IS-3396

A DIRECT METHOD BASED ON PROJECTIONS FOR SOLVING SYSTEMS OF LINEAR EQUATIONS

D. W. Harms

R. F. Keller

AMES LABORATORY, USAEC

IOWA STATE UNIVERSITY

AMES, IOWA

Date of Transmittal: May 1974

PREPARED FOR THE U. S. ATOMIC ENERGY COMMISSION DIVISION OF RESEARCH

UNDER CONTRACT W-7405-eng-82

IS-3396

A DIRECT METHOD BASED ON PROJECTIONS FOR SOLVING SYSTEMS OF LINEAR EQUATIONS

D. W. Harms and R. F. Keller

Based on Ph.D. Thesis Submitted to Iowa State University, May 1974

Ames Laboratory, USAEC
Iowa State University
Ames, Iowa 50010

PREPARED FOR THE U. S. ATOMIC ENERGY COMMISSION
DIVISION OF RESEARCH UNDER CONTRACT NO. W-7405-eng-82

Date of Manuscript - May 1974

Available from: National Technical Information Service
Department A
Springfield, VA 22151

Price: Microfiche $1.45
Paper Copy $5.45

TABLE OF CONTENTS

ABSTRACT

The One-Dimensional Reduction by Projection Method and the Direct Projection Method, as developed and discussed in this paper, are two new methods of solving linear systems.

The One-Dimensional Reduction by Projection Method uses the (n-1)-dimensional projection method to reduce the problem of solving an n-dimensional system $Ax = b$ to the problem of solving two (n-1)-dimensional systems $Bx = V_1$ and $Bx = V_2$. The two (n-1)-dimensional systems have the same coefficient matrix B. Also, B is symmetric and positive definite. This property is important, for the Gauss-Seidel iteration converges, the Orthogonalization Method can be applied, and a compact form of Gaussian Elimination can be applied. These three methods can be shown to have computational advantages over other methods. Gauss-Seidel, when it converges, is fast and accurate on large sparse systems. Orthogonalization and compact Gaussian Elimination are direct methods that take fewer operations and, hence, develop less roundoff than the usual Gaussian Elimination with complete pivoting. Also, in the compact form of Gaussian Elimination no pivots need to be taken on a symmetric positive definite matrix.

The Direct Projection Method uses repeated application of the One-Dimensional Reduction by Projection Method to reduce an n-dimensional system, to (n-1)-dimensional systems, then to (n-2)-dimensional systems and so forth, until 1-dimensional systems are obtained. These then can easily be solved by division.

I. INTRODUCTION AND LITERATURE REVIEW

This paper discusses certain aspects of solving a system of n linear equations in n unknowns written as:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n. \end{aligned} \tag{1.1}$$

In the usual way, A denotes the matrix of coefficients, b denotes the constant vector, and x is the solution vector. Thus, the above system can be written as

$$Ax = b. \tag{1.2}$$

As a matter of notation, let a_i indicate the i^{th} column of the matrix A.

The following theorem can be found in White (17, p. 117).

Theorem 1:

A system of n linear equations in n unknowns (as in 1.1) has a unique solution if and only if the determinant of the coefficient matrix A does not equal zero. That is, the system is nonsingular.

It will be assumed throughout this paper that A is nonsingular.

In general, a given vector x will only approximate the solution to 1.2. Therefore, the following definition is made as a measure of how close x is to the solution.

Definition 1:

The residue vector $R^{(k)}$ is defined as

$$R^{(k)} = b - Ax^{(k)}.$$

The vector $R^{(k)}$ is n-dimensional and is denoted by $R^{(k)} = (R_1^{(k)}, R_2^{(k)}, \cdots, R_n^{(k)})$.

In the following, the inner product of two vectors a and b is denoted by (a,b).

For reference purposes in later parts of this paper, several methods for solving linear systems will now be discussed.

The best known method for solving linear systems is Gaussian Elimination. It is, basically, the elementary procedure in which the first "equation" is used to eliminate the first "variable" from the last n-1 "equations". Then the new second "equation" is used to eliminate the second "variable" from the last n-2 "equations", etc. (See Isaacson and Keller [6, p. 29].)

Gaussian Elimination is best demonstrated by example, so consider the following 3-dimensional system. (See Fox [1, p. 60].)

$$\begin{bmatrix} 4 & -9 & 2 \\ 2 & -4 & 6 \\ 1 & -1 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 5 \\ 3 \\ 4 \end{bmatrix}$$

Row one multiplied by -1/2 is added to row two and row one multiplied by -1/4 is added to row 3, producing the system

$$\begin{bmatrix} 4 & -9 & 2 \\ 0 & 0.5 & 5 \\ 0 & 1.25 & 2.5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 5 \\ 0.5 \\ 2.75 \end{bmatrix}.$$

Row two multiplied by -2.5 is then added to row three, producing the system

$$\begin{bmatrix} 4 & -9 & 2 \\ 0 & 0.5 & 5 \\ 0 & 0 & -10 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 5 \\ 0.5 \\ 1.5 \end{bmatrix}.$$

This system is then solved by a process called back substitution. It can easily be seen that $x_3 = -.15$. This is then substituted into the second row, producing the equation

$$0.5x_2 - .75 = 0.5.$$

This equation is solved, producing $x_2 = 2.5$. Then x_3 and x_2 are substituted into the first row to produce the equation

$$4x_1 - 22.5 - .3 = 5.$$

This equation is solved, resulting in $x_1 = 6.95$.

The multiplier $-1/2$ was obtained by taking the negative quotient a_{21}/a_{11}. In this case, a_{11} is called the pivot. Of course, a_{11} cannot be a pivot if $a_{11} = 0$. It can also be shown (see Fox [1, p. 91]) that there is less computational error in the method if the largest in absolute value element of the matrix is taken as the first pivot. The second pivot is then the element largest in absolute value in the remaining

matrix. This process is called complete pivoting.

Computational error is inherent in Gaussian Elimination with complete pivoting for $2n-1$ operations are involved in computing the n,n position of the diagonal matrix. Since there is a fixed number of digits, say p, in any computing machine, the n,n position might have no more than $p-2$ significant digits if n were in the neighborhood of 100. (See Fox [1, p. 163].)

Also, if the system were ill-conditioned (Definition 3), the pivots have a tendency to become scaled differently than the rest of the matrix, creating less accuracy in the division to calculate the multipliers. This is also a serious handicap for large systems, for they have a tendency to be more ill-conditioned.

Gaussian Elimination is called a direct method, for it calculates the solution directly. Another, less known but powerful, direct method is the compact elimination method of Cholesky. Fox (1, p. 185) says it has significant advantages over other elimination methods. However, it works only for symmetric positive definite coefficient matrices (see Definition 2).

The method of Cholesky (Fox [1, p. 107]) is basically to decompose the matrix A into the form

$$A = LL^T$$

where L is lower triangular and L^T is the transpose of L.

The system $Lf = b$ is then solved by back substitution followed by solving the system $L^Tx = f$ by back substitution. The matrix L can be found by using the following equations:

$$\ell_{kk} = (a_{kk} - \sum_{p=1}^{k-1} \ell_{kp}^2)^{1/2} \qquad 1.3$$

$$\ell_{jk} = \frac{1}{\ell_{kk}} (a_{kj} - \sum_{p=1}^{k-1} \ell_{jp}\ell_{pk}) \quad \text{for} \quad j > k. \qquad 1.4$$

k is set equal to one, and the first equation is solved. The second equation is then solved for $j = 2, 3, \cdots$. k is then set equal to two, and the first equation is solved. The second equation is then solved for $j = 3, 4, \cdots$. This process is continued until L is found.

If the matrix A is positive definite, then the square root in 1.3 is real and ℓ_{kk} for any k does not equal zero in the division in 1.4 (6, p. 54).

The advantages of Cholesky's method are: (1) if $|a_{ij}| < 1$ for all i and j, then $|\ell_{ij}| < 1$ for all i and j and the scaling problems of Gaussian Elimination are avoided; (2) the extra work in pivoting is avoided; (3) only the upper triangle of the coefficient matrix need be stored.

A third direct method for solving linear systems is the Orthogonalization Method (1, p. 125). It is of interest here because it is designed for symmetric and positive definite coefficient matrices. The solution x to the system 1.2 is assumed to be of the form

$$x = \sum_{r=1}^{n} \alpha_r x^{(r)} \qquad 1.5$$

where the α_r are constants and the $x^{(r)}$ are linearly independent n-dimensional vectors that are conjugate, i.e.,

$$x^{(r)T}Ax^{(s)} = x^{(s)T}Ax^{(r)} = 0 \quad \text{for} \quad r \neq x.$$

Substituting 1.5 into 1.2 yields the equation

$$\sum_{r=1}^{n} \alpha_r Ax^{(r)} = b. \tag{1.6}$$

Premultiplying 1.6 by $x^{(r)T}$ yields the equation

$$\alpha_r x^{(r)T}Ax^{(r)} = x^{(r)T}b,$$

and solving for α_r yields

$$\alpha_r = x^{(r)T}b/x^{(r)T}Ax^{(r)}.$$

The problem of finding the $x^{(r)}$'s is left. To do this, a process is used very similar to the Gram-Schmidt process.

$$\begin{aligned} \text{Let } \quad x^{(1)} &= e^{(1)} \\ x^{(2)} &= e^{(2)} - \alpha_{12}x^{(1)} \\ x^{(3)} &= e^{(3)} - \alpha_{13}x^{(1)} - \alpha_{23}x^{(2)} \\ &\text{etc.} \end{aligned} \tag{1.7}$$

where $e^{(j)}$ is a unit vector with all zeros except for a 1 in the j^{th} position.

Since $x^{(1)}$ and $x^{(2)}$ must be conjugate,

$$x^{(1)T}A(e^{(2)} - \alpha_{12}x^{(1)}) = 0,$$

and solving for α_{12} yields

$$\alpha_{12} = \frac{x^{(1)^T} Ae(2)}{x^{(1)^T} ax(1)} .$$

Now, multiplying the third equation in 1.7 by $x^{(1)^T} A$ yields

$$x^{(1)^T} Ax(3) = x^{(1)^T} Ae(3) - \alpha_{13} x^{(1)^T} Ax(1),$$

and solving for α_{13} yields

$$\alpha_{13} = \frac{x^{(1)^T} Ae(3)}{x^{(1)^T} Ax(1)} .$$

It can be shown in general that

$$\alpha_{rs} = \frac{x^{(r)^T} Ae(s)}{x^{(r)^T} Ax(r)} \quad \text{for} \quad r < s.$$

The next method of interest to this paper is the iterative method called Gauss-Seidel (1, p. 191). An iterative method takes an initial guess of the solution $x^{(0)}$ and calculates a new guess $x^{(1)}$. Successive guesses are calculated in this fashion in the hope that the sequence

$$x^{(0)}, x^{(1)}, x^{(2)}, \ldots$$

will converge to the solution of the system.

The Gauss-Seidel is a single-step method, for it changes only one component of the solution vector at a time. It cycles through all the components, usually in order, until a cycle is completed. The Gauss-Seidel iteration for the j^{th} component of the r^{th} cycle is

$$x_j^{(r+1)} = \frac{b - \sum_{i=1}^{j-1} a_{ji}x_i^{(r+1)} - \sum_{i=j+1}^{n} a_{ji}x_i^{(r)}}{a_{jj}}.$$

Decompose A into $U + D + L$ where U consists of the above diagonal elements of A, D consists of the diagonal elements of A, and L consists of the below diagonal elements of A. All three matrices are zero in the other positions. Then the Gauss-Seidel iteration corresponds to the iterative equation

$$(D + L)x^{(r+1)} = b - Ux^{(r)}.$$

The problem with the Gauss-Seidel iteration is that it does not converge for all nonsingular matrices. It has the advantage of being simple to implement and the advantage of not requiring the storage of the entire coefficient matrix. If a matrix is a band matrix, i.e., nonzero only in a band around the diagonal, only the band needs to be stored.

The last method to be discussed in this section is the k-dimensional projection method (16, p. 32). This is really a family of methods, for k can be any number from 1 to n producing a different method, although very similar, for each k.

One step of the k-dimensional projection method changes k positions of the solution vector. For notational convenience, it can be assumed that the positions being changed are $q, q+1, \cdots, q+k^{-1}$. This notation tends to imply the positions changed must be in numerical sequence, but this does not necessarily have to be true. Let dx be

a k-dimensional vector. The vector dx will correspond to the changes to be made in the $q, q+1, \cdots, q+k-1$ positions, and dx'_{q+i} will equal dx_{i+1} for $i = 0, 1, \cdots, k-1$ and be zero in its other positions. Therefore, the new solution vector

$$x^{(r+1)} = x^{(r)} + dx'.$$

The controlling characteristic of the projection method is the desire to maximize the expression

$$(R^{(r)}, R^{(r)}) - (R^{(r+1)}, R^{(r+1)})$$

concerning the residues of $x^{(r)}$ and $x^{(r+1)}$. (See Definition 1.)

