sqrt{2}$ times an many iterations to converge when h is reduced by ½. It appears that SSOR with super extrapolation clearly outperforms the other methods when the number of equations is increased. Comparing the a-values in Tables XI and XII, shows that SSOR-SE and SSOR-SI produce similar results.

Shortly and Weller (11) show that the error function that results from computing u, is "pillow-shaped," with a maximum at the center of the mesh. If the interior of Young's model mesh was initialized to a pillow-shaped function of the form:

$$r(x,y) = \frac{16}{x_{max}^2 \cdot y_{max}^2} \cdot x(x_{max}-x)y(y_{max}-y) \quad , \qquad (3.64)$$

where x_{max} and y_{max} are equal to one, the slopes during the early iterations would stabilize much faster. In other words, overrelaxation is best demonstrated on a pillow-shaped function.

This concludes our examination of the Dirichlet problem and we will now turn our attention to a more difficult and practical problem, the diffused semiconductor resistor.

CHAPTER IV

INTRODUCTION TO THE DIFFUSED RESISTOR PROBLEM

Diffused resistors are used extensively in integrated circuits, but actually there exists little theoretical information on their electrical properties. A typical two-dimensional rectangular resistor is shown in Figure 15.

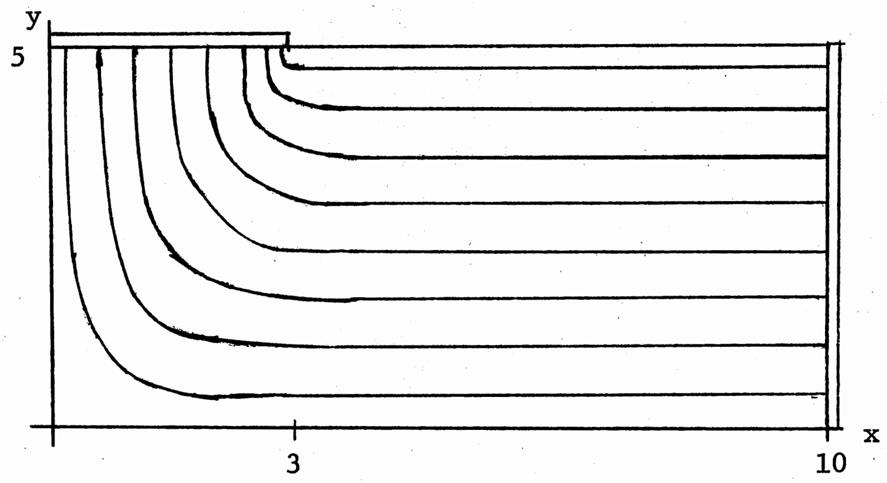

Figure 15. A Two Dimensional Diffused Resistor

Electrical conduction between the two metallic contacts takes place through a semiconducting material in the interior of the resistor. This semiconducting material may be composed of silicon, along with some small impurities, such as atoms of boron or iron (9).

The lines of current flow, shown in Figure 15, are uniformly spaced near the right contact and become crowded together near the right end of the top contact. They are approximately parallel to the length of the resistor further away from the top contact, and are perpendicular to each contact. When the conductivity within the resistor is nonconstant, the lines of current flow are changed and the numerical solution can become more complex.

These lines of current flow in the diffused resistor are somewhat similar to the electric field lines of a certain Dirichlet problem involving an L-shaped region. Figure 16 depicts this situation. The crowding shown in Figure 16 near the right angle bend is similar to the crowding observed in Figure 15 for the diffused resistor. Such a "re-entrant angle" (obtuse internal angle) is known to introduce a mild singularity into the solution of the Dirichlet problem (5), which increases the order of the discretization error. The same sort of thing occurs in the diffused resistor problem.

The operation of the diffused resistor can be approximated by a boundary value problem. Its solution will give us the electric potential at particular points within the resistor which result from applying a potential difference to the contacts. Each contact is assumed to extend across the entire width of the resistor; therefore it can be approximated by a two-dimensional analytical model (9). The two-

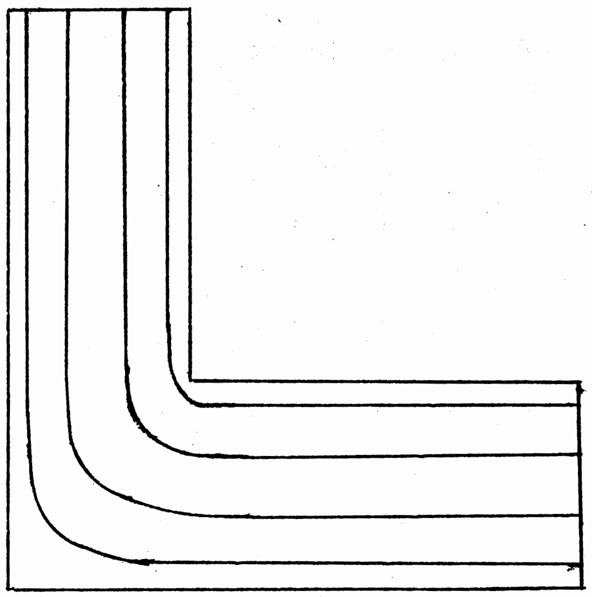

Figure 16. L-Shaped Region

dimensional elliptic partial differential equation used is given below.

$$g(x,y)\cdot u_{xx}(x,y) + g(x,y)\cdot u_{yy}(x,y) + g_x(x,y)\cdot u_x(x,y) + g_y(x,y)\cdot u_y(x,y) \doteq 0 \qquad (4.1)$$

where g(x,y) is the conductivity. Equation (4.1) is an approximate model for a certain case where no net charge exists anywhere in the semiconducting material. Note that the conductivity, g(x,y), is a function of position.

The finite difference approximations to equation (4.1) may be derived in much the same way as for the Dirichlet problem. The derivations given below assume unequal h's for generality. Letting $u_1 = u(x+h_1,y)$, $u_2 = u(x,y+h_2)$, $u_3 = u(x-h_3,y)$, and $u_4 = u(x,y-h_4)$, and expanding about the point $u_0 = u(x,y)$ using Taylor's series as before gives:

$$u_1 = u_0 + h_1u_x + \tfrac{1}{2}h_1^2u_{xx} + 0(h_1^3) \tag{4.2}$$
$$u_2 = u_0 + h_2u_y + \tfrac{1}{2}h_2^2u_{yy} + 0(h_2^3) \tag{4.3}$$
$$u_3 = u_0 - h_3u_x + \tfrac{1}{2}h_3^2u_{xx} + 0(h_3^3) \tag{4.4}$$
$$u_4 = u_0 - h_4u_y + \tfrac{1}{2}h_4^2u_{yy} + 0(h_4^3) \tag{4.5}$$

Adding (4.2) and (4.4) and simplifying gives

$$u_{xx} \doteq \frac{2u_1}{h_1(h_1+h_3)} + \frac{2u_3}{h_3(h_1+h_3)} - \frac{2u_0}{h_1h_3} \tag{4.6}$$

Adding (4.3) and (4.5) and simplifying gives

$$u_{yy} \doteq \frac{2u_2}{h_2(h_2+h_4)} + \frac{2u_4}{h_4(h_2+h_4)} - \frac{2u_0}{h_2h_4} \tag{4.7}$$

