1970

Acceleration of the projection method for solving systems of linear equations

I-Ming Shen
Iowa State University

Follow this and additional works at: http://lib.dr.iastate.edu/rtd

Part of the Computer Sciences Commons

Recommended Citation
SHEN, I-Ming, 1928-
ACCELERATION OF THE PROJECTION METHOD
FOR SOLVING SYSTEMS OF LINEAR EQUATIONS.

The Iowa State University, Ph.D., 1970
Computer Science

University Microfilms, Inc., Ann Arbor, Michigan
ACCELERATION OF THE PROJECTION METHOD FOR SOLVING SYSTEMS OF LINEAR EQUATIONS

by

I-Ming Shen

A Dissertation Submitted to the Graduate Faculty in Partial Fulfillment of The Requirements for the Degree of DOCTOR OF PHILOSOPHY

Major Subject: Computer Science

Approved:

Signature was redacted for privacy.

In Charge of Major Work

Signature was redacted for privacy.

Head of Major Department

Signature was redacted for privacy.

Dean of Graduate College

Iowa State University Of Science and Technology Ames, Iowa

1970
TABLE OF CONTENTS

I. INTRODUCTION ........................... 1

II. REVIEW OF BASIC PROJECTION METHOD ...... 5

III. BASIC ACCELERATION PROCESS ............... 9
   A. Algorithm Formulation ................ 9
   B. Convergence of Method ................. 18
   C. Rate of Convergence .................. 23
   D. Improvement of Numerical Accuracy .... 27

IV. SECOND ACCELERATION PROCESS ............. 31

V. COMPARISON OF METHOD ...................... 36
   A. Measure of Computational Work ........ 36
   B. Theoretical Comparison ............... 38
   C. Numerical Comparison ................. 40

VI. COMPUTER PROGRAM IMPLEMENTATION ....... 45

VII. SUMMARY ................................... 48

VIII. LITERATURE CITED ....................... 50a

IX. ACKNOWLEDGMENT .......................... 51

X. APPENDIX .................................... 52
I. INTRODUCTION

Methods for solving systems of linear equations are generally classified into two categories; namely, the direct methods and the iterative (indirect) methods. Direct methods are characterized by the fact that a solution is obtained by performing a fixed number of arithmetic operations. The solution is exact if all operations are carried out exactly. Methods of this class are numerous. The Gauss elimination method (6) is one of the most widely used. Iterative methods, on the other hand, obtain a solution by successive approximations. A sequence of operations is performed repeatedly, and the approximate solution is improved steadily toward the exact solution. The list of iterative methods is also long.¹ There are linear iterative methods, relaxation methods, gradient methods, etc. The Jacobi method (9) is a representative one of the linear iterative methods. The Gauss method (5) is probably the earliest relaxation method. The projection method (4) belongs to the class of gradient methods.

Despite extensive developments and studies, the popularity of iterative methods has never reached the point as predicted by Gauss in a letter to his pupil Gerling (5, p. 256): "You will hardly ever eliminate directly, at least not when you

¹A large bibliography covering both direct and iterative methods is given by Forsythe (2).
have more than two unknowns". There are many reasons which make iterative methods less attractive than direct methods. Among them, the most important ones are:

1. Unlike the direct methods which can always obtain a solution (if all computations are carried out to the required accuracy), an iterative method may never converge to a solution unless the system of equations to be solved satisfies certain conditions.

2. A theoretically sound iterative method may not be of any practical value for certain systems because the rate of convergence is too slow.

3. An iterative method which solves a system of few equations efficiently by hand or on a desk calculator with the human insight guidance cannot always be implemented efficiently into computer programs for solving large systems. A typical example is the relaxation method advocated by Southwell (11).

Nevertheless, people are more anxious than ever to find better iterative schemes for many reasons. First, a direct method sometimes cannot obtain an accurate or even useful result due to accumulated rounding errors. In such case, an iterative method may be used either to solve the problem from the beginning or to revise the approximate solution obtained by the direct method. Secondly, the size of the systems obtained from many practical applications is too large to be solved efficiently by direct methods even with the help of today's high
speed, large storage computers. Iterative methods are favored specially for those systems which possess certain properties such as that the coefficient matrix $A$ of the system $Ax = b$ is diagonally dominant and/or sparse. Lastly, many existing iterative methods impose restrictions on the systems to be solved. For instance, all the linear iterative methods require the spectral radius of the iteration matrix to be less than unity. Methods which do not impose restrictions such as the class of gradient methods are generally slow in convergence. The subject of study of this thesis is selected under the motivation of these reasons.