It can be shown (see White [16, p. 48]) that the above requirement is met if the following equations hold:

$$\begin{aligned}(a_q, R^{(r+1)}) &= 0\\ (a_{q+1}, R^{(r+1)}) &= 0\\ &\vdots\\ (a_{q+k-1}, R^{(r+1)}) &= 0.\end{aligned}$$

Since $R^{(r+1)} = b - Ax^{(r+1)}$ and $x^{(r+1)} = x^{(r)} + dx'$, the above can be rewritten as

$$\begin{aligned}(a_q, b - Ax^{(r)} - Adx') &= 0\\ (a_{q+1}, b - Ax^{(r)} - Adx') &= 0\\ &\vdots\\ (a_{q+k-1}, b - Ax^{(r)} - Adx') &= 0.\end{aligned}$$

By using the laws of dot products and the definition

$$b - Ax^{(r)} = R^{(r)},$$

the above can be rewritten as

$$\begin{aligned}
(a_q,\ Adx') &= (a_q,\ R^{(r)}) \\
(a_{q+1},\ Adx') &= (a_{q+1},\ R^{(r)}) \\
&\vdots \\
(a_{q+k-1},\ Adx') &= (a_{q+k-1},\ R^{(r)}).
\end{aligned}$$

It can be seen that the above is equivalent to the following k-dimensional linear system.

$$\begin{aligned}
(a_q,a_q)dx_q + (a_q,a_{q+1})dx_{q+1} + \cdots + (a_q,a_{q+k-1})dx_{q+k-1} &= (a_q,R^{(r)}) \\
(a_{q+1},a_q)dx_q + (a_{q+1},a_{q+1})dx_{q+1} + \cdots + (a_{q+1},a_{q+k-1})dx_{q+k-1} &= (a_{q+1},R^{(r)}) \\
&\vdots \\
(a_{q+k-1},a_q)dx_q + (a_{q+k-1},a_{q+1})dx_{q+1} + \cdots + (a_{q+k-1},a_{q+k-1})dx_{q+k-1} &= (a_{q+k-1},R^{(r)}).
\end{aligned}$$

Therefore, each step of a k-dimensional projection method requires the solution to a k-dimensional system.

The advantage of the projection method over Gauss-Seidel is that the projection method will converge for any nonsingular coefficient matrix. The problem with the method is that it is usually slow with an arbitrary ordering of projections. Mok Tokko (14) and Dennis Georg (2) have shown that reordering the projections can speed the convergence up by a factor of 50 or more. Of course, the problem is to decide on an optimal ordering. This usually takes a lot of computation.

It is the hope of the author to show in the following pages a method that will combine the speed of Gauss-Seidel with the dependability of the

projection method. As an added feature, a heretofore unknown direct method based on projections will be presented.

II. DEVELOPMENT

In the words of linear algebra, the first n-1 columns of a nonsingular matrix A span an (n-1)-dimensional vector subspace. Two n-dimensional vectors orthogonal to each of the n-1 columns must be parallel. The following theorem states this fact.

Theorem 2:

Let $\{a_1, a_2, \cdots, a_{n-1}\}$ be the set of vectors that form the first n-1 columns of a nonsingular matrix A. Given two vectors $R^{(1)}$ and $R^{(2)}$ such that

$$(a_i, R^{(1)}) = 0 \quad \text{and} \quad (a_i, R^{(2)}) = 0$$

for all $a_i \in \{a_1, a_2, \quad , a_{n-1}\}$, then $tR^{(1)} = R^{(2)}$ for some real t.

The following theorem shows that if two distinct vectors $x^{(1)}$ and $x^{(2)}$ produce parallel and distinct residues, $R^{(1)}$ and $R^{(2)}$, then the solution to 1.2 can be obtained as a linear combination of $x^{(1)}$ and $x^{(2)}$.

Theorem 3:

Let $x^{(1)}$ and $x^{(2)}$ be two vectors such that $x^{(1)} \neq x^{(2)}$ and t a real number such that $R^{(2)} = tR^{(1)}$ where $R^{(1)} = b - Ax^{(1)}$, $R^{(2)} = b - Ax^{(2)}$, and $t \neq 1$. Define

$$\hat{x} = x^{(1)} - (x^{(1)} - x^{(2)})/(1 - t),$$

then $\hat{x}$ is the solution of the system $Ax = b$.

Proof:

Substituting $\hat{x}$ for x yields

$$A\hat{x} = A[x^{(1)} - (x^{(1)} - x^{(2)})/(1 - t)].$$

This right side becomes

$$Ax^{(1)} - (Ax^{(1)} - Ax^{(2)})/(1 - t)$$

after multiplying through by A. Adding and subtracting b two different places gives

$$Ax^{(1)} - b + b - (Ax^{(1)} - b + b - Ax^{(2)})/(1 - t).$$

Combining terms yields

$$-(b - Ax^{(1)}) + b - \frac{-(b - Ax^{(1)}) + (b - Ax^{(2)})}{(1 - t)}.$$

Now, substituting $R^{(1)} = b - Ax^{(1)}$ and $R^{(2)} = b - Ax^{(2)}$ yields

$$-R^{(1)} + b - (-R^{(1)} + R^{(2)})/(1 - t).$$

Now $tR^{(1)} = R^{(2)}$, so this becomes

$$-R^{(1)} + b - (R^{(1)}(-1 + t))/(1 - t)$$

which simplifies to b. Therefore,

$$A\hat{x} = b$$

and $\hat{x}$ solves the equation $Ax = b$.

The problem of solving the system $Ax = b$ has been converted to the problem of finding $x^{(1)}$ and $x^{(2)}$ as above. But the (n-1)-dimensional projection method applied to the first n-1 columns of A will supply $x^{(1)}$ and $x^{(2)}$.

Theorem 4:

Let $x_0^{(1)}$ and $x_0^{(2)}$ be two vectors that differ in the n^{th} position. Applying the (n-1)-dimensional projection method to $x_0^{(1)}$ and $x_0^{(2)}$ using the first n-1 columns of A yields two vectors $x^{(1)}$ and $x^{(2)}$, respectively, that satisfy the conditions of Theorem 3.

Proof:

By construction, $R^{(1)}$ and $R^{(2)}$ are both orthogonal to the first n-1 columns of A. Therefore, by Theorem 2, $R^{(1)} = tR^{(2)}$.

If $t = 1$, then $R^{(1)} = R^{(2)}$, or by Definition 1,

$$b - Ax^{(1)} = b - Ax^{(2)}.$$

Applying algebra yields

$$Ax^{(1)} = Ax^{(2)}. \qquad 2.1$$

Since A is nonsingular, A^{-1} exists. Multiplying both sides of 2.1 by A^{-1} yields

$$x^{(1)} = x^{(2)}.$$

However, this is a contradiction since $x_0^{(1)}$ and $x_0^{(2)}$ are unequal in the n^{th} position, and the n^{th} positions of $x_0^{(1)}$ and $x^{(1)}$ are equal

as are the n^{th} positions of $x_0^{(2)}$ and $x^{(2)}$.

To apply the One-Dimensional Reduction by Projection Method, then, the two systems

$$Bdx^{(1)} = V^{(1)} \quad \text{and} \quad Bdx^{(2)} = V^{(2)}$$

must be solved. Notice the coefficient matrix B is the same in both cases. This observation makes it wise to study some properties of the matrix B. This study will begin with a standard definition found in Fox (1, p. 46).

Definition 2:

A matrix M is positive definite if and only if for any real vector x

$$x^T Mx > 0$$

where x^T indicates the transpose of the vector x.

Theorem 5:

The matrix B is symmetric and positive definite.

Proof:

That B is symmetric is clear because the two dot products (a_i, a_j) and (a_j, a_i) are equal for any i and j.

Let X be any (n-1)-dimensional vector; then

$$x^T Bx = (x_1, x_2, \cdots, x_{n-1}) \begin{bmatrix} (a_1,a_1),(a_1,a_2),\cdots,(a_1,a_{n-1}) \\ (a_2,a_1),(a_2,a_2),\cdots,(a_2,a_{n-1}) \\ \vdots \\ (a_{n-1},a_1),(a_{n-1},a_2),\cdots,(a_{n-1},a_{n-1}) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \end{bmatrix}$$

$$= (x_1(a_1, a_1) + x_2(a_2, a_1) + \cdots + x_{n-1}(a_{n-1}, a),$$
$$x_1(a_1, a_2) + x_2(a_2, a_2) + x_3(a_3, a_2) + \cdots + x_{n-1}(a_{n-1}, a_2)).$$

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \end{bmatrix} = x_1^2(a_1, a_1) + x_1x_2(a_2, a_1) + \cdots + x_1x_{n-1}(a_{n-1}, a_1)$$
$$+ x_1x_2(a_1, a_2) + x_2^2(a_2, a_2) + \cdots + x_2x_{n-1}(a_{n-1}, a_2) + \cdots$$
$$+ x_1x_{n-1}(a_1, a_{n-1}) + x_2x_{n-1}(a_2, a_{n-1}) + \cdots + x_{n-1}^2(a_{n-1}, a_{n-1})$$
$$= (x_1a_1, x_1a_1 + x_2a_2 + \cdots + x_{n-1}a_{n-1}) + (x_2a_2, x_1a_1 + x_2a_2$$
$$+ \cdots + x_{n-1}a_{n-1}) + \cdots + (x_{n-1}a_{n-1}, a_1x_1 + a_2x_2 + \cdots$$
$$+ a_{n-1}x_{n-1})$$
$$= (a_1x_1 + a_2x_2 + \cdots + a_{n-1}x_{n-1}, a_1x_1 + a_2x_2 + \cdots + a_{n-1}x_{n-1}).$$

This last dot product is of the form (a, a). Therefore, it is greater than or equal to zero. However, $(a, a) = 0$ only if $a = 0$, that is, only if

$$a_1x_1 + a_2x_2 + \cdots + a_{n-1}x_{n-1} = 0.$$

But this is impossible since A is nonsingular.

The following theorem found in Fox (1, p. 193) will demonstrate why the two systems with symmetric and positive definite coefficient matrix may be easier to solve than the original system.

Theorem 6:

The Gauss-Siedel and the Jacobi iterations converge for symmetric and positive definite coefficient matrix.

Also, there exist special methods that are designed for symmetric and positive definite systems. (See Chapter I.) Two of these are

orthogonalization and the compact elimination method of Cholesky.

So far, what can be called the One-Step Parallel Residue Method has been discussed. That is, solving the two systems

$$Bdx^{(1)} = V^{(1)} \quad \text{and} \quad Bdx^{(2)} = V^{(2)}. \tag{2.2}$$

and using Theorem 3 for computing x from $x^{(1)}$, $x^{(2)}$, and t. This method, however, can also be applied to each of the two systems, obtaining four systems

$$\begin{aligned} Cdx_c^{(1)} &= V_c^{(1)} \\ Cdx_c^{(2)} &= V_c^{(2)} \\ Cdx_c^{(3)} &= V_c^{(3)} \\ \text{and} \quad Cdx_c^{(4)} &= V_c^{(4)}. \end{aligned}$$

This process can then be continued down to one-by-one systems which can easily be solved by division. This method will be called the Direct Parallel Residue Method. The rest of this paper will be divided into two sections to study the two methods separately.

III. ONE-DIMENSIONAL REDUCTION BY PROJECTION

A. Error Estimate

To calculate t from the equation

$$R^{(2)} = tR^{(1)},$$

it is only necessary to divide $R_1^{(2)}$ by $R_1^{(1)}$. If $R^{(2)}$ and $R^{(1)}$ are exact, that is, if $x^{(2)}$ and $x^{(1)}$ are exact, it should be true that

$$R_1^{(2)}/R_1^{(1)} = R_2^{(2)}/R_1^{(1)} = \cdots = R_n^{(2)}/R_n^{(1)}.$$

Unfortunately, due to the inaccuracies of calculating, these ratios are not in general equal. They will, however, give a way of estimating the error in the computation. Assume that $x^{(2)}$ and $x^{(1)}$ are exact. Clearly, this is an unreasonable assumption, but the error in $x^{(2)}$ and $x^{(1)}$ is reflected in the error in t.

Let ε_y be the error in calculating the solution y and let ε_t be the error in calculating the constant t. Then

$$y + \varepsilon_y = x^{(1)} - \frac{(x^{(1)} - x^{(2)})}{1 - t - \varepsilon_t}$$

or

$$\varepsilon_y = x^{(1)} - \frac{(x^{(1)} - x^{(2)})}{1 - t - \varepsilon_t} - y.$$

Since

$$y = x^{(1)} - \frac{(x^{(1)} - x^{(2)})}{1 - t},$$

$$\varepsilon_y = x^{(1)} - \frac{(x^{(1)} - x^{(2)})}{1 - t - \varepsilon_t} - x^{(1)} + \frac{(x^{(1)} - x^{(2)})}{1 - t}$$

$$= (x^{(1)} - x^{(2)}) \frac{(1 - t - \varepsilon_t) - (1 - t)}{(1 - t - \varepsilon_t)(1 - t)}$$

$$= (x^{(1)} - x^{(2)}) \frac{-\varepsilon_t}{(1 - t)(1 - (t + \varepsilon_t)} .$$

Applying norms yields

$$\|\varepsilon_y\| = \|x^{(1)} - x^{(2)}\| \frac{|\varepsilon_t|}{|1 - t|\ |1 - (t + \varepsilon_t)|} .$$

$x^{(1)}$ and $x^{(2)}$ are calculated. For ε_t the greatest difference between two of the ratios $R_1^{(2)}/R_1^{(1)}$, $R_2^{(2)}/R_2^{(1)}, \cdots, R_n^{(2)}/R_n^{(1)}$ can be used. The denominator can be made smaller, hence the fraction larger, if t and $t+\varepsilon_T$ are replaced with the above ratio that is closest to 1. This error estimate is applied in the examples at the end of this section.