Solving for u_x using (4.2) and (4.4) gives

$$u_x \doteq \frac{h_1^2(u_0-u_3) + h_3^2(u_1-u_0)}{h_1h_3^2 + h_1^2h_3} \tag{4.8}$$

Solving for u_y using (4.3) and (4.5) gives

$$u_y \doteq \frac{h_2^2(u_0-u_4) + h_4^2(u_2-u_0)}{h_2h_4^2 + h_2^2h_4} \tag{4.9}$$

Thus, finite difference approximation to equation (4.1) is

$$g\left[\frac{2u_1}{h_1(h_1+h_3)} + \frac{2u_3}{h_3(h_1+h_3)} - \frac{2u_0}{h_1h_3} + \frac{2u_2}{h_1(h_1+h_4)} + \frac{2u_4}{h_4(h_2+h_4)} - \frac{2u_0}{h_2h_4}\right]$$
$$+ g_x\left[\frac{h_1^2(u_0-u_3)+h_3^2(u_1-u_0)}{h_1h_3^2 + h_1^2h_3}\right] + g_y\left[\frac{h_2^2(u_0-u_4)+h_4^2(u_2-u_0)}{h_2h_4^2 + h_2^2h_4}\right] \doteq 0 \tag{4.10}$$

Equation (4.10) can be transformed to the following:

$$u_0 \doteq - \frac{h_1h_2h_3h_4}{(g_x(h_1-h_3)-2g)h_2h_4+(g_y(h_2-h_4)-2g)h_1h_3}\left[\frac{(2g+g_xh_3)u_1}{h_1(h_1+h_3)} + \frac{(2g+g_yh_4)}{h_2(h_2+h_4)}u_2 + \frac{(2g-g_xh_1)}{h_3(h_1+h_3)}u_3 + \frac{(2g-g_yh_2)}{h_4(h_2+h_4)}u_4\right] \quad (4.11)$$

Equation (4.11) is the five-point rule for the diffused resistor problem, using unequal h's. In the case where $h_1 = h_2 = h_3 = h_4 = h$, equation (4.11) may be reduced to:

$$u_0 \doteq \frac{1}{8g}[(2g+g_xh)u_1 + (2g+g_yh)u_2 + (2g-g_xh)u_3 + (2g-g_yh)u_4] \quad (4.12)$$

So the Jacobi, Gauss-Seidel, and SOR methods may be given by equations (4.13), (4.14), and (4.15), respectively, using fixed h's.

$$u^{(i+1)}(x,y) = \frac{1}{8g}[(2g+g_xh)\,u^{(i)}(x+h,y) + (2g+g_yh)\,u^{(i)}(x,y+h) +(2g-g_xh)\,u^{(i)}(x-h,y) + (2g-g_yh)u^{(i)}(x,y-h)] \quad (4.13)$$

$$u^{(i+1)}(x,y) = \frac{1}{8g}\ (2g+g_xh)\,u^{(i)}(x+h,y) + (2g+g_yh)u^{(i+1)}(x,y+h) +(2g-g_xh)\,u^{(i+1)}(x-h,y) + (2g-g_yh)u^{(i)}(x,y-h) \quad (4.14)$$

$$u^{(i+1)}(x,y) = u^{(i)}(x,y) + w\left[\frac{1}{8g}[(2g+g_xh)\,u^{(i)}(x+h,y) + (2g+g_yh)u^{(i+1)}(x,y+h) + (2g-g_xh)\,u^{(i+1)}(x-h,y) + (2g-g_yh)u^{(i)}(x,y-h)] - u^{(i)}(x,y)\right] \quad (4.15)$$

The JOR method is identical to equation (4.15) when the i+1 iterates are replaced by the i^{th} iterates.

The boundary conditions for the diffused resistor are somewhat different from those for the Dirichlet problem. Here, the only parts of the boundary where u(x,y) is given as f(x,y) are at the contacts. This situation is sometimes called <u>mixed boundary conditions</u> (7). The author used f(x,y) = 0 to define u(x,y) on the top contact and f(x,y) = 1 to define u(x,y) on the right contact.

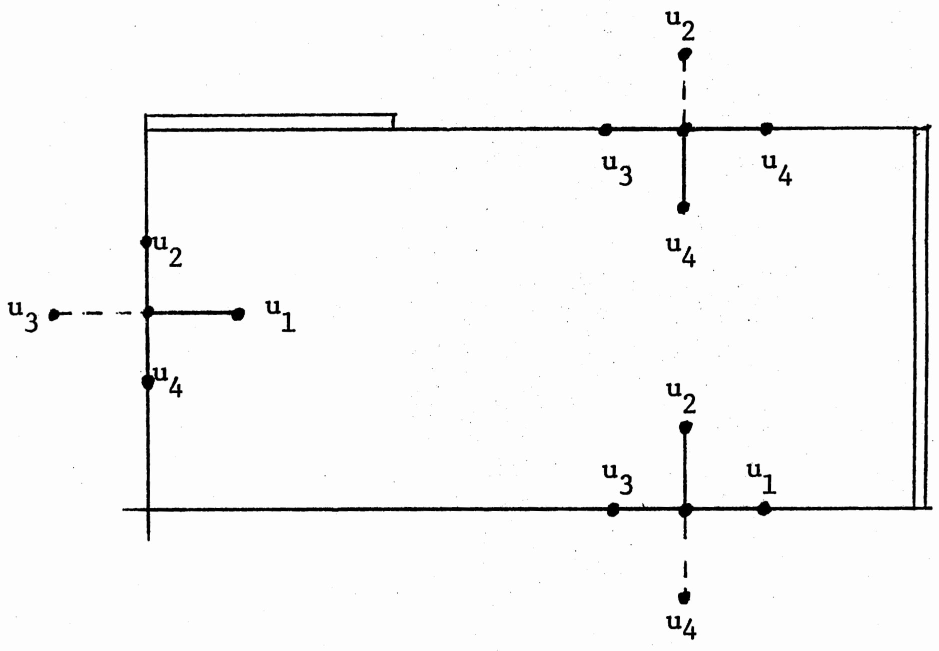

Figure 17. Mixed Boundary Conditions

Figure 17 illustrates how the five-point rule can be applied to portions of the boundary where a contact is not located. Considering the top boundary for a moment, we can see that u_2 is located outside the boundary of the resistor and hence cannot be used in equations (4.11) or (4.12). The potential gradient at the boundary is parallel to the boundary where the contact is not located, so $u_y = 0$. Using equation (4.9) derived earlier we may compute an approximate value for u_2 in terms of u_4.

Setting $u_y = 0$ in equation (4.9) gives

$$\frac{h_2^2(u_0-u_4) + h_4^2(u_2-u_0)}{h_2h_4^2 + h_2^2h_4} \doteq 0 \qquad (4.16)$$

Solving for u_2, we find that

$$u_2 \doteq \frac{-h_2^2}{h_4^2}(u_0-u_4) + u_0 \qquad (4.17)$$

Taking $h_2 = h_4$ we have

$$u_2 \doteq u_4 \tag{4.18}$$

Thus we can reflect u_2 across the boundary and use the value of u_4 for u_2 as shown in Figure 17.