The proposed acceleration process of the so-called projection method has the following desirable properties:

1. The algorithm of the acceleration process is as simple as the original projection method. It does not involve, for example, the difficult job of determining an optimum acceleration parameter.

2. It preserves the nice feature of the original method that no restrictions are imposed on the systems to be solved.

3. For a certain class of problems, it is possible to apply a second acceleration process in addition to the regular acceleration process to speed up the rate of convergence. The approximate solution obtained by the regular acceleration process can be revised to the true solution (within reasonable accuracy) by a fixed number of operations.
The results of some test problems show that the rate of convergence of the acceleration process is much faster than that of the projection method. Sample problems which are very well-conditioned for both the Gauss-Seidel method and the Jacobi method have also been tested. The results show that the acceleration process converges even faster than the two methods just mentioned while the rate of convergence of the projection method falls far behind.

A FORTRAN subroutine is implemented on an IBM 360 computer. The description of the program is given in Chapter VI. The listing of the program and a sample calling program is given in the Appendix.
II. REVIEW OF BASIC PROJECTION METHOD

Among the many methods for solving system of linear equations are the gradient methods. For a given system of n linear equations of the form

\[ Ax = b, \] (II.1)

a quadratic form \((r^k, r^k)\), the inner product of \(r^k\) and \(r^k\), is minimized. Where \(r^k = b - Ax^k\) (the residual vector obtained at the end of the \(k^{th}\) iteration step), and \(x^k\) is the \(k^{th}\) approximate solution vector. A gradient process is described by the non-steady scheme

\[ x^{k+1} = x^k + d^k w^k \] (II.2)

where \(d^k\) is a scalar and \(w^k\) is a vector. For a given \(w^k\), a \(d^k\) is chosen so that \((r^k, r^k)\) is reduced in the change from \(x^k\) to \(x^{k+1}\). The optimum \(d^k\) which makes the most substantial reduction of \((r^k, r^k)\) is\(^1\)

\[ d^k = (u^k, r^k)/(u^k, u^k) \] (II.3)

where \(u^k = Aw^k\).

A. de la Garza (4) described a method which essentially chooses \(w^k\) to be a vector composed of elements of zeroes and ones. Thus, \(u^k = Aw^k\) is the summation of the column vectors of

\(^1\)See Fox (3, p. 205) for the derivation of \(d^k\).
A selected by the non-zero elements of \( w^k \). The iteration process is carried out in a manner such that within a finite number of iterations, every column vector of \( A \) is selected at least once to form the vector \( u^k \).

The simplest choice of \( w^k \) is, of course, a coordinate vector \( e_i \) (the \( i \)th column of the identity matrix). If the coordinate vector is selected in a cyclic order, it results in what we called the projection method. In each iteration step, only the \( i \)th element, \( x_i^k \), of the solution vector \( x \) is modified. The algorithm of the projection method is summarized as follows:

1. Compute the change of the \( i \)th component of the solution vector, \( \Delta x_i^k \),

\[
\Delta x_i^k = d_i^k = (A_i^k, r^k)/(A_i^k, A_i),
\]

where \( A_i \) is the \( i \)th column vector of \( A \).

2. Compute the \( i \)th element of the refined solution vector, \( x_i^{k+1} \),

\[
x_i^{k+1} = x_i^k + \Delta x_i^k.
\]

3. Compute the change of the residual vector, \( \Delta r^k \),

\[
\Delta r^k = r^k - r^{k+1}
= r^k - (b - Ax_i^{k+1})
= r^k - (b - Ax_i^k - A\Delta x_i^k)
\]
4. Compute the resulting residual vector, \( r^{k+1} \),

\[
r^{k+1} = r^k - \Delta r^k.
\]

The above sequence of computations constitutes one iteration step of the projection method. It is repeated by taking \( k = 0, 1, 2, \ldots \), and \( i = 1, 2, \ldots, n, 1, 2, \ldots, n, \ldots \). The process is terminated when the desired accuracy of the solution vector is reached. The initial residual vector \( r^0 \) is computed from \( r^0 = b - Ax^0 \) with an arbitrary choice of \( x^0 \). The term "cycle" refers to each \( n \) successive iteration steps.