B. Condition Number

The condition number of a matrix A is a measure of ill-conditioned-ness of the matrix, that is, a measure of how well the system can be solved. In general, the larger the condition number, the harder the system is to solve accurately. The following definition is due to Fox (1, p. 141).

Definition 3:

The condition number of a matrix A is defined to be $k(A) = \|A\|\ \|A^{-1}\|$.

The following examples show that the condition number of the matrix A

from the original system can be but does not necessarily have to be larger than the condition number of the matrix B of the reduced system. The norm used is the spectral radius, i.e., the largest Eigenvalue in absolute value.

The first example shows $k(A) > k(B)$, and the second example shows $k(A) < k(B)$.

Example 1:

$$A = \begin{bmatrix} 1 & 3 & 1 \\ 1 & -1 & 2 \\ 1 & 4 & 1 \end{bmatrix} \qquad A^{-1} = \begin{bmatrix} 9 & -1 & -7 \\ -1 & 0 & 1 \\ -5 & 1 & 4 \end{bmatrix}$$

$$\lambda^3 - \lambda^2 - 13\lambda + 1 = 0 \qquad \lambda^3 - 13\lambda^2 - \lambda + 1 = 0$$

$$\lambda = 3.18, 4.11, .08 \qquad \lambda = -0.31, 13.07, .24$$

$$k(A) = \|A\| \ \|A^{-1}\| = 53.72$$

$$B = \begin{bmatrix} 3 & 6 \\ 6 & 26 \end{bmatrix} \qquad B^{-1} = \begin{bmatrix} 13/21 & -1/7 \\ -1/7 & 1/14 \end{bmatrix}$$

$$\lambda^2 - 29\lambda + 42 = 0 \qquad \lambda^2 - .69\lambda + .06 = 0$$

$$\lambda = 27.47, 1.52 \qquad \lambda = .49, .13$$

$$k(B) = \|B\| \ \|B^{-1}\| = 13.46$$

Example 2:

$$A = \begin{bmatrix} 1 & -1 & 0 \\ 2 & 3 & 1 \\ 1 & -2 & 2 \end{bmatrix} \qquad A^{-1} = \begin{bmatrix} .73 & .18 & -.09 \\ -.27 & .18 & -.09 \\ -.64 & .09 & .45 \end{bmatrix}$$

$\lambda^3 - 6\lambda^2 + 15\lambda - 11 = 0$ $\qquad$ $\lambda^3 - 1.36\lambda^2 + .54\lambda - .09 = 0$

$\lambda = 1.18, 2.41 \pm 1.87i$ $\qquad$ $\lambda = .85, .26 \pm .20i$

$$k(A) = \|A\| \ \|A^{-1}\| = 2.59$$

$$B = \begin{bmatrix} 6 & 3 \\ 3 & 14 \end{bmatrix} \qquad B^{-1} = \begin{bmatrix} .19 & -.04 \\ -.04 & .08 \end{bmatrix}$$

$\lambda^2 - 20\lambda + 75 = 0$ $\qquad$ $\lambda^2 - .27\lambda + .0136 = 0$

$\lambda = 15, 5$ $\qquad$ $\lambda = .20, .07$

$$k(B) = \|B\| \ \|B^{-1}\| = (15)(.20) = 3$$

C. Storage

To calculate the error bound of Part A, the ratios $R_i^{(2)}/R_i^{(1)}$ must be calculated. Therefore, A must be retained in storage. Thus, B must be stored separately. The following expressions count the amount of storage needed to apply the One-Dimensional Reduction by Projection Method if the error estimate is desired:

1. $\frac{(n-1)(n-1)}{2} + \frac{(n-1)}{2} + 2(n-1)$ for B

   (B is symmetric with two right-hand sides).

2. $2(n-1)$ for $x^{(1)}$ and $x^{(2)}$.

3. n for t.

The above expressions total $S_E = \frac{(n^2 + 9n - 8)}{2}$.

If the error estimate is not desired, only the first row of A must be retained and B can be written over the last n-1 rows of A. So only one extra storage position is needed for t, and $2(n-1)$ storage positions are needed for $x^{(1)}$ and $x^{(2)}$. This totals

$$S = 2n - 1.$$

These results are tabulated below. These values are the storage needed in excess of that already used to store the nxn system.

As was pointed out in Chapter I, one advantage of an iterative method is that sparse matrices do not have to be stored in their entirety. If calculation of the reduced matrix B destroys the property of "sparseness," the One-Dimensional Reduction by Projection Method would be considered less useful. However, consider the following "band" matrix A:

$$A = \begin{bmatrix} a_{11} & a_{12} & 0 & 0 & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 & 0 & 0 & 0 \\ 0 & a_{32} & a_{33} & a_{34} & 0 & 0 & 0 \\ 0 & 0 & a_{43} & a_{44} & a_{45} & 0 & 0 \\ 0 & 0 & 0 & a_{54} & a_{55} & a_{56} & 0 \\ 0 & 0 & 0 & 0 & a_{65} & a_{66} & a_{67} \\ 0 & 0 & 0 & 0 & 0 & a_{76} & a_{77} \end{bmatrix}.$$

Calculating B yields a matrix of the form

$$B = \begin{bmatrix} b_{11} & b_{12} & b_{13} & 0 & 0 & 0 \\ b_{21} & b_{22} & b_{23} & b_{24} & 0 & 0 \\ b_{31} & b_{32} & b_{33} & b_{34} & b_{35} & 0 \\ 0 & b_{42} & b_{43} & b_{44} & b_{45} & b_{46} \\ 0 & 0 & b_{53} & b_{54} & b_{55} & b_{56} \\ 0 & 0 & 0 & b_{64} & n_{65} & b_{66} \end{bmatrix}.$$

The band width has been increased by two diagonals, and the number of nonzero positions has increased from 19 to 24. However, since B is symmetric, the number of positions which must be stored has been reduced from 19 to 15.

Table 1. Storage in the One-Dimensional Reduction by Projection Method

System size	S_E Excess with error estimate	S Excess with no error estimate	Amount used to store system
2	7	3	6
3	14	5	12
4	22	7	20
5	31	9	30
6	41	4	42
7	52	13	56
8	64	15	72
9	77	17	90
10	91	19	110
11	106	21	132
12	122	23	156
13	139	25	182
25	371	49	650
50	1471	99	2550
100	5446	199	10100

D. Operations

As a measure of the efficiency of the One-Dimensional Reduction by Projection Method in solving an nxn linear system, the number of arithmetic operations will be counted. For the purposes of this paper, additions and subtractions will be counted together and multiplications and divisions will be counted together.

If it is assumed that

$$x_0^{(1)} = (0, 0, \cdots, 0) \quad \text{and} \quad x_0^{(2)} = (0, 0, \cdots, 0, 1),$$

then

$$R_0^{(1)} = b - Ax_0^{(1)} = b \quad \text{and} \quad R_0^{(2)} = b - a_n.$$

Thus, the computation involved in computing the residues $R_0^{(1)}$ and $R_0^{(2)}$ is cut down.

Since the n-1 x n-1 matrix B is symmetric, only the n-1 diagonal elements and the $\frac{(n-1)(n-1)}{2} - \frac{(n-1)}{2}$ above diagonal elements need be calculated.

Each of these positions is a dot product of two n-dimensional vectors and therefore requires n additions and n multiplications to be computed. To calculate each of $V^{(1)}$ and $V^{(2)}$, n-1 dot products must be computed also requiring n additions and n multiplications. In addition, $V^{(2)}$ requires n subtractions for each of the dot products to calculate $R_0^{(2)}$.

The two systems

$$Bx = V^{(1)} \quad \text{and} \quad Bx = V^{(2)} \qquad 3.1$$

can be solved by any method, but assume that they are solved by Gaussian elimination. Then Fox (1, p. 176) shows that

$$(n-1)[\tfrac{1}{3}(n-1)^2 + \tfrac{2}{3} + 2(n-1)] \quad \text{multiplications}$$

and

$$(n-1)[\tfrac{1}{3}(n-1)^2 - \tfrac{1}{2}(n-1) + \tfrac{1}{6} + 2(n-1)] \quad \text{additions} \qquad 3.2$$

are required.

Assuming that the error estimate is not calculated, t is

calculated by dividing $R_1^{(2)}$ by $R_1^{(1)}$. $R_1^{(2)}$ and $R_1^{(1)}$ each require n multiplications, n additions, and one subtraction to be computed.

The solution $\hat{x}$ can then be calculated with n divisions and 2n additions. One subtraction must be done here to calculate 1 - t.

The total number of operations required then to apply the One-Dimensional Reduction by Projection Method not counting those required to solve the systems in 3.1 are:

additions and subtractions

$$\frac{(n-1)(n-1)}{2} + \frac{(n-1)}{2}n + 2(n-1)n + n(n-1) + 2(n+1) + 1 + 2n = \frac{n^3 + 5n^2 + 4n + 4}{2} \qquad 3.3$$

multiplications

$$\frac{(n-1)(n-1)}{2} + \frac{(n-1)}{2}n + 2(n-1)n + 2n + 1 + n = \frac{n^3 + 3n^2 + 2n + 2}{2} . \qquad 3.4$$

Adding in the totals given in 3.2 for the Gaussian elimination of B, this becomes:

$$\frac{5n^3 + 18n^2 - 11 + 30}{6} \text{ additions}$$

and

$$\frac{5n^3 + 15n^2 - 8n + 12}{6} \text{ multiplications.}$$

Again, according to Fox (1, p. 176),

$$n(\tfrac{1}{3}n^2 + \tfrac{2}{3} + n) \text{ multiplications and divisions}$$

and

$$n[\tfrac{1}{3}n^2 - \tfrac{1}{2}n + \tfrac{1}{6} + (n-1)] \text{ additions}$$

would be required to solve the system $Ax = b$ directly by Gaussian elimination.

The following table shows how much work is done to solve an n-dimensional system by the One-Dimensional Reduction by Projection Method using Gaussian elimination to solve the reduced system. It compares this with the number of operations needed to solve the original system by Guassian elimination.

Table 2. Operations in the One-Dimensional Reduction by Projection Method

System size	Gaussian elimination		One-Dimensional Reduction by Projection Method		Extra steps used in One-Dimensional Reduction by Projection Method	
	Additions	Multiplications	Additions	Multiplications	Additions	Multiplications
2	3	8	20	16	17	8
3	11	20	49	43	38	23
4	26	40	99	90	73	50
5	50	70	175	162	125	92
6	85	112	282	264	197	152
7	133	168	425	401	292	233
8	196	240	609	578	413	338
9	276	330	839	800	563	470
10	375	440	1,120	1,072	745	632
11	495	572	1,457	1,399	962	827
12	638	728	1,855	1,786	1,217	1,058
13	806	910	2,319	2,238	1,513	1,328
25	5,550	5,850	14,855	14,552	9,355	8,702
50	42,875	44,200	111,580	110,352	68,705	66,152
100	338,250	343,400	863,155	858,202	524,905	514,802

E. Comparisons

The following is a comparison of the total number of arithmetic operations required to solve an n-dimensional linear system by various methods. Results 1, 2, 3, and 4 are due to Mok Tokko (14, p. 29),

whereas result 5 comes from 3.3 and 3.4 of this paper. In 5, x represents the number of operations required to solve the two (n-1)-dimensional reduced systems, and in 1, 2, and 3, c represents the number of cycles required for the method to converge.

Method	Total number of operations
1. Gauss-Seidel	$c(2n^2 + n)$
2. 2-dimensional projection	$c(4n^2 + 4n) + (n^3 + 2n)$
3. 3-dimensional projection	$c(4n^2 + 5n) + (n^3 + 8n)$
4. Gaussian Elimination	$2n^3 - 2n^2 + 2n - 1$
5. One-Dimensional Reduction by Projection	$n^3 + 4n^2 + 3n + 3 + x$

Gauss-Seidel, used to solve the reduced system of the One-Dimensional Reduction by Projection Method, will definitely converge. If the number of cycles for which it converges is c_1, then the total number of operations required for the solution is:

$$n^3 + 4n^2 + 3n + 3 + c_1(4(n-1)^2 + (n-1)).$$

If

$$c_1 < \frac{n^3 - 6n^2 - n - 4}{c_1(4(n-1)^2 + (n-1))},$$

the amount of computation for One-Dimensional Reduction by Projection is less. If $n = 100$, c_1 must be less than about 1/4 n. Gauss-Seidel cannot be reasonably expected to converge in 25 iterations, but the better accuracy of an iterative method can be worth the expense of added calculations.

Now a comparison will be made between the 2-dimensional projection

method and the One-Dimensional Reduction by Projection Method using Gauss-Seidel to solve the reduced system. Let c_1 be the number of cycles to solve the reduced system by Gauss-Seidel, and let c_2 be the number of iterations to solve the system by the 2-dimensional projection method. If

$$c_1 < \frac{c_2(4n^2 + 4n) - 2n^2 - n - 3}{4n^2 - 6n - 6},$$

the amount of computation for the One-Dimensional Reduction by Projection Method is less than the computation for the 2-dimensional projection method.

If $n = 100$, the above equation becomes approximately

$$c_1 < 1.025\ c_2 - .510.$$

Experience has shown that, unless much computation is done in optimizing the 2-dimensional projection method, when Gauss-Seidel converges, it converges in fewer cycles than the projection method. Therefore, it can be reasonably expected that this last condition will be met.