A similar argument can be made for the bottom boundary, except that in this case we would reflect u_4 across the boundary. Similarly, at the left boundary, we have $u_x = 0$. Setting $u_x = 0$ in equation (4.8) gives

$$\frac{h_1^2(u_0-u_3) + h_3^2(u_1-u_0)}{h_1h_3^2 + h_1^2h_3} \doteq 0 \tag{4.19}$$

Solving for u_3 and taking $h_1 = h_3$ gives

$$u_3 \doteq u_1 \tag{4.20}$$

Thus, when computing the value of u_0 along the left boundary, we may reflect u_3 across the boundary and use the value of u_1 for u_3.

Convergence of the diffused resistor problem is not expected to be as fast as it was for the Dirichlet problem for the reason discussed below. It can be shown that the crowding of the lines of current flow in the diffused resistor is associated with a square root singularity in $\underline{u}$. In this area, near the right end of the top contact, u(x,y) may be given as:

$$u(x,y) \doteq C\cdot(\tfrac{1}{2}(x+r))^{\frac{1}{2}} \tag{4.21}$$

where

$$r = (x^2 + y^2)^{\frac{1}{2}} \tag{4.22}$$

and C is some constant. Equations (4.21) and (4.22) come from the use of a conformal mapping and their derivation in beyond the scope of this paper. A square root singularity in $\underline{u}$ also exists in the Dirichlet problem with an L-shaped region (5).

In an attempt to improve the accuracy in this area of crowding, variable h's instead of fixed h's were used. Additional mesh points were added near the area of crowding and the mesh points were spread out farther in other areas. One simple method to compute these variable h's is given below.

The h's in the x-direction from the right end of the top contact to the right contact may be defined by (4.23) below.

$$h_i = d_r b^i \tag{4.23}$$

Each h_i increases by a factor of b as the right contact is approached. If b is given, then d_r can be computed from (4.24).

$$\Sigma h_i = C_r \tag{4.24}$$

where C_r is the distance. Using Figure 15, $C_r = 7$ in our case.

Similarly, the h's in the x-direction from the right end of the top contact to the left boundary can be computed by (4.25) below.

$$h_i = d_L b^i \tag{4.25}$$

where

$$\Sigma h_i = C_y \tag{4.26}$$

And the h's in the y-direction from the top boundary to the bottom boundary can be computed by (4.27)

$$h_i = d_y b^i \tag{4.27}$$

where

$$\Sigma h_i = C_y \tag{4.28}$$

The convergence criterion used for the diffused resistor, problem is similar to that used for the Dirichlet problem. R, P, and G are given by the following equations:

$$R^{(i)}(x,y) = \frac{1}{8g}[(2g+g_x h)\ u^{(i)}(x+h,y) + (2g+g_y h)u^{(i+1)}(x,y+h) + (2g-g_x h)u^{(i+1)}(x-h,y) + (2g-g_y h)\ u^{(i)}(x,y-h)]-u^{(i)}(x,y) \tag{4.29}$$

where (i+1) is replaced by (i) when using Jacobi or JOR.

$$P = \log_{10}\left[\sum_{k=1}^{n}(R^{(k)}(x,y))^2\right]^{\frac{1}{2}} \tag{4.30}$$

and

$$G = \sum_{k=1}^{n}|R^{(k)}(x,y)| \tag{4.31}$$

where n is the number of difference equations.

CHAPTER V

THE NUMERICAL SOLUTION FO THE DIFFUSED RESISTOR PROBLEM

Numerical solution of the diffused resistor problem proved to be a more difficult task than solving the Dirichlet problem. Preliminary solutions used equation (5.1) for the conductivity of the rectangualar resistor in Figure 15.

$$g(x,y) = e^{0.2y} \tag{5.1}$$

The variable mesh scheme, outlined in Chapter IV was used initially. Figure 18 illustrates this further. Using SOR with w equal to 1.8, the results were satisfactory but convergence was slow. Computing the slope as described in Chapter III, using P_{50} and P_{25}, the value of $-\frac{1}{m}$ was found to be 27.19, which indicates that about 27 iterations were required to produce one digit of accuracy.

Obviously, these results are not as good as those obtained for the Dirichlet problem. The apparent reason for this is the fact that the mesh points became too spread out near the lower right and lower left of the resistor shown in Figure 18. Possibly a different method for computing the h's could be used, but this was not pursued further. The discussion presented next will use fixed h's.

Using h = 1, ½, and ¼, and the variable conductivity in equation (5.1), the results of the SOR method is shown in Table XIII.

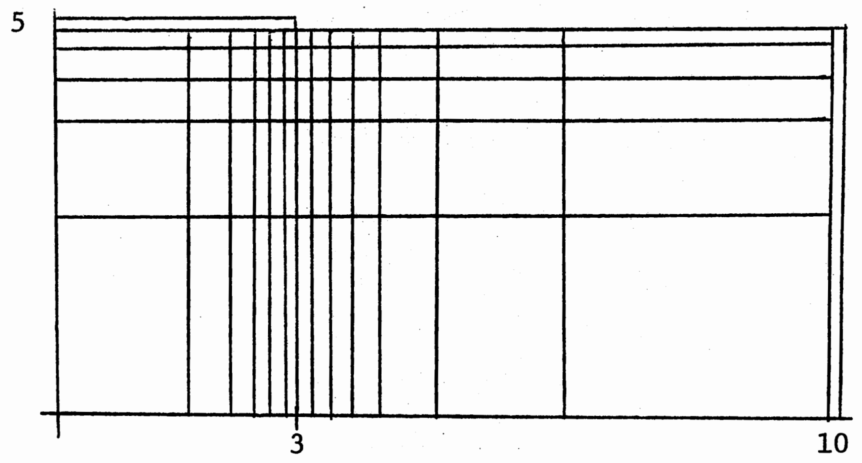

Figure 18. Variable Mesh

The optimum w shown in Table XIII when h = 1 and h = ½ was computed using Gauss-Seidel and equations (3.20) and (3.21) given in Chapter III. When h = ¼ (861 mesh points) this method proved ineffective. Forsythe and Wasow (5) cite this major disadvantage when computing the optimum w for fine meshes. This is due to the fact that d in (3.20) is computed from a limit. The results in Table XIII also seem to indicate that the value of the optimum w increases as the number of mesh points increases. This is shown by Young (17) and is analogous to our results of the Dirichlet problem.

TABLE XIII

SOR, VARIABLE CONDUCTIVITY

h	w	$-\frac{1}{m}$	a
1	$1.64478 = w_b$	4.97	
			-1.00
½	$1.80129 = w_b$	9.94	
			-0.87
¼	1.89	18.19	

The slopes in Table XIII have been computed using the methods of Chapter III, also. When h = 1, P_{50} and P_{25} were used to compute m. When h = ½, P_{75} and P_{50} were used, and when h = ¼, P_{100} and P_{75} were used to compute m.

The values of a shown in Table XIII were computed by equation (3.63) using the $-\frac{1}{m}$ values for h = 1 and h = ½, and for h = ½ and h = ¼. Young (17) has shown that $a \doteq -1$ when using SOR on the Dirichlet problem. It appears that for the diffused resistor problem, the number of iterations required for convergence is also proportional to h^{-1}, when using SOR. This is investigated further for the case when the conductivity is equal to one. Table XIV outlines the results when using the SOR method with g(x,y) = 1.