F. Examples

Chapter I points out that Gauss-Seidel is useful in solving larger systems. Theorem 6 points out that Gauss-Seidel must converge on the system produced by the One-Dimensional Reduction by Projection Method. Therefore, the first two examples of this section are important because Gauss-Seidel diverges on the original system but is used to find the solution to the reduced system.

For the code used in all examples, see the Appendix.

Example 3:

$$A = \begin{bmatrix} 3 & 2 & 1 & 1 \\ -4 & 5 & -1 & 0 \\ 6 & 4 & 1 & 2 \\ -1 & 2 & 2 & 2 \end{bmatrix} \qquad b = \begin{bmatrix} 3 \\ 2 \\ 7 \\ -3 \end{bmatrix} \qquad \text{Solution} = \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix}$$

$$B = \begin{bmatrix} 62 & 8 & 11 \\ 8 & 49 & 5 \\ 11 & 5 & 7 \end{bmatrix} \qquad v^{(1)} = \begin{bmatrix} 46 \\ 38 \\ 2 \end{bmatrix} \qquad v^{(2)} = \begin{bmatrix} 32 \\ 24 \\ -5 \end{bmatrix}$$

$$x^{(1)} = \begin{bmatrix} 0.955904072 \\ 0.802025459 \\ -1.789296011 \\ 0 \end{bmatrix} \qquad x^{(2)} = \begin{bmatrix} 0.911808144 \\ 0.604050918 \\ -2.578592024 \\ 1 \end{bmatrix}$$

$$t_1 = 2.000000006$$

$$t_2 = 1.999999917$$

$$t_3 = 1.999999987$$

$$t_4 = 1.999999942$$

$$\hat{x} = \begin{bmatrix} 1.000000000 \\ 0.999999999 \\ -1.000000003 \\ -0.999999994 \end{bmatrix}$$

$$\|\varepsilon_y\| \leq 4 \times 10^{-9}$$

Example 4:

$$A = \begin{bmatrix} .4 & 0 & -.3 & .1 & -.3 & .2 & -.8 & .9 & 0 \\ .5 & .2 & .1 & .3 & .4 & 0 & .8 & .7 & .1 \\ 0 & -.2 & .1 & 0 & .2 & .1 & .3 & -.6 & .3 \\ .4 & .1 & 0 & -.7 & 0 & -.3 & 0 & .2 & .1 \\ .9 & 0 & -.1 & 0 & .2 & .4 & -.6 & 0 & .8 \\ .8 & .2 & .3 & .6 & 0 & .1 & 0 & .2 & .3 \\ 0 & .1 & 0 & -.4 & .9 & 0 & .3 & -.6 & 0 \\ .1 & .9 & -.8 & .8 & -.7 & 0 & -.4 & .3 & .1 \\ .2 & 0 & -.3 & .5 & -.2 & .2 & -.4 & 0 & .3 \end{bmatrix}$$

$$b = \begin{bmatrix} -.4, & -.1, & 0, & -.2, & .4, & 1.3, & .9, & .3, & .1 \end{bmatrix}$$

$$\text{Solution} = \begin{bmatrix} 1, & 1, & 1, & 1, & 1, & -1, & -1, & -1, & -1 \end{bmatrix}$$

$$B = \begin{bmatrix} 2.07 & .39 & -.06 & .57 & .15 & .44 & -.58 & .98 \\ .39 & .95 & -.66 & .79 & -.50 & -.03 & -.23 & .53 \\ -.06 & -.66 & .94 & -.61 & .75 & -.12 & .85 & -.44 \\ .57 & .79 & -.61 & 2 & -.93 & .39 & -.48 & .76 \\ .15 & -.50 & .75 & -.93 & 1.67 & 0 & 1.13 & -.86 \\ .44 & -.03 & -.12 & .39 & 0 & .35 & -.45 & .08 \\ -.58 & -.23 & .85 & -.48 & 1.13 & -.45 & 2.14 & -.64 \\ .98 & .53 & -.44 & .76 & -.86 & .08 & -.64 & 2.19 \end{bmatrix}$$

$$v^{(1)} = \begin{bmatrix} 1.16, & .58, & .19, & .78, & .74, & .29, & .11, & -.66 \end{bmatrix}$$

$$v^{(2)} = \begin{bmatrix} .04, & .46, & .31, & .41, & .61, & -.12, & .58, & -.66 \end{bmatrix}$$

$$x^{(1)} = \begin{bmatrix} 0.231008759 \\ 1.162949703 \\ 1.414908510 \\ 1.258609718 \\ 1.488939717 \\ -1.967732620 \\ -1.419178539 \\ -0.596850715 \\ 0 \end{bmatrix} \qquad x^{(2)} = \begin{bmatrix} -0.537982483 \\ 1.325899406 \\ 1.829817020 \\ 1.517219437 \\ -1.977879435 \\ -2.935465243 \\ -1.838357079 \\ -0.193701431 \\ 0 \end{bmatrix}$$

$$t_1 = 2.000000000$$
$$t_2 = 1.999999823$$
$$t_3 = 1.999999993$$
$$t_4 = 2.000000653$$
$$t_5 = 1.999999930$$
$$t_6 = 2.000000052$$
$$t_7 = 1.999999803$$
$$t_8 = 1.999999966$$
$$t_9 = 1.999999996$$

$$\hat{x} = \begin{bmatrix} 1.000000001 \\ 1.000000000 \\ 1.000000000 \\ 0.999999999 \\ 0.999999999 \\ -1.999999997 \\ -0.999999999 \\ -0.999999999 \\ -1.000000000 \end{bmatrix}$$

$$\|\epsilon_y\| = 2.55 \times 10^{-7}$$

The following example points out a problem that may be encountered with the method. Notice t_1, t_2 and t_3 are close to two. However, t_4 is much farther from two. Examining why this happens, we note that $R_4^{(1)} = .0000017$ and $R_4^{(2)} = .0000011$. In other words, both are close to zero, and the numbers reflect only truncation error. Therefore, care should be used that a t calculated meaninglessly from such a row is

not used as the t of the method. Also, it should be thrown out in computations of the error estimate.

Example 5:

$$A = \begin{bmatrix} 1 & 1 & 2 & 0 \\ -1 & 1 & 0 & 2 \\ 3 & -2 & 1 & 1 \\ 2 & 2 & 1 & 1 \end{bmatrix} \qquad b = \begin{bmatrix} 0 \\ -2 \\ -1 \\ 2 \end{bmatrix} \qquad \text{Solution} = \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix}$$

$$B = \begin{bmatrix} 15 & -2 & 7 \\ -2 & 10 & 2 \\ 7 & 2 & 6 \end{bmatrix} \qquad v^{(1)} = \begin{bmatrix} 3 \\ 4 \\ 1 \end{bmatrix} \qquad v^{(2)} = \begin{bmatrix} 0 \\ 2 \\ -1 \end{bmatrix}$$

$$x^{(1)} = \begin{bmatrix} .7333323 \\ .7333326 \\ -.9333315 \\ 0 \end{bmatrix} \qquad x^{(2)} = \begin{bmatrix} .4666660 \\ .4666663 \\ -.8666657 \\ -1 \end{bmatrix}$$

$$t_1 = 2.000001$$
$$t_2 = 2.000000$$
$$t_3 = 1.999991$$
$$t_4 = 0.647059$$

$$\hat{x} = \begin{bmatrix} .9999980 \\ .9999985 \\ -.9999972 \\ -.9999981 \end{bmatrix}$$

$$\|\varepsilon_y\| = 2.7 \times 10^{-6}$$

In the following example, the exact solutions to 7-place accuracy for $x^{(1)}$ and $x^{(2)}$ can be surmised to be:

$$x^{(1)} = \begin{bmatrix} 12.66667 \\ 12.66667 \\ 9.333333 \\ 0 \end{bmatrix} \qquad x^{(2)} = \begin{bmatrix} 12.40000 \\ 12.40000 \\ 9.400000 \\ 1 \end{bmatrix}$$

It can be seen that the computational error in computing $x^{(1)}$ and $x^{(2)}$ is reflected to the same order of magnitude in the error in computing $\hat{x}$. Example 6 has the same coefficient matrix as Example 5 and, therefore, has the same problem with t_4. Also, it presents a scaling problem with the constant vector.

Example 6:

$$A = \begin{bmatrix} 1 & 1 & 2 & 0 \\ -1 & 1 & 0 & 2 \\ 3 & -2 & 1 & 1 \\ 2 & 2 & 1 & 1 \end{bmatrix} \qquad b = \begin{bmatrix} 40 \\ 20 \\ 30 \\ 60 \end{bmatrix} \qquad \text{Solution} = \begin{bmatrix} 10 \\ 10 \\ 10 \\ 10 \end{bmatrix}$$

$$B = \begin{bmatrix} 15 & -2 & 7 \\ -2 & 10 & 2 \\ 7 & 2 & 6 \end{bmatrix} \qquad v^{(1)} = \begin{bmatrix} 230 \\ 120 \\ 170 \end{bmatrix} \qquad v^{(2)} = \begin{bmatrix} 227 \\ 118 \\ 168 \end{bmatrix}$$

$$x^{(1)} = \begin{bmatrix} 12.66665 \\ 12.66666 \\ 9.333347 \\ 0 \end{bmatrix} \qquad x^{(2)} = \begin{bmatrix} 12.39999 \\ 12.40000 \\ 9.400016 \\ 1 \end{bmatrix}$$

$$t_1 = .9000049$$
$$t_2 = .8999999$$
$$t_3 = .8999992$$
$$t_4 = .1212121$$

$$\hat{x} = \begin{bmatrix} 9.999968 \\ 9.999868 \\ 10.00006 \\ 10.00049 \end{bmatrix}$$

$$\|\varepsilon_y\| = 1.52 \times 10^{-4}$$

Since the matrix B is symmetric and positive definite, it can be solved by the Orthogonalization Method. The following example is solved in three different ways. The first applies the Fortran Compiler-supplied Gaussian Elimination with complete pivoting routine to solve the original system directly. The second applies the One-Dimensional Reduction by Projection Method using the same routine as the first to solve B. The third applies the One-Dimensional Reduction by Projection Method using orthogonalization to solve B. The corresponding residue from the first row is given. As can be seen, the residue from orthogonalization is the smallest. Also, as can be seen, the error bound for orthogonalization is smaller. The times used in calculating were recorded and were all 1/64 of a second. The author wrote the orthogonalization routine and makes no claim about its efficiency or accuracy. However, the Fortran routine has been carefully written by professionals. It appears that a carefully-written Orthogonalization routine paired with the One-Dimensional Reduction by Projection Method can be a useful tool in solving linear systems.

Example 7:

$$A = \begin{bmatrix} 101 & 3 & 6 & 4 & 7 & 1 \\ 1 & 96 & 3 & 2 & 4 & 6 \\ 3 & 2 & 201 & 4 & 1 & 3 \\ 1 & 2 & 3 & 176 & 1 & 1 \\ 2 & 2 & 1 & 2 & 93 & 6 \\ 1 & 2 & 3 & 7 & 2 & 316 \end{bmatrix} \qquad b = \begin{bmatrix} 98 \\ 88 \\ 199 \\ -172 \\ -96 \\ -320 \end{bmatrix}$$

$$B = \begin{bmatrix} 10217 & 413 & 1220 & 605 & 903 \\ 413 & 9241 & 722 & 582 & 599 \\ 1220 & 722 & 40465 & 1385 & 357 \\ 605 & 582 & 1385 & 31065 & 416 \\ 903 & 599 & 357 & 416 & 8720 \end{bmatrix}$$

$$V^{(1)} = \begin{bmatrix} 9899 \\ 7964 \\ 39279 \\ -31340 \\ -8503 \end{bmatrix} \qquad V^{(2)} = \begin{bmatrix} 9454 \\ 6733 \\ 37695 \\ -33768 \\ -9728 \end{bmatrix}$$

Solution by Gaussian Elimination:

$$\hat{x} = \begin{bmatrix} .9997231, & 1.000040, & 1.005026, & -1.000066, & -.9998398, & -1.003208 \end{bmatrix}$$

Time = 1/64 second

$$R_1^{\hat{x}} = 4.15 \times 10^{-5}$$

Solution by One-Dimensional Reduction by Projection Method using Gaussian Elimination to solve the reduced systems:

$$x^{(1)} = \begin{bmatrix} .9801563 \\ .8826002 \\ .9720404 \\ -1.072735 \\ -1.125862 \\ 0 \end{bmatrix} \qquad x^{(2)} = \begin{bmatrix} .9606512 \\ .7655345 \\ .9391600 \\ -1.145173 \\ -1.251480 \\ 1 \end{bmatrix}$$

$$t_1 = 1.996802$$
$$t_2 = 1.996802$$
$$t_3 = 1.996784$$
$$t_4 = 1.996854$$
$$t_5 = 1.996796$$
$$t_6 = 1.996799$$

$$\hat{x} = \begin{bmatrix} .9997231 \\ 1.000040 \\ 1.005026 \\ -1.000066 \\ - .9998398 \\ -1.003208 \end{bmatrix}$$

$$\|\varepsilon_y\| = 1.37 \times 10^{-6}$$

Time = 1/64 second

$$R_1^{\hat{x}} = -4.15 \times 10^{-5}$$

Solution by One-Dimensional Reduction by Projection Method using Orthogonalization to solve B:

$$x^{(1)} = \begin{bmatrix} .9801559 \\ .8825995 \\ .9720410 \\ -1.072737 \\ -1.125861 \\ 0 \end{bmatrix} \qquad x^{(2)} = \begin{bmatrix} .9606512 \\ .7655343 \\ .9391606 \\ -1.145174 \\ -1.251480 \\ 1 \end{bmatrix}$$

$$t_1 = 1.996785$$
$$t_2 = 1.996790$$
$$t_3 = 1.996776$$
$$t_4 = 1.996806$$
$$t_5 = 1.996822$$
$$t_6 = 1.996799$$

$$\hat{x} = \begin{bmatrix} .9997236 \\ 1.000042 \\ 1.005027 \\ -1.000066 \\ -.9998371 \\ -1.003224 \end{bmatrix}$$

$$\|\varepsilon_y\| = 9.03 \times 10^{-7}$$

Time = 1/64 second

$$R_1^{\hat{x}} = -2.39 \times 10^{-5}$$

The following example demonstrates a technique that may be useful in improving the results from the One-Dimensional Reduction by Projection Method. The systems $Bdx = V^{(1)}$ and $Bdx = V^{(2)}$ are solved by the Gauss-Seidel iteration which fails to converge on the system $Ax = b$. The residues $R^{(1)}$ and $R^{(2)}$ are given. Notice the residues $R_1^{(1)}$ and $R_1^{(2)}$ are quite small, t should have a value of two, and the error in t_1 is quite large. If the assumption is made that the least error in calculating t will be obtained from the largest component of the residue $R^{(1)}$, then t_5 should have the least error. In this example, t_5 does in fact have the least error of all the t's. $\hat{x}_{t_1}$ is $\hat{x}$ using t_1 for t, $\hat{x}_{t_5}$ is $\hat{x}$ using t_5 for t, and $\hat{x}_{GE}$ is $\hat{x}$ using Gaussian Elimination to solve the original system. Notice $\hat{x}_{t_5}$ has less error than $\hat{x}_{t_1}$.

Also, $\hat{x}_{t_1}$ and $\hat{x}_{t_5}$ compare quite favorably to $\hat{x}_{GE}$. $x_n^{(2)}$ was picked to be -1 this time for $y_n = 1$ and if $x_n^{(2)}$ is picked to be 1,

then $x^{(2)}$ must equal $\hat{x}$, giving an unfair advantage to the One-Dimensional Reduction by Projection Method.

Perhaps a useful technique, if one position of the solution were known (say y_n), would be to pick $x_n^{(1)}$ as y_n. The solution $\hat{x}$ then would be $x^{(1)}$ and only the system $Bdx = V^{(1)}$ would have to be solved.

Example 8:

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

$$b = \begin{bmatrix} 2.4 \\ 2.2 \\ 1.2 \\ 1.7 \\ 1.6 \\ 1.5 \\ .7 \\ .9 \\ 1.5 \end{bmatrix} \qquad \text{Solution} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$B = \begin{bmatrix} .61 & .20 & .15 & .58 & .15 & .20 & .14 & .18 \\ .20 & .46 & .20 & .15 & .49 & .50 & .17 & .44 \\ .15 & .20 & .41 & .13 & .19 & .18 & .35 & .16 \\ .58 & .15 & .13 & .61 & .15 & .18 & .11 & .17 \\ .15 & .49 & .19 & .15 & .62 & .61 & .16 & .54 \\ .20 & .50 & .18 & .18 & .61 & .63 & .14 & .53 \\ .14 & .17 & .35 & .11 & .16 & .14 & .33 & .15 \\ .18 & .44 & .16 & .17 & .54 & .53 & .15 & .49 \end{bmatrix}$$

$$V^{(1)} = \begin{bmatrix} 2.79 \\ 2.76 \\ 1.86 \\ 2.66 \\ 3.04 \\ 3.15 \\ 1.63 \\ 2.83 \end{bmatrix} \qquad V^{(2)} = \begin{bmatrix} 3.37 \\ 2.91 \\ 1.95 \\ 3.24 \\ 3.17 \\ 3.33 \\ 1.71 \\ 3.00 \end{bmatrix}$$

$$x^{(1)} = \begin{bmatrix} 1.799151 \\ 0.202765 \\ 1.981502 \\ 1.058986 \\ -0.730883 \\ 1.442227 \\ -0.091087 \\ 3.191485 \\ 0 \end{bmatrix} \qquad x^{(2)} = \begin{bmatrix} 2.598289 \\ -0.594464 \\ 2.962997 \\ 1.117982 \\ -2.461781 \\ 1.884465 \\ -1.182164 \\ 5.382971 \\ -1 \end{bmatrix}$$

$$R^{(1)} = \begin{bmatrix} -.0066540 \\ -.0033271 \\ .0239743 \\ .0232897 \\ -.0394360 \\ .0173206 \\ .0147769 \\ -.0206475 \\ .0026419 \end{bmatrix} \qquad R^{(2)} = \begin{bmatrix} -.0133083 \\ .0066543 \\ .0479491 \\ .0465795 \\ -.0788720 \\ .0346411 \\ .0295530 \\ -.0412952 \\ .0052841 \end{bmatrix}$$

$t_1 = 2.000025$
$t_2 = 1.999979$
$t_3 = 2.000019$
$t_4 = 2.000002$
$t_5 = 1.999999$
$t_6 = 1.999997$
$t_7 = 1.999946$
$t_8 = 2.000005$
$t_9 = 2.000072$

$$\hat{x}_{t_1} = \begin{bmatrix} 1.000032 \\ 0.999973 \\ 1.000031 \\ 0.999991 \\ 0.999972 \\ 1.000000 \\ 0.999963 \\ 1.000049 \\ 0.999975 \end{bmatrix} \qquad \hat{x}_{t_5} = \begin{bmatrix} 1.000011 \\ 0.999994 \\ 1.000031 \\ 0.999989 \\ 1.000018 \\ 0.999989 \\ 0.999992 \\ 0.999992 \\ 1.000001 \end{bmatrix}$$

$$\|\varepsilon_{\hat{x}_1}\| \leq 6.9 \times 10^{-5} \qquad \|\varepsilon_{\hat{x}_2}\| \leq 3.1 \times 10^{-5}$$

$$\hat{x}_{GE} = \begin{bmatrix} 0.999990 \\ 1.000007 \\ 0.999991 \\ 1.000000 \\ 1.000007 \\ 1.000001 \\ 1.000009 \\ 0.999982 \\ 1.000009 \end{bmatrix}$$

$$\|\varepsilon_{GE}\| = 1.8 \times 10^{-5}$$

$$\|\varepsilon_y\| = 1.0 \times 10^{-4}$$

IV. THE DIRECT PROJECTION METHOD

A. Overview

The Direct Projection Method takes a system of size n

$$Ax = b$$

and reduces it to two systems each of size $n-1$:

$$\begin{aligned} Bdx_B &= V_B^{(1)} \\ Bdx_B &= V_B^{(2)}. \end{aligned} \tag{4.1}$$

These two systems are broken down into four systems of size $n-2$:

$$\begin{aligned} Cdx_C &= V_C^{(1)} \\ Cdx_C &= V_C^{(2)} \\ Cdx_C &= V_C^{(3)} \\ Cdx_C &= V_C^{(4)}. \end{aligned} \tag{4.2}$$

This then is continued until 2^{n-1} systems of size 1 are obtained:

$$\begin{aligned} Hdx_H &= V_H^{(1)} \\ Hdx_H &= V_H^{(2)} \\ &\vdots \\ Hdx_H &= V_H^{(2^{n-1})}. \end{aligned} \tag{4.3}$$

The latter can then be solved by dividing $V_H^{(i)}$ by H.

The error estimate of Chapter III, Part A can be applied merely by retaining the storage of A. The condition number considerations of

Chapter III, Part B apply at each reduction.

B. Storage

The amount of storage needed to apply the Direct Projection Method will be calculated in this section. Only the storage needed in excess of that required for the original system $Ax = b$ will be counted.

To calculate the error estimate, B cannot be stored over the last $n-1$ rows of A. Therefore, $(n-1)(n+1)$ positions are needed for the systems (4.2).

Now, the systems (4.3) can be stored over the last $n-2$ rows of the system B, but $n-2$ more positions are needed. These are necessary because the systems (4.3) are $(n-2) \times (n-2)$ with four right-hand sides. Temporary storage is needed for the overlaying, but positions used to store the solution x could be used for this purpose. For the next reduction, the system would be $(n-3) \times (n-3)$ with eight right-hand sides. Storing this over the last $n-3$ rows of C which was overlaid on B would require an additional $4(n-3)$ positions. This continues until $(2^{n-1}+1) - (n+1)$ more positions are needed for the 1×1 systems H. The above totals

$$(n-1)(n+1) + \sum_{i=1}^{n-2} (n-1)(2^{i+1} - i - 2).$$

Now, to cut down on the amount of storage and operations, assume the guesses in the last positions of the $x^{(1)}$ vectors are zero and the last positions of the $x^{(2)}$ vectors are one. So at most $2^{n-2}+1$ t's must be calculated. In case n is 2 or 3, $2^{n-1} < n$. Since at least n t's must be calculated in the last step to produce the error bound, 1 is

added to 2^{n-2}.

The solutions $x^{(1)}$ and $x^{(2)}$ can be written over the matrix they solve. However, 2n positions are needed for computing the residues and n positions are needed for the solution vector x. The residues to calculate the error estimate can be written over the matrix B.

The above totals

$$S_E = 3n + 2 + 2^{n-2} + (n-1)x(n+1) + \sum_{i=1}^{n-2} (n-1-i)x(2^{i+1}-i+2).$$

Without the error estimate, B can be written over A and only

$$S = 3n + 1 + \sum_{i=1}^{n-2} (n-1-i)x(2^{i+1}-i+2) \text{ positions}$$

are needed. The following table lists the S_E and S values for various n.

Table 3. Storage in the Direct Projection Method

System size	Amount required to store system	Excess required to get error estimate	Excess required with no error estimate
2	6	5	2
3	12	26	15
4	20	51	31
5	30	95	62
6	42	175	123
7	56	326	245
8	72	619	491
9	90	1,197	988
10	110	2,347	1,991
11	132	4,644	4,011
12	156	9,239	8,071
13	182	18,435	16,218
25	650	75,496.324	67,107,092
50	2,550	2.53×10^{15}	2.25×10^{15}
100	10,100	2.85×10^{30}	2.63×10^{30}

As can be seen from the above table, storage is a problem for larger systems.

C. Operations

As with the One-Dimensional Reduction by Projection Method (see Chapter III, Part D), the efficiency of the Direct Projection Method in solving an $n \times n$ linear system is measured by counting arithmetic operations. As before, additions and subtractions will be counted together and called additions, and multiplications and divisions will be counted together and called multiplications.

To cut down on the computation of the residues R_0 and R_1 at each step, the guesses at each step will be assumed to be

$$x_0^{(1)} = (0, 0, \cdots, 0)$$

and

$$x_0^{(2)} = (0, 0, \cdots, 0, 1).$$

Since B is symmetric, $\frac{(n-1)(n-1)}{2} + \frac{(n-1)}{2}$ positions must be calculated. Each position is a dot product of two n-dimensional columns of A. Each dot product requires n additions and n multiplications. The residue vector R_0 equals b, so no operations are required for its computation. The residue vector R_1 equals $b-a_n$, so n subtractions are required for its computation.

Now, two (n-1)-dimensional vectors $V_B^{(1)}$, $V_B^{(2)}$ which form the system B must be calculated. This is done with $(n-1)n$ additions and $(n-1)n$ multiplications.

The above totals:

$$\frac{(n-1)(n-1) + (n-1)}{2} + 2(n-1)n \text{ multiplications}$$

and 4.4

$$\frac{(n-1)(n-1) + (n-1)}{2} + 2(n-1)n + n \text{ additions.}$$

To reduce the system B to the system C, the calculation is approximately the same as above except now four right-hand sides must be calculated. The dimensions of C are $(n-2)x(n-2)$, and each dot product requires $n-1$ additions and $n-1$ multiplications.

Call the four right-hand sides $V_C^{(0)}$, $V_C^{(1)}$, $V_C^{(2)}$, and $V_C^{(3)}$. Each position of $V_C^{(0)}$ is calculated by the dot product of $V_B^{(0)}$ and a column of B; each position of $V_C^{(1)}$ is calculated by the dot product of $V_B^{(0)} - B_{n-1}$ and a column of B; $V_C^{(2)}$ is calculated by the dot product of $V_B^{(1)}$ and a column of B; and $V_C^{(3)}$ is calculated by the dot product of $V_B^{(1)} - B_{n-1}$ and a column of B.

The total number of operations required to get the system C, then, is:

$$\frac{(n-2)(n-2) + (n-2)}{2} + 4(n-2)(n-1) \text{ multiplications}$$

and 4.5

$$\frac{(n-2)(n-2) + (n-2)}{2} + 4(n-2)(n-1) + 2(n-1) \text{ additions.}$$

This is continued until the 1×1 system H with 2^{n-1} right-hand sides is calculated. H has one position that is obtained from the dot product of a 2-dimensional vector with itself. Each right-hand side position is obtained as the dot product of two 2-dimensional vectors. As for the calculation of system C above, the even-numbered $V_H^{(i)}$'s require additional subtractions.