Table XIV shows that when the conductivity is constant, the results are similar to those in Table XIII when the conductivity is variable. The $-\frac{1}{m}$ values in Tables XIII and XIV for SOR are larger than the $-\frac{1}{m}$ value for SOR in Table I, even though the diffused resistor mesh contains fewer points. So, convergence of the diffused resistor problem

TABLE XIV

SOR, CONSTANT CONDUCTIVITY

h	w	$-\frac{1}{m}$	a
1	$1.65150 = w_b$	5.21	
			-1.04
½	$1.81625 = w_b$	10.69	
			-1.08
¼	1.905	22.66	

is probably slower than it is for the Dirichlet problem. It would appear that the mixed boundary conditions are probably more of an influence on convergence than the nonconstant conductivity. The singularity in <u>u</u> near the right end of the top contact is probably also influencing the convergence here.

Table XV outlines the results of the Gauss-Seidel, Jacobi, and JOR methods for the case where h = 1. The slopes shown here were

TABLE XV

GAUSS-SEIDEL, JACOBI, AND JOR, h = 1 AND g(x,y) = 1

Method	w	$-\frac{1}{m}$
Gauss-Seidel	1.00	50.28
Jacobi	1.00	99.39
JOR	.95	74.30

computed using P_{100} and P_{75}. Table XV shows that Gauss-Seidel, Jacobi, and JOR are all very slow methods even when h = 1. This was expected. Table XV also shows that Gauss-Seidel is about twice as fast as Jacobi. The same result was obtained by the author and others (12,14,15,17) for the Dirichlet problem.

A Note On Discretization Error

When forming the approximations for Laplace's equation and the diffused resistor equation, we replaced each partial differential equation by a system of n divided difference equations. This process is called discretization (5), and is only as accurate as the difference approximations used. The error which occurs during the process of discretization, is termed the discretization error (5). In Chapters II and III, we showed that this error was $O(h^2)$ for Laplace's equation when using the five-point rule.

We may approximate the discretization error of equation (4.12) for the diffused resistor as follows. Using Taylor's series, the following equation may be used to compute this error.

$$u(h) = u(0) + C\cdot h^p + \ldots \qquad (5.2)$$

where u(h) is the value of u(x,y) at a particular mesh point for some h, and u(0) is the value of u(x,y) at that point when h = 0. Using the values of u(1), u(½), and u(¼) the following system can be formed.

$$\begin{aligned} u(1) &\doteq u(0) + C\cdot 1^p \\ u(\tfrac{1}{2}) &\doteq u(0) + C\cdot(\tfrac{1}{2})^p \\ u(\tfrac{1}{4}) &\doteq u(0) + C\cdot(\tfrac{1}{4})^p \end{aligned} \qquad (5.3)$$

This system has the following solution:

$$\frac{u(1)-u(\tfrac{1}{2})}{u(\tfrac{1}{2})-u(\tfrac{1}{4})} \doteq 2^p \qquad (5.4)$$

Using the values of u(x,y) at the mesh point (x,y) = (8,2) obtained from the SOR method, we may compute p from the equation below.

$$2^p = \frac{.7864-.7910}{.7910-.7934} \tag{5.5}$$

$$p = .997 \tag{5.6}$$

Since p = 1, the discretization error appears to be $O(h^1)$ for the diffused resistor. Performing similar computations at the mesh point (x,y) = (3,4), which is near the area of "crowding", $p \doteq .995$. The singularity in $\underline{u}$ is probably the cause for this slower order of convergence of the discretization error to zero. Forsythe and Wasow (5) show that the discretization error for the Dirichlet problem with a L-shaped region is also $O(h^1)$ as h approaches zero.

Vector Aitken Extrapolation of the Diffused Resistor Problem

Vector Aitken extrapolation and super extrapolation can be applied to the diffused resistor problem in the same manner as they were applied to the Dirichlet problem. The results from Chapter III proved to be useful when applying vector extrapolation to the diffused resistor.

The results of extrapolation of the Gauss-Seidel method are displayed in Table XVI. SDM extrapolation is used exclusively, since the results of the Dirichlet problem and Jennings' (8) work show that FDM extrapolation only works effectively on iteration methods which have symmetric iteration matrices. In Table XVI, h = 1, g(x,y) = 1, the iteration period is one, and the extrapolation period is two.

TABLE XVI

SDM EXTRAPOLATION OF GAUSS-SEIDEL
h = 1 AND g(x,y) = 1

No. of Prep. Iter.	No. of Prep. Extrap.	$-\frac{1}{m}$
0	NA	8.14
1	NA	4.68
2	NA	10.86
3	NA	11.03
2	1	8.11
2	2	8.84
3	1	9.62

The slopes in Table XVI, and in all the tables which contain the results of extrapolation, are computed from equation (3.45) in the same manner as discussed in Chapter III. P_a and P_b were chosen from iterations where extrapolation with a large s-value or super extrapolation occurred. In Table XVI, iterations near 25 and 50 were used. In later tables where Jacobi or large meshes were used, iterations near 100 and 50 were used for P_a and P_b, respectively.

Table XVI shows that extrapolation of Gauss-Seidel using one preparatory iteration converged the fastest. Between four and five iterations were required to produce one digit of accuracy. This result is even better than that obtained using SOR with the optimum w shown in Table XIV.

Performing one preparatory iteration corresponds to extrapolating after every 3 iterations. This result is different from the results shown by Boyle and Jennings (1). For their problem, they found that extrapolating after every two iterations (no preparatory iterations) was best.

Super extrapolation did not perform as well as expected. It will be shown later that when h = ¼, super extrapolation becomes more effective.

Table XVII outlines the results of SDM extrapolation of Jacobi. Here, h = 1, and g(x,y) = 1.

TABLE XVII

SDM EXTRAPOLATION OF JACOBI
h = 1 AND g(x,y) = 1

No. of Prep. Iter.	Period of Iter.	No. of Prep. Extrap.	Period of Extrap.	$-\frac{1}{m}$
1	1	NA	NA	99.39
1	2	NA	NA	15.91
2	2	NA	NA	27.22
3	2	NA	NA	28.86
2	2	1	2	15.06
3	2	1	1	32.44

Table XVII shows clearly that the iteration period for the Jacobi method is two. This reinforces the theory dealing with the eigenvalues and type of iteration matrix (8) and the zigzagging of the $\Delta \underline{u}$ vector developed in Chapter III. Extrapolation, using one preparatory iteration, again performed very well. Since our iteration period is two, this corresponds to extrapolating after every five iterations. Super extrapolation showed only minimal additional improvement, requiring about 15 iterations to produce one digit of accuracy.

Table XVIII illustrates the results of SDM extrapolation of the Gauss-Seidel method when $h = \frac{1}{2}$. When super extrapolation is used, the extrapolation period is two.

With $h = \frac{1}{2}$ and the mesh containing 231 points, extrapolation with two preparatory iterations and an iteration period of one, performed the best. Around 14 iterations were required to produce one digit of accuracy. This is slightly more iterations than SOR required.

Table XVIII also shows that when using almost every reasonable combination of periods and preparatory iterations, super extrapolation did not outperform regular extrapolation. It should be noted that, when super extrapolation was applied to the results of the best regular extrapolation, the number of iterations to produce one digit of accuracy increased from about 14 to 43. The reason for this is the s-values from the regular extrapolation were irregular and did not oscillate, so that super extrapolation was ineffective.