This totals:

$$[1 + 2^{n-1}](2) \text{ multiplications}$$

and 4.6

$$[1 + 2^{n-1}](2) + 2^{n-2}(2) \text{ additions.}$$

The sum of 4.4, 4.5, and 4.6 is:

$$\sum_{j=1}^{n-1} [\frac{(n-j)(n-j) + (n-j)}{2} + 2^{j}(n-j)](n-j+1) \text{ multiplications}$$

and 4.7

$$\sum_{j=1}^{n-1} [\frac{(n-j)(n-j) + (n-j)}{2} + 2^{j}(n-j)(n-j+1) + 2^{j-1}(n-j+1) \text{ additions.}$$

Now the system H is solved by 2^{n-1} divisions. From this solution, 2^{n-2} t's must be calculated. Each t is calculated by dividing one residue by another. Each of these residues is computed by two additions, two multiplications, and one subtraction.

From these t's, $2^{n-2}x$ must be calculated by 2^{n-2} subtractions to get the (1-t)'s $2 \times 2^{n-2}$ divisions and $2 \times 2 \times 2^{n-2}$ subtractions.

Thus, the solutions to the 2×2 systems are obtained. The above solutions are then used to get the solutions to the 3×3 systems with 2^{n-3} right-hand sides. 2^{n-3} t's must be calculated. Each of these t's is calculated by dividing one residue by another. Each of these residues is computed by three additions, three multiplications, and one subtraction. From these t's, $2^{n-3}\hat{x}$'s must be calculated by 2^{n-3} subtractions to get the (1-t)'s, $3 \times 2^{n-3}$ divisions, and $2 \times 3 \times 2^{n-3}$ subtractions. This then is the solution to the 3×3 systems.

This process is continued until the solution to the $n \times n$ is

obtained. The sum for this is:

$$2^{n-1} + \sum_{j=1}^{n-1} 2^{n-j-1}(1+2(j+1) + (n+1)) \quad \text{multiplications}$$

and 4.8

$$\sum_{j=1}^{n-1} 2^{n-j-1}(2+2(j+1) + 1 + 2(j+1)) \quad \text{additions.}$$

The grand total of additions and multiplications, then, is the sum of 4.7 and 4.8:

$$2^{n-1} + \sum_{j=1}^{n-1} 2^{n-j-1}(4+2j) + \frac{(n-j)^2 + (n-j)}{2} + 2^j(n-j)(n-j+1) \quad \text{multiplications}$$

and

$$\sum_{j=1}^{n-1} 2^{n-j-1}(7+4j) + \frac{(n-j)^2 + (n-j)}{2} + 2^j(n-j)(n-j+1) + 2^{j-1}(n-j+1) \quad \text{additions.}$$

In the following table, these results are listed for various system sizes. As a comparison, the number of operations for Gaussian Elimination is also given.

Table 4. Operations in the Direct Projection Method

System size	Gaussian Elimination		Direct Projection Method	
	Additions	Multiplications	Additions	Multiplications
2	3	8	19	15
3	11	20	75	59
4	26	40	210	168
5	50	70	503	407
6	85	112	1,104	898
7	133	168	2,302	1,874
8	196	240	4,661	3,787
9	276	330	9,292	7,524
10	375	440	18,397	14,839
11	495	572	36,357	29,217
12	638	728	71,908	57,602
13	806	910	142,493	113,853
25	5,500	5,850	5.705×10^8	4.53×10^8
50	42,875	44,200	1.91×10^{16}	1.52×10^{16}
100	338,250	343,155	2.15×10^{31}	1.71×10^{31}

The above table shows that the Direct Projection Method requires a very large number of operations to solve larger systems.

D. Examples

The two examples of this section demonstrate the Direct Projection Method. At the same time, they demonstrate that repeated reduction of the system has a tendency to make the numbers in the reduced matrices very large. This can increase inaccuracies due to computing.

In the following examples, $\hat{x}_D^{(i)}$ indicates the solution to the system $D = V_D^{(i)}$; the vectors

$$\begin{bmatrix} x_D^{(1)} \\ 0 \end{bmatrix} \qquad \begin{bmatrix} x_D^{(2)} \\ 1 \end{bmatrix}$$

are used as the $x^{(0)}$ and $x^{(1)}$ (see Chapter I) to solve the systems $C = V_C^{(1)}$; and the other x's are paired off in a similar manner.

Let $t_{C1}^{(1)}$ indicate the t calculated to solve the system $C = V_C^{(1)}$. The 1 next to the C indicates that the t comes from the ratio of the residues of the first row. Let the other t's be similarly denoted.

Example 9:

$$A = \begin{bmatrix} 5 & 1 & 2 & 2 \\ 3 & 5 & -1 & 3 \\ 1 & -3 & 5 & 7 \\ 4 & 1 & 0 & 5 \end{bmatrix} \qquad b = \begin{bmatrix} 2 \\ 6 \\ -14 \\ 0 \end{bmatrix} \qquad \text{Solution} = \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix}$$

$$B = \begin{bmatrix} 51 & 21 & 12 \\ 21 & 36 & -18 \\ 12 & -18 & 30 \end{bmatrix} \qquad V_B^{(1)} = \begin{bmatrix} 14 \\ 74 \\ -72 \end{bmatrix} \qquad V_B^{(2)} = \begin{bmatrix} -32 \\ 73 \\ -108 \end{bmatrix}.$$

$$C = \begin{bmatrix} 3186 & 1611 \\ 1611 & 2161 \end{bmatrix} \qquad V_C^{(1)} = \begin{bmatrix} 1404 \\ 4254 \end{bmatrix} \qquad V_C^{(2)} = \begin{bmatrix} 810 \\ 5190 \end{bmatrix}$$

$$V_C^{(3)} = \begin{bmatrix} -1395 \\ 3900 \end{bmatrix} \qquad V_C^{(4)} = \begin{bmatrix} -1989 \\ 4836 \end{bmatrix}$$

$D = 12745917$ $\qquad V_D^{(1)} = 11326338$ $\qquad V_D^{(2)} = 2873421$

$V_D^{(3)} = 10941750$ $\qquad V_D^{(4)} = 2488833$ $\qquad V_D^{(5)} = 1838430$

$V_D^{(6)} = 6614487$ $\qquad V_D^{(7)} = 1453842$ $\qquad V_D^{(8)} = -6999075$

$x_D^{(1)} = 0.888624804$ $\qquad x_D^{(2)} = 0.225438546$

$x_D^{(3)} = 0.858451477$ $\qquad x_D^{(4)} = 0.195265119$

$x_D^{(5)} = 0.144236778$ $\qquad x_D^{(6)} = -0.51894948$

$x_D^{(7)} = 0.114063351$ $\qquad x_D^{(8)} = -0.549122907$

$t_{C1}^{(1)} = 0.648314204$ $\qquad t_{C1}^{(2)} = 0.739270329$

$t_{C2}^{(1)} = 0.648314204$ $\qquad t_{C2}^{(2)} = 0.739370439$

$t_{C1}^{(3)} = 0.729360457$ $\qquad t_{C1}^{(4)} = 0.786639103$

$t_{C2}^{(3)} = 0.729360457$ $\qquad t_{C2}^{(4)} = 0.786639103$

$$\hat{x}_C^{(1)} = \begin{bmatrix} -0.997110318 \\ 2.84344722 \end{bmatrix} \qquad \hat{x}_C^{(2)} = \begin{bmatrix} -1.685126636 \\ 3.835390107 \end{bmatrix}$$

$$\hat{x}_C^{(3)} = \begin{bmatrix} -2.306204318 \\ 3.694951555 \end{bmatrix} \qquad \hat{x}_C^{(4)} = \begin{bmatrix} -2.994220636 \\ 4.686894442 \end{bmatrix}$$

$$t_{B1}^{(1)} = 1.439285714 \qquad t_{B1}^{(2)} = 1.28146453$$
$$t_{B2}^{(1)} = 1.439285714 \qquad t_{B2}^{(2)} = 1.281464531$$
$$t_{B3}^{(1)} = 1.439285714 \qquad t_{B3}^{(2)} = 1.28146453$$

$$\hat{x}_B^{(1)} = \begin{bmatrix} 0.569105692 \\ 0.585365853 \\ -2.276422765 \end{bmatrix} \qquad \hat{x}_B^{(2)} = \begin{bmatrix} 0.138211383 \\ 0.170731706 \\ -3.552845529 \end{bmatrix}$$

$$t_{A1}^{(1)} = 2.000000000$$
$$t_{A2}^{(1)} = 2$$
$$t_{A3}^{(1)} = 2.000000002$$
$$t_{A4}^{(1)} = 2.000000000$$

$$y = \begin{bmatrix} 1 \\ 0.999999999 \\ -1 \\ -1 \end{bmatrix}$$

Example 10:

$$A = \begin{bmatrix} 1 & 1 & 2 & 0 \\ -1 & 1 & 0 & 2 \\ 3 & -2 & 1 & 1 \\ 2 & 2 & 1 & 1 \end{bmatrix} \qquad b = \begin{bmatrix} 0 \\ -2 \\ -1 \\ 2 \end{bmatrix}$$

$$B = \begin{bmatrix} 15 & -2 & 7 \\ -2 & 10 & 2 \\ 7 & 2 & 6 \end{bmatrix} \qquad V_B^{(1)} = \begin{bmatrix} 3 \\ 4 \\ 1 \end{bmatrix} \qquad V_B^{(2)} = \begin{bmatrix} 0 \\ 2 \\ -1 \end{bmatrix}$$

$$C = \begin{bmatrix} 278 & -36 \\ -36 & 108 \end{bmatrix} \qquad V_C^{(1)} = \begin{bmatrix} 44 \\ 36 \end{bmatrix} \qquad V_C^{(2)} = \begin{bmatrix} -99 \\ 18 \end{bmatrix}$$

$$V_C^{(3)} = \begin{bmatrix} -11 \\ 18 \end{bmatrix} \qquad V_C^{(4)} = \begin{bmatrix} -154 \\ 0 \end{bmatrix}$$

$D = 78580 \qquad V_D^{(1)} = 10936 \qquad V_D^{(2)} = 24832$

$V_D^{(3)} = -28170 \qquad V_D^{(4)} = -14274 \qquad V_D^{(5)} = -3706$

$V_D^{(6)} = 10190 \qquad V_D^{(7)} = -42812 \qquad V_D^{(8)} = -28916$

$\hat{x}_D^{(1)} = 0.139170272 \qquad \hat{x}_D^{(2)} = 0.316009163$

$\hat{x}_D^{(3)} = -0.358488165 \qquad \hat{x}_D^{(4)} = -0.181649275$

$\hat{x}_D^{(5)} = -0.047162128 \qquad \hat{x}_D^{(6)} = 0.129676763$

$\hat{x}_D^{(7)} = -0.544820565 \qquad \hat{x}_D^{(8)} = -0.367981675$

$t_{C1}^{(1)} = -1.478260869 \qquad t_{C1}^{(2)} = -18.95$

$t_{C2}^{(1)} = -1.478260870 \qquad t_{C2}^{(2)} = -18.95000000$

$t_{C1}^{(3)} = -5.234275 \qquad t_{C1}^{(4)} = 6.181818181$

$t_{C2}^{(3)} = -5.234375 \qquad t_{C2}^{(4)} = 6.181818182$

$$\hat{x}_C^{(1)} = \begin{bmatrix} 0.210526316 \\ 0.403608772 \end{bmatrix} \qquad \hat{x}_C^{(2)} = \begin{bmatrix} -0.349624060 \\ 0.050125313 \end{bmatrix}$$

$$\hat{x}_C^{(3)} = \begin{bmatrix} -0.018796992 \\ 0.160401003 \end{bmatrix} \qquad \hat{x}_C^{(4)} = \begin{bmatrix} -0.578947368 \\ 0.192982456 \end{bmatrix}$$

$$t_{B1}^{(1)} = 2.071428571 \qquad t_{B1}^{(2)} = 2.153846154$$
$$t_{B2}^{(1)} = 2.071428573 \qquad t_{B2}^{(2)} = 2.153846154$$
$$t_{B3}^{(1)} = 2.071428571 \qquad t_{B3}^{(2)} = 2.153846154$$

$$\hat{x}_B^{(1)} = \begin{bmatrix} 0.733333333 \\ 0.733333333 \\ -0.933333333 \end{bmatrix} \qquad \hat{x}_B^{(2)} = \begin{bmatrix} 0.466666667 \\ 0.466666667 \\ -0.866666667 \end{bmatrix}$$

$$t_{A1}^{(1)} = 2$$
$$t_{A2}^{(1)} = 2.000000000 \qquad y = \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix}$$
$$t_{A3}^{(1)} = 2$$
$$t_{A4}^{(1)} = 0.25$$

The constant $t_{A4}^{(1)}$ is so inaccurate because it was calculated from small residues. (See Example 4.)

V. FUTURE RESEARCH AND CONCLUSIONS

A. Future Research

The following questions offer possibilities for future research.

How does the One-Dimensional Reduction by Projection compare to other methods as to time and accuracy? Of particular interest is how this method compares to the various projection methods.

Are there certain types of matrices for which this method works particularly well?

The problem of condition number can be looked at more closely. For example, if $K(A)$ is large, is $K(B)$ necessarily smaller? Are there certain matrices for which $K(B)$ is necessarily smaller than $K(A)$?