Table XIX outlines the results of SDM extrapolation of Gauss-Seidel with $h = \frac{1}{4}$. When super extrapolation is used, the period of extrapolation is two.

TABLE XVIII

SDM EXTRAPOLATION OF GAUSS-SEIDEL
h = ½ AND g(x,y) = 1

No. of Prep. Iter.	Period of Iter.	No. of Prep. Extrap.	$-\frac{1}{m}$
0	1	NA	18.49
1	1	NA	29.51
2	1	NA	14.51
3	1	NA	51.28
1	2	NA	53.53
1	1	1	19.28
1	1	2	19.73
2	1	1	43.59
2	1	2	42.56
3	1	2	24.81

Table XIX shows that super extrapolation performed the best when h = ¼. Since the mesh now contains 861 points, convergence has slowed considerably. Even when using super extrapolation, around 39 iterations are required for one digit of accuracy. Table XIV shows that this is somewhat slower than SOR with h = ¼.

SDM extrapolation of Gauss-Seidel is summarized in Table XX for h = 1, ½ and ¼. The values shown are the optimum values obtained from previous tables.

TABLE XIX

SDM EXTRAPOLATION OF GAUSS-SEIDEL
$h = \frac{1}{4}$ AND $g(x,y) = 1$

No. of Prep. Iter.	Period of Iter.	No. of Prep. Extrap.	$-\frac{1}{m}$
0	1	NA	91.47
1	1	NA	57.48
2	1	NA	44.28
1	2	NA	56.83
1	1	1	76.16
1	1	2	39.21
2	1	1	45.39
2	1	2	44.75

The values for a in Table XX were computed by the methods discussed in Chapter III. Table XX shows that these a-values for the diffused resistor are similar to those obtained for the Dirichlet problem in Table XII. The number of iterations required for convergence is approximately proporional to $O(h^{-1.6})$ using regular extrapolation and $O(h^{-1})$ using super extrapolation. So, as h is reduced by ½, extrapolation alone will require over 3 times as many iterations to converge, and super extrapolation will require around 2 times as many iterations to converge.

TABLE XX

SUMMARY OF SDM EXTRAPOLATION FOR THE DIFFUSED RESISTOR PROBLEM

h	Extrapolation	Super Extrapolation	$-\frac{1}{m}$	a
1	YES	NO	4.68	
				-1.63
½	YES	NO	14.51	
				-1.61
¼	YES	NO	44.28	
1	YES	YES	8.11	
				-1.25
½	YES	YES	19.28	
				-1.02
¼	YES	YES	39.21	

Comparisons and Conclusions

Now that we have analyzed and solved both the Dirichlet problem and the diffused resistor problem, some general comparisons and conclusions can be drawn.

The particular form of variable mesh used was ineffective in solving the diffused resistor problem. The variable h method may be feasible as long as the distance between two adjacent mesh points does not become too large, as it did here.

The SOR method worked well for both the Dirichlet and diffused resistor problems. For both problems, the optimum w was shown to increase as the number of mesh points increases. In other words, the value of the optimum w is dependent upon the number of finite difference equations to be solved. It also should be noted that the optimum w could not be computed from equations (3.20) and (3.21) when $h \leq \frac{1}{4}$ for the diffused resistor problem. The large number of mesh points, mixed boundary conditions, and singularity in $\underline{u}$ were probably all factors in this case.

Vector Aitken extrapolation worked well for both the Dirichlet problem and the diffused resistor problem. However, super extrapolation was not as effective on the diffused resistor problem. The reason for this is that in some cases, the s-values from the original extrapolation were not constant, or did not oscillate in a rhythmic pattern. So, when super extrapolation was applied, little improvement in convergence was shown.

Usually, extrapolation is easier to use than SOR. The iteration period and extrapolation period are not dependent upon the number of equations to be solved. The periods and the recommended type of extrapolation (SDM of FDM) are shown in Table XXI. The results shown are applicable to both problems.

The mixed boundary conditions of the diffused resistor made it impractical to apply the SSOR method. The forward/backward iterations of SOR in SSOR would almost double the amount of code required, and the results of the Dirichlet problem indicate that for meshes containing a small number of points, little additional improvement in convergence would be shown. However, the author's results and Young's (18,19,20) results indicate that SSOR with extrapolation show more improvement when applied to finer meshes; that is, meshes containing a larger number of points.

TABLE XXI

RECOMMENDED PARAMETERS FOR EXTRAPOLATION

Iteration Method	Type of Extrap.	Iter. Period	Extrap. Period
Jacobi	FDM or SDM	2	2
Gauss-Seidel	SDM	1	2
SSOR	SDM	1	2

CHAPTER VI

SUMMARY AND CONCLUSIONS

The numerical analysis and solution of the Dirichlet and diffused resistor problems have been completed with some interesting results. These results and the methods developed can be applied to most other boundary value problems in the elliptic class of partial differential equations.

The most likely method to be used for the numerical solution of the difference equations would be successive overrelaxation, if storage is at a premium. However, if adequate storage is available, modified vector Aitken extrapolation plus super extrapolation of either the Gauss-Seidel or Jacobi methods does indeed provide a feasible alternative. The advantages and disadvantages of both of these choices have been discussed previously. Table XXII presents a summary of the storage requirements for the methods which have been discussed in this paper. This table gives the maximum number of equations which can be solved, assuming 1000 storage locations are available for components of $\underline{u}$.

Table XXII shows that when using a direct elimination method, a maximum of 31 equations can be solved if the coefficient matrix is unsymmetric, since n^2 storage locations are required. For symmetric coefficient matrices, $\frac{1}{2}n(n-1)$ storage locations are required. Gauss-Seidel, SOR, and SSOR require only n locations when extrapolation is not used and either 3n or 5n locations when extrapolation or super extrapolation is used. Table XXII also shows that 2n storage locations are required by the Jacobi (or JOR) method. When using a square or rectangular mesh and applying the Jacobi method in reading order, only two rows of the ith mesh need to be stored when computing $\underline{u}^{(i+1)}$. In this case, only n+c storage locations would be required.

TABLE XXII

SUMMARY OF STORAGE REQUIREMENTS

Method(s)	Max. No. of Equations (n)
Direct Elimination	31 or 44*
GS,SOR,SSOR	1000
Jacobi, JOR	500
GS,SSOR+VA	333
Jacobi+VA	250
GS,SSOR+SE	200
Jacobi+SE	167

1000	- Storage Locations
*	- Symmetric coefficient matrix
VA	- Vector Aitken extrapolation
SE	- Super Extrapolation

The method recently reexamined by Young (18,19,20), symmetric successive overrelaxation, does appear to be promising when extrapolated. If storage is at a premium, it may be necessary to extrapolate SSOR using semi-iterative methods (17) rather than Aitken extrapolation. tion.

BIBLIOGRAPHY

(1) Boyle, E.F. and A. Jennings. "Accelerating the Convergence of Elastic-Plastic Stress Analysis." International Journal of Numerical Methods in Engineering, Vol. VII (1974), pp. 232-235.

(2) Brezinski, E. and A.E. Rieu. "The Solution of Systems of Equations tions Using the -Algorithm, and an Application to Boundary - Value Problems." Mathematics of Computation, Vol. XXVIII (1974), pp. 731-741.

(3) Chandler, J.P. Private Communication. Stillwater, Oklahoma: Oklahoma State University, 1976.

(4) Conte, S.D. and C. deBoor. Elementary Numerical Analysis: An Algorithmic Approach. New York: McGraw-Hill, 1972.

(5) Forsythe, G.E. and W.R. Wasow. Finite-Difference Methods for Partial Differential Equations. New York: John Wiley & Sons, Inc., 1960.

(6) Gekeler, E. "On the Solution of Equations by the Epsilon Algorithm of Wynn." Mathematics of Computation, Vol. XXVI (1972), pp. 427-436.

(7) Greenspan, D. Discrete Numerical Methods in Physics and Engineering. New York: Academic Press, 1974.

(8) Jennings, A. "Accelerating the Convergence of Matrix Iterative Processes," Journal of the Institute for Mathematics and Applications, Vol. VIII (1971), pp. 99-110.

(9) Kennedy, D.P. and P.E. Murley. "A Two-Dimensional Mathematic Analysis of the Diffused Semiconductor Resistor." IBM Journal of Research and Development, Vol. XII (1968), pp. 242-250.

(10) Panov, D.J. Formulas for the Numerical Solution of Partial Differential Equations by the Method of Differences. New York: Grederick Ungar Co., 1963.

(11) Shortley, G.H. and R. Weller. "The Numerical Solution of Laplace's Equation." Journal of Applied Physics, Vol. IX (1938), pp. 334-348.

(12) Smith, G.D. Numerical Solution of Partial Differential Equations. London: Oxford University Press, 1965.

(13) Stewart, G.W. Introduction to Matrix Computations. New York: Academic Press, 1973.

(14) Varga, R.S. Matrix Iterative Analysis. New Jersey: Prentice-Hall, Inc., 1962.

(15) Wachspress, E.Y. Iterative Solution of Elliptic Systems and Applications to the Neutron Diffusion Equations of Reactor Physics. New Jersey: Prentice-Hall, Inc., 1966.

(16) Wynn, P. "Acceleration Techniques for Iterated Vector and Matrix Problems." Mathematics of Computation, Vol. XVI (1962), pp. 301-322.

(17) Young, D.M Iterative Solution of Large Linear Systems. New York: Academic Press, 1971.

(18) Young, D.M. "On the Accelerated SSOR Method for Solving Elliptic Boundary Value Problems." Proceedings of the Conference on the Numerical Solution of Differential Equations. Dundee Scotland: Berlin, New York, Springer-Verlag, 1973.

(19) Young, D.M. "On the Accelerated SSOR Method for Solving Large Linear Systems." Paper CNA-92. Austin, Texas: The University of Texas, 1974.

(20) Young, D.M. "Iterative Solution of Linear and Nonlinear Systems Derived from Elliptic Partial Differential Equations." Proceedings of the International Conference on Computational Methods for Nonlinear Mechanics. Austin, Texas: Texas Institute for Computational Mechanics - The University of Texas, 1974.

(21) Zienkiewicz, O.C. and Y.K. Cheung. "Finite Elements in the Solution of Field Problems." The Engineer (Sept., 1965) pp. 507-511.