Can the One-Dimensional Reduction by Projection Method be useful in solving nonlinear systems?

Does the choice of which $n-1$ columns are used affect the performance of the method?

Would it be useful to consider a method somewhere between the One-Dimensional Reduction by Projection Method and the Direct Projection Method? That is, would it be useful to reduce an n-dimensional system down to four (n-2)-dimensional systems and solve those?

Can a way be found to pick the n^{th} positions of the vectors $x_0^{(1)}$ and $x_0^{(2)}$ such that the solution is guaranteed to be more accurate?

Can a way be found to cut down on the storage required in the Direct Projection Method to make it more useful?

The author wishes to emphasize that many of these questions could

be answered by simply running many more examples of larger systems.

B. Conclusions

The Direct Projection Method is quite accurate, but, for all except small systems, requires too much storage and too many operations to be anything but theoretically important. With additional research, perhaps it could become a practical method of solving linear systems.

The One-Dimensional Reduction by Projection Method, on the other hand, is a straightforward and general technique that is easily implemented. The matrix B of the One-Dimensional Reduction by Projection Method is a submatrix of the matrix A^TA, and A^TA is also symmetric and positive definite. Fox (1, p. 42) says A^TA can be significantly more illconditioned than A. In general, the condition number of A^TA is at least as big as the condition number of A. Example 1 has shown that the matrix B can have a lower condition number than the matrix A. Of course, the condition number of B then is less than the condition number of A^TA. Thus, the One-Dimensional Reduction by Projection Method using a direct method to solve B can work better than a direct method used on the matrix A. Perhaps the choice of which column to delete would guarantee a reduction in condition number. An obvious choice for this would be to delete one of two columns that might be close to parallel. Thus, the One-Dimensional Reduction by Projection Method could always work better than other direct methods.

Gauss-Siedel is guaranteed to converge on A^TA as well as on B. The following example shows a case where the spectral radius of the Gauss-Siedel iteration matrix $(D+L)^{-1}U$ is smaller for B than it is

for A^TA. Thus, the One-Dimensional Reduction by Projection Method using Gauss-Siedel to solve B can be better than using Gauss-Siedel to solve A^TA.

Example 11:

$$A = \begin{bmatrix} 1 & 3 & 1 \\ 1 & -1 & 2 \\ 1 & 4 & 1 \end{bmatrix} \qquad A^TA = \begin{bmatrix} 3 & 6 & 4 \\ 6 & 26 & 5 \\ 4 & 5 & 6 \end{bmatrix}$$

$$(D+L)^{-1}U = \begin{bmatrix} 0 & 2 & 4/3 \\ 0 & -6/13 & -3/26 \\ 0 & -37/39 & -14,469/18,252 \end{bmatrix}$$

Spectral radius = .99

$$B = \begin{bmatrix} 3 & 6 \\ 6 & 26 \end{bmatrix} \qquad (D+L)^{-1}U = \begin{bmatrix} 0 & 2 \\ 0 & -6/13 \end{bmatrix}$$

Spectral radius = .46

Thus, the One-Dimensional Reduction by Projection Method can be a powerful tool in solving linear systems.

VI. BIBLIOGRAPHY

1. Fox, L. An Introduction to Numerical Linear Algebra. New York: Oxford University Press, 1956.

2. Georg, Dennis D. Criteria for Ordering Columns in Two-Dimensional Projection Methods. Unpublished M.S. thesis. Ames, Ia.: Library, Iowa State University, 1973.

3. Henrici, Peter. Elements of Numerical Analysis. New York: John Wiley and Sons, 1964.

4. Householder, A. S. Principles of Numerical Analysis. New York: McGraw-Hill Book Company, 1953.

5. Householder, A. S. "Some Numerical Methods for Solving Systems of Linear Equations." Amer. Math. Monthly, 57 (1950), 453-459.

6. Isaacson, Eugene and Herbert Bishop Keller. Analysis of Numerical Methods. New York: John Wiley and Sons, 1966.

7. Keller, R. F. "A Single-Step Gradient Method for Solving Systems of Linear Equations." University of Missouri at Columbia, Math. Sci. Tech. Report No. 8, 1964.

8. Keller, R. F., H. D. Pyron, and I-Ming Shen. "On Determining an Optimal 2-Dimensional Projection Method." Document Library, Ames Laboratory, USAEC, Ames, Iowa, Report No. IS-2755, 1968.

9. MacEachern, Alexander. A Generalized Projection Method for Systems of Non-Linear Equations. Unpublished Ph.D. thesis. Ames, Ia.: Library, Iowa State University, 1971.

10. MacEachern, Alexander and Roy F. Keller. "A Projection Method for Solving Non-Linear Systems of Equations." Document Library, Ames Laboratory, USAEC, Ames, Iowa, Report No. IS-3040, 1972.

11. Ortega, J. M. and W. C. Rheinboldt. Iterative Solution of Non-Linear Equations in Several Variables. New York: Academic Press, 1970.

12. Pyron, H. D. A Non-Stationary Two-Dimensional Acceleration for the One-Dimensional Projection Method. Unpublished Ph.D. thesis. Ames, Ia.: Library, Iowa State University, 1970.

13. Shen, I-Ming. Acceleration of the Projection Method for Solving Systems of Linear Equations. Unpublished Ph.D. thesis. Ames, Ia.: Library, Iowa State University, 1970.

14. Tokko, Mok. Optimal Three-Dimensional Projection Method for Solving Linear Algebraic Equations. Unpublished Ph.D. thesis. Ames, Ia.: Library, Iowa State University, 1972.

15. Varga, R. S. Matrix Iterative Analysis. Englewood Cliffs, N.J.: Prentice-Hall, 1962.

16. White, Michael William. A Class of Projection Methods for Solving Systems of n Non-Linear Equations in n Unknowns Allowing Projections of Order One through n. Unpublished Ph.D. thesis. Ames, Ia.: Library, Iowa State University, 1971.

17. White, Paul A. Linear Algebra. Belmont, Calif.: Dickinson Publishing Company, Inc., 1966.

18. Wilansky, Albert. Functional Analysis. New York: Blaisdell Publishing Company, 1964.

VII. APPENDIX

A. H.P. 9830 Basic for One-Dimensional Reduction by Projection Method

```
10 REM THIS PROGRAM WAS USED ON EXAMPLE 3.
20 REM CODE SIMILAR TO THIS WAS USED ON EXAMPLES 4 AND 8.
30 REM IT ATTEMPTS THE GAUSS-SEIDEL ITERATION ON THE SYSTEM AX = B
40 REM AND THEN APPLIES THE ONE-DIMENSIONAL REDUCTION BY PROJECTION METHOD
50 DIM A(9,9), K(9), B(8,10), Y(2,8), X(1,9)
60 REM Y(1,I) STORES THE SOLUTION TO THE SYSTEM BDX = V1
70 REM AND Y(2,I) STORES THE SOLUTION TO THE SYSTEM BDX = V2
80 REM B(I,9) STORES V1 AND B(I,10) STORES V2
90 REM X IS THE SOLUTION VECTOR
100 DATA 0.4, 0, -0.3, 0.1, -0.3, 0.2, -0.8, 0.9, 0
110 DATA 0.5, 0.2, 0.1, 0.3, 0.4, 0, 0.8, 0.7, 0.1
120 DATA 0, -0.2, 0.1, 0, 0.2, 0.1, 0.3, -0.6, 0.3
130 DATA 0.4, 0.1, 0, -0.7, 0, -0.3, 0, 0.2, 0.1
140 DATA 0.9, 0, -0.1, 0, 0.2, 0.4, -0.6, 0, 0.8
150 DATA 0.8, 0.2, 0.3, 0.6, 0, 0.1, 0, 0.2, 0.3
160 DATA 0, 0.1, 0, -0.4, 0.9, 0, 0.3, -0.6, 0
170 DATA 0.1, 0.9, -0.8, 0.8, -0.7, 0, -0.4, 0.3, 0.1
180 DATA 0.2, 0, -0.3, 0.5, -0.2, 0.2, -0.4, 0, 0.3
190 MAT READ A(9,9)
200 PRINT "A="
210 MAT PRINT A;
220 DATA -0.4, -0.1, 0, -0.2, 0.4, 1.3, 0.9, 0.3, 0.1
230 MAT READ K(9)
240 PRINT "K="
250 MAT PRINT K;
260 D=1
270 REM LINES 280 TO 310 COMPUTE B
280 MAT B=ZER
290 FOR I=1 TO 8
300 FOR J=1 TO 8
310 FOR K=1 TO 9
320 B(I,J)=B(I,J)+A(K,I)*A(K,J)
330 NEXT K
340 B(J,I)=B(I,J)
350 NEXT J
360 FOR K=1 TO 9
370 REM THE FOLLOWING ASSUMES X1 SUB 9=0 AND X2 SUB 9=1
380 B(I,9)=B(I,9)+A(K,I)*K(K)
390 B(I,10)=B(I,10)+A(K,I)*(K(K)-A(K,9))
400 NEXT K
410 NEXT I
420 PRINT "B="
```

```
430 MAT PRINT B;
440 REM LINES 350 TO 440 ATTEMPT GAUSS-SEIDEL ON AX=K
450 MAT X=ZER
460 FOR I=1 TO 9
470 S=0
480 FOR J=1 TO 9
490 IF J=I THEN 510
500 S=S+A(I,J)*X(1,J)
510 NEXT J
520 X(1,I)=(K(I)-S)/A(I,I)
530 NEXT I
540 PRINT "D=", D
550 MAT PRINT X;
560 REM D IS AN ITERATION COUNTER, INPUTTED FROM KEYBOARD
570 REM 999 INPUTTED SKIPS TO ONE-DIMENSIONAL REDUCTION BY PROJECTION
     METHOD
580 INPUT D
590 IF D=999 THEN
600 GO TO 460
610 D=1
620 REM LINES 630 TO 740 SOLVES BDX=V1 AND BDX=V2 BY G-S
630 MAT Y=ZER
640 FOR I=1 TO 8
650 S1=0
660 S2=0
670 FOR J=1 TO 8
680 IF J=I THEN 570
690 S1=S1+B(I,J)*Y(1,J)
700 S2=S2+B(I,J)*Y(2,J)
710 NEXT J
720 Y(1,I)=(B(I,9)-S1)/B(I,I)
730 Y(2,I)=(B(I,10)-S2)/B(I,I)
740 NEXT I
750 IF (D/10)=INT (D/10) THEN 790
760 D=D+1
770 GO TO 640
780 REM THE FOLLOWING CODE PRINTS OUT EVERY TENTH ITERATION
790 PRINT "D=", D
800 MAT PRINT Y;
810 D=D+1
820 GO TO 640
830 REM THIS CODE IS EXECUTED WHEN THE
840 REM DESIRED ACCURACY FOR Y IS REACHED
850 REM LINES 860 TO 990 CALCULATE AND PRINT T(I)
860 FOR I=1 TO 9
870 S1=0
880 S2=0
890 FOR J=1 TO 8
```

```
900 S1=S1+A(I,J)*Y(1,J)
910 S2=S2+A(I,J)*Y(2,J)
920 NEXT J
930 S2=S2+A(I,9)*Y(2,9)
940 S1=K(I)-S1
950 S2=K(I)-S2
960 C=S2/S1
970 IF I=1 THEN T=1-C
980 PRINT "T(",I,")=",C
990 NEXT I
1000 REM LINES 1010 TO 1050 CALCULATE AND PRINT THE SOLUTION X
1010 FOR I=1 TO 9
1020 X(1,I)=Y(1,I)-((Y(1,I)-Y(2,I))/T)
1030 NEXT I
1040 PRINT "THE SOLUTION IS"
1050 MAT PRINT X;
```