APPENDIX

PROGRAM LISTINGS OF VAITK AND DRIVER ROUTINE

```
      SUBROUTINE VAITK (N,X,TA,TB,METHD,KOUNT,NPREP,NPER,KPER,
     *     SMAX,SMIN,NTRAC,KW,S,XOUT)
C
C  VAITK 1.3        A.N.S.I. STANDARD BASIC FORTRAN        MARCH 1976
C  VECTOR AITKEN EXTRAPOLATION
C  J. P. CHANDLER, COMPUTER SCIENCE DEPT., OKLAHOMA STATE UNIVERSITY
C
C  A. JENNINGS, ACCELERATING THE CONVERGENCE OF MATRIX ITERATIVE
C       PROCESSES, J.INST.MATH.APPLIC. 8 (1971) 99-110
C  E. F. BOYLE AND A. JENNINGS, INT.J.NUM.METH.ENG. 7 (1974) 232-235
C  E. GEKELER, MATHEMATICS OF COMPUTATION 26 (1972) 427-436
C  C. BREZINSKI AND A. C. RIEU, MATH. OF COMP. 28 (1974) 731-741
C  P. WYNN, MATH. OF COMP. 16 (1962) 301-322
C
C  INPUT QUANTITIES.....  N,X(*),METHD,KOUNT,NPREP,NPER,KPER,SMAX,SMIN,
C                              NTRAC,KW
C  OUTPUT QUANTITIES....  KOUNT,KPER,S,XOUT(*)
C  SCRATCH ARRAYS.......  TA(*),TB(*)
C
C     N         --  NUMBER OF COMPONENTS IN EACH VECTOR
C     X(*)      --  INPUT VECTOR ITERATE
C     METHD     --  =1 TO USE THE FDM METHOD OF JENNINGS,
C                   =2 TO USE THE SDM METHOD
C                        (USUALLY METHD=2 WORKS BETTER, BUT NOT ALWAYS.)
C     KOUNT     --  ITERATION COUNTER
C     NPREP     --  NUMBER OF ITERATES DISCARDED BEFORE EACH
C                        EXTRAPOLATION
C                        (NPREP.GE.0 IS USUALLY FASTEST.  NPREP=-1 GIVES
C                         THE METHOD OF BOYLE AND JENNINGS.)
C     NPER      --  =1 IF THE X(*) ITERATES ARE CONVERGING (OR DIVERGING)
C                        SMOOTHLY,
C                   =2 IF THE X(*) ITERATES ARE ZIGZAGGING WITH
C                        PERIOD =2, ETC.
C                        (NPER=2 CORRESPONDS TO THE -SQUARED-
C                         EXTRAPOLATION METHODS OF JENNINGS.)
C     KPER      --  COUNTER FOR ZIGZAGGING
C     SMAX      --  UPPER LIMIT ON THE EXTRAPOLATION FACTOR
C     SMIN      --  LOWER LIMIT ON THE EXTRAPOLATION FACTOR
C                        (SUGGESTION...  SET SMAX=100.  SET SMIN=0. IF
C                         THE ITERATION IS KNOWN TO BE MONOTONICALLY
C                         CONVERGENT (OR DIVERGENT), AND SET SMIN=-100.
C                         OTHERWISE.)
C     NTRAC     --  PRINT SWITCH, SET EQUAL TO
C                       +1 TO OBTAIN FULL PRINT,
C                        0 TO OBTAIN NORMAL PRINT,
C                       -1 TO OBTAIN NO PRINT
C     KW        --  LOGICAL UNIT NUMBER OF THE PRINTER
```

```
C     S         --  RETURNS THE VALUE OF THE EXTRAPOLATION FACTOR
C     XOUT(*)  --  RETURNS THE OUTPUT VECTOR AT EACH CALL TO VAITK
C
C  THE USER CALLS VAITK REPEATEDLY.  THE X(*) VECTORS ARE SUCCESSIVE
C  VECTORS FROM SOME ITERATION SCHEME HAVING ROUGHLY LINEAR CONVERGENCE.
C  ON RETURN FROM EACH CALL TO VAITK, THE VECTOR XOUT(*) CONTAINS THE
C  BEST EXTRAPOLATED EXTIMATE OF THE LIMIT OF THE INFINITE SEQUENCE
C  OF X(*) VECTORS.
C  X( ) AND XOUT( ) MAY BE THE SAME ARRAY.
C  THE COUNTERS KOUNT AND KPER MUST BE SET TO ZERO BEFORE THE FIRST
C  CALL TO VAITK FOR A GIVEN PROBLEM, AND NOT CHANGED UNTIL THE NEXT
C  PROBLEM IS TO BE STARTED, AT WHICH TIME THEY MUST BE SET TO ZERO
C  AGAIN.
C
C     DOUBLE PRECISION X,TA,TB,SMAX,SMIN,S,XOUT,
C    X     DXMAX,DXSQ,DXDDX,DDXSQ,DX,ABSDX
C     DIMENSION X(N),TA(N),TB(N),XOUT(N)
      DIMENSION X(1),TA(1),TB(1),XOUT(1)
C
      S=0.
      MXP=NPREP
      IF(MXP)1000,1010,1010
 1000 MXP=0
 1010 IF(KOUNT-MXP)1020,1050,1030
 1020 KOUNT=KOUNT+1
      GO TO 1370
 1030 IF(KPER-(NPER-1))1040,1050,1050
 1040 KPER=KPER+1
      GO TO 1370
 1050 KPER=0
      KOUNT=KOUNT+1
      IF(KOUNT-(MXP+2))1350,1330,1060
C
C                              IT IS TIME TO EXTRAPOLATE.  COMPUTE THE
C                                    EXTRAPOLATION FACTOR, S.
 1060 DXMAX=0.
      DXSQ=0.
      DXDDX=0.
      DDXSQ=0.
      DO 1110 J=1,N
        DX=X(J)-TA(J)
        DDX=DX-TB(J)
        ABSDX=DX
        IF(ABSDX)1070,1080,1080
 1070   ABSDX=-ABSDX
 1080   IF(ABSDX-DXMAX)1100,1100,1090
 1090   DXMAX=ABSDX
 1100   DXSQ=DXSQ+DX*DX
        DXDDX=DXDDX+DX*DDX
 1110   DDXSQ=DDXSQ+DDX*DDX
      IF(METHD-1)1120,1120,1140
C
C                            METHD=1 ...   S = -(DX,DX)/(DDX,DX)
 1120 IF(DXDDX)1130,1330,1130
 1130 S=-DXSQ/DXDDX
      GO TO 1160
C                            METHD=2 ...   S = -(DX,DDX)/(DDX,DDX)
 1140 IF(DDXSQ)1150,1330,1150
 1150 S=-DXDDX/DDXSQ
C                            CHECK THE LIMITS ON S.
 1160 IF(S-SMAX)1180,1180,1170
 1170 S=SMAX
 1180 IF(S-SMIN)1190,1200,1200
 1190 S=SMIN
 1200 IF(S)1210,1330,1210
 1210 CONTINUE
C                            PRINT INFORMATION IF REQUESTED.
      IF(NTRAC)1300,1220,1220
 1220 WRITE(KW,1230)DXMAX,S
 1230 FORMAT(/9H VAITK...,5X,9H DXMAX = ,E14.7,5X,5H S = ,E12.5)
      IF(NTRAC)1300,1300,1240
 1240 JROW=0
 1250 JL=JROW+1
      JH=JROW+10
      IF(JH-N)1270,1270,1260
 1260 JH=N
 1270 WRITE(KW,1280)JROW,(X(J),J=JL,JH)
 1280 FORMAT(/1X,I4,5X,10E11.3)
      WRITE(KW,1290)(TB(J),J=JL,JH)
 1290 FORMAT(10X,10E11.3)
      JROW=JH
      IF(JROW-N)1250,1300,1300
C
C                            EXTRAPOLATE.
 1300 DO 1310 J=1,N
        XOUT(J)=X(J)+S*(X(J)-TA(J))
 1310   TA(J)=XOUT(J)
      KOUNT=0
      IF(NPREP)1320,1390,1390
 1320 KOUNT=1
      GO TO 1390
C                            ON THE NEXT CYCLE WE WILL EXTRAPOLATE.
C                                  SAVE THE FIRST DIFFERENCE VECTOR.
 1330 DO 1340 J=1,N
 1340    TB(J)=X(J)-TA(J)                                                         VAITK142
C                                 SAVE THE X(*) ITERATE.                          VAITK143
 1350 DO 1360 J=1,N                                                               VAITK144
 1360    TA(J)=X(J)                                                               VAITK145
C                                 SET XOUT(*).                                    VAITK146
 1370 DO 1380 J=1,N                                                               VAITK147
 1380    XOUT(J)=X(J)                                                             VAITK148
C                                                                                 VAITK149
 1390 RETURN                                                                      VAITK150
C                                                                                 VAITK151
C END VAITK.                                                                      VAITK152
      END                                                                         VAITK153
```

```
C  TEST DRIVER FOR SUBROUTINE VAITK.             MARCH 1976
C
      DIMENSION XX(11,11),X(81),TA(81),TB(81),TC(81),TD(81)
C
      KW=6
      LSIDE=10
      LSIDE=5
      NITA=30
      NITB=40
      NITB=30
      SMAX=100.
      SMIN=-100.
      NTRAC=0
      HUGE=1.E35
C
C  SET UP A SYSTEM OF LINEAR EQUATIONS WITH WHICH TO TEST VAITK.
C  THIS IS THE SYSTEM WHICH RESULTS FROM THE APPROXIMATE SOLUTION OF
C  THE DIRICHLET PROBLEM ON A SQUARE, USING A FIVE-POINT DIFFERENCE
C  RULE.
      LSP=LSIDE+1
      DO 10 J=1,LSP
        DO 10 K=1,LSP
   10     XX(J,K)=0.
C
      N=(LSIDE-1)**2
      WRITE(KW,20)
   20 FORMAT(//1H1,42H INITIAL ILLUSTRATION OF ITERATION WITHOUT,
     *     17H ACCELERATION....//31H (DXMAX IS THE MAXIMUM ABSOLUTE ,
     *     39H CHANGE IN ANY COMPONENT OF THE VECTOR) /1H )
      CALL INIT (N,X)
      DO 30 IT=1,NITA
        CALL XITER (LSIDE,XX,X,DXMAX)
   30   WRITE(KW,40)IT,DXMAX
   40 FORMAT( 10H ITERATION ,I3,10X,9H DXMAX = ,E15.7)
C
      WRITE(KW,50)
   50 FORMAT(////44H TEST VAITK USING FOUR COMBINATIONS OF METHD ,
     *     13H AND NPER.... )
      DXBES=HUGE
      NPREP=0
      DO 100 M=1,2
        METHD=M
        DO 90 NP=1,2
          NPER=NP
          WRITE(KW,60)M,NPREP,NPER
   60     FORMAT(//////28H TEST OF VAITK WITH METHD = ,I1,
     *         10H, NPREP = ,I2,13H, AND NPER = ,I2//1H )
```

```
        CALL INIT (N,X)                                                  VAITDR48
        KOUNT=0                                                          VAITDR49
        KPER=0                                                           VAITDR50
        DO 70 IT=1,NITA                                                  VAITDR51
          CALL XITER (LSIDE,XX,X,DXMAX)                                  VAITDR52
          WRITE(KW,40)IT,DXMAX                                           VAITDR53
   70     CALL VAITK (N,X,TA,TB,METHD,KOUNT,NPREP,NPER,KPER,             VAITDR54
     *         SMAX,SMIN,NTRAC,KW,S,X)                                   VAITDR55
        IF(DXMAX-DXBES)80,90,90                                          VAITDR56
   80   DXBES=DXMAX                                                      VAITDR57
        MESAV=METHD                                                      VAITDR58
        NPSAV=NPER                                                       VAITDR59
   90   CONTINUE                                                         VAITDR60
  100 CONTINUE                                                           VAITDR61
C                                                                        VAITDR62
      WRITE(KW,110)MESAV,NPSAV                                           VAITDR63
  110 FORMAT(////36H THE BEST VALUES ABOVE WERE METHD = ,I1,             VAITDR64
     *     12H AND NPER = ,I1//34H FOR THIS COMBINATION, TRY VARIOUS ,   VAITDR65
     *     17H VALUES OF NPREP. )                                        VAITDR66
      METHD=MESAV                                                        VAITDR67
      NPER=NPSAV                                                         VAITDR68
      DXBES=HUGE                                                         VAITDR69
      DO 150 NP=1,4                                                      VAITDR70
        NPREP=NP-2                                                       VAITDR71
        WRITE(KW,120)NPREP                                               VAITDR72
  120   FORMAT(//////19H TEST WITH NPREP = ,I2/1H )                      VAITDR73
        CALL INIT (N,X)                                                  VAITDR74
        KOUNT=0                                                          VAITDR75
        KPER=0                                                           VAITDR76
        DO 130 IT=1,NITA                                                 VAITDR77
          CALL XITER (LSIDE,XX,X,DXMAX)                                  VAITDR78
          WRITE(KW,40)IT,DXMAX                                           VAITDR79
  130     CALL VAITK (N,X,TA,TB,METHD,KOUNT,NPREP,NPER,KPER,             VAITDR80
     *         SMAX,SMIN,NTRAC,KW,S,X)                                   VAITDR81
        IF(DXMAX-DXBES)140,150,150                                       VAITDR82
  140   DXBES=DXMAX                                                      VAITDR83
        NPSAV=NPREP                                                      VAITDR84
  150   CONTINUE                                                         VAITDR85
C                                                                        VAITDR86
      WRITE(KW,160)NPSAV                                                 VAITDR87
  160 FORMAT(//////29H THE BEST VALUE OF NPREP WAS ,I2//                 VAITDR88
     *     49H NOW TRY EXTRAPOLATING THE EXTRAPOLATED VECTORS. ,         VAITDR89
     *     36H THIS IS CALLED SUPER-EXTRAPOLATION. )                     VAITDR90
      NPREP=NPSAV                                                        VAITDR91
      NPREPB=0                                                           VAITDR92
      DO 230 MP=1,2                                                      VAITDR93
        METHB=MP                                                         VAITDR94
      DO 220 NP=1,2
        NPERB=NP
        WRITE(KW,170)METHD,NPREP,NPER,METHB,NPREPB,NPERB
  170   FORMAT(//////9H METHD = ,I1,8X,9H NPREP = ,I2,8X,
     *       8H NPER = ,I1//9H METHB = ,I1,8X,10H NPREPB = ,I2,8X,
     *       9H NPERB = ,I1/1H )
        CALL INIT (N,X)
        KOUNT=0
        KPER=0
        KOUNTB=0
        KPERB=0
        DO 210 IT=1,NITB
          CALL XITER (LSIDE,XX,X,DXMAX)
          WRITE(KW,40)IT,DXMAX
          CALL VAITK (N,X,TA,TB,METHD,KOUNT,NPREP,NPER,KPER,
     *         SMAX,SMIN,NTRAC,KW,S,X)
          IF(S)180,210,180
  180     CALL VAITK (N,X,TC,TD,METHB,KOUNTB,NPREPB,NPERB,KPERB,
     *         SMAX,SMIN,NTRAC,KW,S,X)
          IF(S)190,210,190
  190     WRITE(KW,200)
  200     FORMAT(46H SUPER-EXTRAPOLATION WAS PERFORMED JUST ABOVE. /
     *         1H )
  210     CONTINUE
  220   CONTINUE
  230 CONTINUE
C
      STOP
C
C END VAITK TEST DRIVER.
      END
      SUBROUTINE INIT (N,X)
C
C  INITIALIZES THE VECTOR X(*) FOR THE VAITK TEST DRIVER.
C
      DIMENSION X(1)
      DO 200 J=1,N
  200   X(J)=1.
      RETURN
      END
      SUBROUTINE XITER (LSIDE,XX,X,DXMAX)
C
C  PERFORMS ONE BASIC ITERATION ON THE VECTOR X(*), FOR THE VAITK
C  TEST DRIVER.
C
C  IF MITER=0, THE JACOBI METHOD (METHOD OF SIMULTANEOUS REPLACEMENTS)
C  IS USED TO SOLVE A SYSTEM OF N LINEAR EQUATIONS.
```

```
C  IF MITER=1, THE GAUSS-SEIDEL METHOD (METHOD OF SUCCESSIVE
C  REPLACEMENTS) IS USED.
C
      DIMENSION XX(11,11),X(81),XNEW(81)
C
      MITER=0
      MITER=1
C
      RFOUR=4
      DXMAX=0.
C                                MOVE X(*) INTO XX(*,*).
      L=0
      DO 300 J=2,LSIDE
        DO 300 K=2,LSIDE
          L=L+1
  300     XX(J,K)=X(L)
C                                COMPUTE XNEW(*) FROM XX(*,*).
      L=0
      DO 370 J=2,LSIDE
        DO 360 K=2,LSIDE
          L=L+1
          XNEW(L)=(XX(J-1,K)+XX(J+1,K)+XX(J,K-1)+XX(J,K+1))/RFOUR
          DX=XNEW(L)-X(L)
          IF(DX)310,320,320
  310     DX=-DX
  320     IF(DX-DXMAX)340,340,330
  330     DXMAX=DX
  340     IF(MITER)350,360,350
  350     XX(J,K)=XNEW(L)
  360     CONTINUE
  370   CONTINUE
C                                UPDATE X(*) FROM XNEW(*).
      N=(LSIDE-1)**2
      DO 380 J=1,N
  380   X(J)=XNEW(J)
C
      RETURN
C
C END XITER.
      END
```

VITA

Ronald Charles Walters

Candidate for the Degree of

Master of Science

Thesis: THE INVESTIGATION AND NUMERICAL SOLUTION OF SELECTED BOUNDARY VALUE PROBLEMS

Major Field: Computing and Information Sciences

Biographical:

Personal Data: Born in Miami, Oklahoma, September 24, 1952, the son of Mr. and Mrs. Gordon L. Walters.

Education: Graduated from Miami High School, Miami, Oklahoma, in May, 1970; received the Degree of Associate of Arts from Northeastern A & M, Miami, Oklahoma, in May, 1972; received the Degree of Bachelor of Science from Oklahoma State University, Stillwater, Oklahoma, in May, 1974; with a major in Mathematics; completed requirements for the Master of Science degree at Oklahoma State University, in May, 1976.

Professional Experience: Graduate Teaching Assistant in the Computing and Information Sciences Department at Oklahoma State University from August, 1974 to May, 1976; member of the Association for Computing Machinery; Vice President of the OSU Student Chapter of the ACM from August, 1975 until May, 1976.