B. Fortran IV, G Level for One-Dimensional Reduction by Projection Method

```
C     THE FOLLOWING CODE WITH SLIGHT MODIFICATIONS WAS USED IN
C     EXAMPLES 5, 6, AND 7
C     K STORES THE CONSTANT VECTOR LITTLE B
C     THE FOLLOWING 3 LINES ARE CHANGED FOR A DIFFERENT SYSTEM SIZE
      REAL A(4,4), K(4), B(3,3), X1(4), S2(4)
      REAL R1(4), R2(4), V(3,2), T(4), Y(4)
      N=4
C     THE FOLLOWING LINE IS INSERTED HERE IF ORTHOGONALIZATION IS
C     USED TO SOLVE B
C     COMMON B,V
C     THE FOLLOWING 2 LINES SET X1(N) AND X2(N) TO DIFFERENT VALUES
      X1(N)=0
      X2(N)=1
      NM=N-1
C     THE SYSTEM TO BE SOLVED IS READ IN HERE
      DO 10 I=1,N
 10   READ(5,20)(A(I,J),J=1,N)
 20   FORMAT (10F10.2)
      READ (5,20)(K(I),I=1,N)
C     THE FOLLOWING LINES INITIALIZE THE CLOCK
      T=0.0
      X=0.0
      CALL STARTM(T,X)
C     X1 AND X2 ARE ZEROED EXCEPT IN NTH POSITION
      DO 30 I=1,NM
      X1(I)=0.0
 30   X2(I)=0.0
      DO 40 I=1,N
      R1(I)=K(I)-X1(N)*A(I,N)
 40   R2(I)=K(I)-X2(N)*A(I,N)
```

```
C     THE FOLLOWING CODE COMPUTES B
      DO 50 I=1,NM
      DO 50 J=I,NM
      B(I,J)=0.0
      DO 60 K=1,N
 60   B(I,J)=B(I,J)+A(K,I)*A(K,J)
 50   B(J,I)=B(I,J)
C     THE FOLLOWING CODE COMPUTES V(I,1) AND V(I,2)
      DO 70 I=1,NM
      V(I,1)=0.0
      V(I,2)=0.0
      DO 70 J=1,N
      V(I,1)=V(I,1)+R1(J)*A(J,I)
 70   V(I,2)=V(I,2)+R2(J)*A(J,I)
C     THE FOLLOWING CODE PRINTS OUT B,V(I,1) AND V(I,2)
      DO 80 I=1,NM
 80   WRITE (6,90)(AP(I,J),J=1,NM),V(I,1),V(I,2)
 90   FORMAT ('',10F10.2)
C     THE FOLLOWING LINES CALL G.E. TO SOLVE B
      EPS=1.0
      CALL GELG (V,B,NM,2,EPS,IER)
C     THE SOLUTION IS RETURNED IN V
C     THE FOLLOWING LINES WOULD BE INSERTED TO USE ORTHOGONALIZATION
C     TO SOLVE B
C     CALL ORTHO(NM)
C     THE SOLUTION WOULD BE RETURNED IN V
C     THE FOLLOWING CODE SET X1(I)=V(I,1) AND X2(I)=V(I,2)
      DO 100 I=1,NM
      X1(I)=V(I,1)
 100  X2(I)=V(I,2)
C     THE FOLLOWING LINES PRINT OUT X1 AND X2
      DO 110 I=1,N
 110  WRITE (6,120)X1(I),X2(I)
 120  FORMAT ('',2E14.7)
C     THE FOLLOWING CODE CALCULATES T(I)
      DO 130 I=1,N
      SUM1=0.0
      SUM2=0.0
      DO 140 J=1,N
      SUM1=SUM1+X1(J)*A(I,J)
 140  SUM2=SUM2+X2(J)*A(I,J)
      SUM1=K(I)-SUM1
      SUM2=K(I)-SUM2
      T(I)=SUM2/SUM1
 130  WRITE (6,150)J,T(J)
 150  FORMAT ('','T(',I2,')=',E14.7
C     THE FOLLOWING CODE CALCULATES THE SOLUTION Y
      DO 160 I=1,N
 160  Y(I)=X1(I)-(X1(I)-X2(I))/(1-T(1))
```

```
C     THE FOLLOWING CODE STOPS THE CLOCK AND PRINTS OUT TIME
      CALL STOPTM(T,X)
      WRITE (6, 170),X
 170  FORMAT ('','TIME=',E14.7)
C     THE FOLLOWING CODE PRINTS OUT A AND K
      WRITE (6, 180)
C     THE SPACING 40X IS 10*N AND CHANGES WITH SYSTEM SIZE
 180  FORMAT ('0','A=',40X,'B=')
      DO 190 I=1,N
 190  WRITE (6,200)(A(I,J),J=1,N),K(I)
 200  FORMAT ('',11F10.2)
C     THE FOLLOWING CODE PRINTS OUT THE SOLUTION
      WRITE (6,210)(Y(I),I=1,N)
 210  FORMAT ('0','SOLUTION',4X,10E14.7)
C     THE FOLLOWING SOLVES AX=K BY G.E. FOR COMPARISON
C     IT INITIALIZES THE CLOCK AND STOPS THE CLOCK AND PRINTS THE
C     SOLUTION
      T=0.0
      X=0.0
      CALL STARTM(T,X)
      CALL GELG(K,A,N,1,EPS,IER)
      CALL STOPTM(T,X)
      WRITE (6,130)(K(I),I=1,N)
      WRITE (6,170),X
      RETURN
      END
C     THE FOLLOWING IS THE ORTHOGONALIZATION ROUTINE
C     LINES 3 AND 4 ARE CHANGED DEPENDING ON SYSTEM SIZE
      SUBROUTINE ORTHO(N)
      INTEGER N
      REAL B(3,3),V(3,2),X(3,3),ALP(3),SUM(3),ALPH(3)
      COMMON B,V
      DO 10 I=1,N
      DO 10 J=1,N
 10   X(I,J)=0
      DO 20 I=1,N
      X(I,I)=1.0
      IH=I-1
      IF (IH.EQ.0) GO TO 50
      DO 30 J=1,IH
      SUM1=0
      DO 40 K=1,N
 40   SUM1=SUM1+X(K,J)*B(K,I)
      ALP(J)=SUM1/SUM(J)
      DO 30 K=1,N
 30   X(K,I)=X(K,I)-ALP(J)*X(K,J)
 50   CONTINUE
      SUM(I)=0
      DO 20 K=1,N
```

```
20    SUM(I)=SUM(I)+X(K,I)*B(K,I)
      DO 60 I=1,N
      SUM1=0
      SUM2=0
      DO 70 K=1,N
      SUM1=SUM1+X(K,I)*V(K,1)
70    SUM2=SUM2+X(K,I)*V(K,2)
      ALP(I)=SUM1/SUM(I)
60    ALPH(I)=SUM2/SUM(I)
      DO 80 I=1,N
      V(K,1)=0
80    V(I,2)=0
      DO 90 I=1,N
      DO 90 J=1,N
      V(K,1)=V(K,1)+ALP(J)*X(I,J)
90    V(K,2)-V(K,2)+ALPH(J)*X(I,J)
      RETURN
      END
```

C. H.P. 9830 Basic for Direct Parallel Residue Method

```
10 REM THIS PROGRAM WAS USED ON EXAMPLE 9
20 REM CODE SIMILAR TO THIS WAS USED ON EXAMPLE 10
30 REM A IS THE ORIGINAL 4X4 SYSTEM
40 REM THE CONSTANT MATRIX IS STORED IN A(1,5)
50 REM B IS THE FIRST REDUCTION OF A
60 REM B(I,4) AND B(I,5) STORE THE CONSTANT VECTORS OF B
70 REM C IS THE SECOND REDUCTION OF A
80 REM C(I,3),C(I,4),C(I,5), AND C(I,6) STORE THE CONSTANT VECTORS OF C
90 REM D IS THE THIRD AND LAST REDUCTION OF A
100 REM D(I,2)-D(I,9) STORE THE CONSTANT VECTORS OF D
110 REM V(2,8) STORES THE SOLUTION TO D, THAT IS, THE GUESS FOR C
120 REM U(3,4) STORES THE SOLUTION TO C, THAT IS, THE GUESS FOR B
130 REM T(4,2) STORES THE SOLUTION TO B, THAT IS, THE GUESS FOR A
140 REM S(4,1) STORES THE SOLUTION TO A
150 REM E(4,8) STORES THE CALCULATED T'S
160 REM R(4,8) STORES THE NEEDED RESIDUES
170 DIM A(4,5),B(3,5),C(2,6),D(1,9)
180 DIM S(4,1),T(4,2),U(3,4),V(2,8)
190 DIM E(4,8),R(4,8)
200 DATA 5,1,2,2,2
210 DATA 3,5,-1,3,6
220 DATA 1,-3,5,7,-14
230 DATA 4,1,0,5,0
240 MAT READ A(4,5)
250 PRINT "A="
```

```
260 MAT PRINT A;
270 MAT B=ZER
280 MAT C=ZER
290 MAT D=ZER
300 FOR I=1 TO 3
310 FOR J=I TO 3
320 FOR K=1 TO 4
330 B(I,J)=B(I,J)+A(K,I)*A(K,J)
340 NEXT K
350 B(J,I)=B(I,J)
360 NEXT J
370 FOR K=1 TO 4
380 B(I,4)=B(I,4)+A(K,I)*A(K,5)
390 B(I,5)=B(I,5)+A(K,I)*(A(K,5)-A(K,4))
400 NEXT K
410 NEXT I
420 PRINT "B="
430 MAT PRINT B;
440 PRINT
450 FOR I=1 TO 2
460 FOR J=I TO 2
470 FOR K=1 TO 3
480 C(I,J)=C(I,J)+B(K,I)*B(K,J)
490 NEXT K
500 C(K,I)=C(I,J)
510 NEXT J
520 FOR K=1 TO 3
530 C(I,3)=C(I,3)+B(K,I)*B(K,4)
540 C(I,4)=C(I,4)+B(K,I)*(B(K,4)-B(K,3))
550 C(I,5)=C(I,5)+B(K,I)*B(K,5)
560 C(I,6)=C(I,6)+B(K,I)*(B(K,5)-B(K,3))
570 NEXT K
580 NEXT I
590 PRINT "C="
600 MAT PRINT C;
610 PRINT
620 FOR K=1 TO 2
630 D(1,1)=D(1,1)+C(K,1)*C(K,1)
640 D(1,2)=D(1,2)+C(K,1)*C(K,3)
650 D(1,3)=D(1,3)+C(K,1)*(C(K,3)-C(K,2))
660 D(1,4)=D(1,4)+C(K,1)*C(K,4)
670 D(1,5)=D(1,5)+C(K,1)*(C(K,4)-C(K,2))
680 D(1,6)=D(1,6)+C(K,1)*C(K,5)
690 D(1,7)=D(1,7)+C(K,1)*(C(K,5)-C(K,2))
700 D(1,8)+D(1,8)+C(K,1)*C(K,6)
710 D(1,9)+D(1,9)+C(K,1)*(C(K,6)-C(K,2))
720 NEXT K
730 PRINT "D="
740 MAT PRINT D;
```

```
750 PRINT
760 FOR I=2 TO 9
770 V(1,I-1)=D(1,I)/D(1,1)
780 NEXT I
790 FOR I=1 TO 7 STEP 2
800 V(2,I)=0
810 V(2,I+1)=1
820 NEXT I
830 PRINT "V="
840 MAT PRINT V;
850 PRINT
860 MAT E=ZER
870 FOR I=1 TO 7 STEP 2
880 FOR J=1 TO 2
890 R1=0
900 R2=0
910 FOR K=1 TO 2
920 R1=R1+C(J,K)*V(K,I)
930 R2=R2+C(J,K)*V(K,I+1)
940 NEXT K
950 R1=C(J,(I+1)/2+2)-R1
960 R2=C(J,(I+1)/2+2)-R2
970 E(J,(I+1)/2)=R2/R1
980 NEXT J
990 NEXT I
1000 PRINT "E3="
1010 MAT PRINT E;
1020 PRINT
1030 FOR I=1 TO 4
1040 FOR J=1 TO 2
1050 U(J,I)=V(J,2*I-1)-(V(J,2*I-1)-V(J,2*I))/(1-E(1,I))
1060 NEXT J
1070 NEXT I
1080 FOR I=1 TO 3 STEP 2
1090 U(3,I)=0
1100 U(3,I+1)=1
1110 NEXT I
1120 PRINT "U="
1130 MAT PRINT U;
1140 PRINT
1150 MAT E=ZER
1160 FOR I=1 TO 3 STEP 2
1170 FOR J=1 TO 3
1180 R1=0
1190 R2=0
1200 FOR K=1 TO 3
1210 R1=R1+B(J,K)*U(K,I)
1220 R2=R2+B(J,K)*U(K,I+1)
1230 NEXT K
```

```
1240 R1=B(J,(I+1)/2+3)-R1
1250 R2=B(J,(I+1)/2+3)-R2
1260 E(J,(I+1)/2)=R2/R1
1270 NEXT J
1280 NEXT I
1290 PRINT "E2="
1300 MAT PRINT E;
1310 PRINT
1320 FOR I=1 TO 2
1330 FOR J=1 TO 3
1340 T(J,I)=U(J,2*I-1)=(U(J,2*I-1)-U(J,2*I))/(1-E(1,I))
1350 NEXT J
1360 NEXT I
1370 T(4,1)=0
1380 T(4,2)=1
1390 PRINT "T="
1400 MAT PRINT T;
1410 PRINT
1420 MAT E=ZER
1430 FOR J=1 TO 4
1440 R1=0
1450 R2=0
1460 FOR K=1 TO 4
1470 R1=R1+A(J,K)*T(K,1)
1480 R2=R2+A(J,K)*T(K,2)
1490 NEXT K
1500 R1=A(J,5)-R1
1510 R2=A(J,5)-R2
1520 E(J,1)=R2/R1
1530 NEXT J
1540 PRINT "E1="
1550 MAT PRINT E;
1560 PRINT
1570 FOR K=1 TO 4
1580 S(K,1)=T(K,1)-(T(K,1)-T(K,2))/(1-E(1,1))
1590 NEXT K
1600 PRINT "THE SOLUTION IS"
1610 MAT PRINT S;
1620 END
```

VIII. ACKNOWLEDGEMENTS

I wish to thank my major professor, Prof. Roy Keller, for his guidance and assistance during the preparation of this paper and during my graduate program.

I would like to thank my wife, Carol, for her work in typing the rough draft, helping me with sentence structure, spelling, etc., and, in general, with putting up with the problems of being a graduate student's wife.

Also, I would like to thank my parents, Walt and Dora Harms, for offering me encouragement and financial assistance all through my twenty-one years of schooling. It was indeed a great burden for them.

In addition, I am grateful to my employer, Hewlett-Packard, for offering me financial assistance, computation time on an HP-9830, and for providing motivation to finish this project.

Finally, I would like to thank the members of my committee — Prof. James Dyer, Prof. Stefan Silverston, Prof. Eugene Steiner, and Prof. Charles Wright.