The name "projection method" comes from the fact that the change of \( x^k_i \) is the orthogonal projection of the residual vector \( r^k \) onto \( A_i \). A detailed geometrical interpretation of this method is given by Householder (7).

The convergence of the projection method is assured as long as the system possesses a unique solution, i.e. \( A \) is nonsingular. Proof of this statement may be found in (4) and (8).

The projection method, though simple in formulation, is generally slow in convergence. One technique to improve the rate of convergence as suggested by Garza (4) is to choose \( e_i \) in some non-cyclic order such that the current change of \( (r^k, r^k) \) would be maximized. This, of course, involves considerable
work and can hardly be considered of any practical value except for solving small systems.
III. BASIC ACCELERATION PROCESS

A. Algorithm Formulation

As described in Chapter II, the projection method modifies one component of the approximate solution vector $x$ in each iteration step and the process is performed in a cyclic order. If, instead of carrying out the process in this manner, we do the following:

Step 1: Correct $x^k_2$ and $x^k_1$ alternatively by the regular projection method until the quantity $(r^k_1, r^k_2)$ is minimized to $(r^{k+1}_1, r^{k+1}_2)$ and no further reduction can be made.

Step 2: Same as step 1 but the iterations are performed on $x^{k+1}_3$ and $x^{k+1}_2$.

\vdots

Step n: Same as the other steps except that the iterations are performed on $x^{k+n-1}_1$ and $x^{k+n-1}_n$.

Step 1 through step n form an iteration cycle and are repeated until the desired solution is reached. Each step of this algorithm contains many steps of the projection method [if one requires that $(r^k_1, r^k_2)$ be reduced to absolute minimum, the number of steps required is actually infinite]. This algorithm, at the first glance, does not seem to offer any advantages over the original method. However, a further study shows that in each step, we need only to compute one change for each of the two components involved. The total change of each component is
the summation of an infinite series which can be easily evaluated by a closed formula. This makes the algorithm, henceforth will be called the acceleration process, in most cases, far superior over the projection method. In order to illustrate this point, we prove first two lemmas.

**Lemma III.1** The residual vector \( r^{k+1} \) obtained from Equation II.7 by the projection method is orthogonal to \( A_i \).

**Proof:** From Equations II.6 and II.7,

\[
r^{k+1} = r^k - A_i \Delta x^k_i.
\]

Substitute Equation II.4 into the above equation,

\[
r^{k+1} = r^k - (A_i, r^k) A_i / (A_i, A_i).
\]

Take the inner product of both side by \( A_i \), we get

\[
(r^{k+1}, A_i) = (r^k, A_i) - (A_i, r^k)(A_i, A_i) / (A_i, A_i) = 0.
\]

Q.E.D.

**Lemma III.2** The residual vector \( r^{k+1} \) obtained at the end of the \( k \)th step by the acceleration process is orthogonal to both vectors \( A_i+1 \) and \( A_i \) where \( i = k \) modulo \( n \).

**Proof:** Since \( r^{k+1} \) cannot be reduced further by iterating on \( x^k_{i+1} \) and \( x^k_i \), from Equation II.4,

\[
\Delta x^k_i = (A_i, r^{k+1}) / (A_i, A_i) = 0,
\]

and

\[
\Delta x^k_{i+1} = (A_{i+1}, r^{k+1}) / (A_{i+1}, A_{i+1}) = 0.
\]
It is clear that $r^{k+1}$ is orthogonal to $A_{i+1}$ and $A_i$. Q.E.D.

Next, we introduce the following notations:

- $\Delta_j x_i^k$ - The $j^{th}$ correction to the $i^{th}$ component of the solution vector in the $k^{th}$ step of the acceleration process.
- $\Delta_j r_i^k$ - The change of the residual vector corresponding to the correction $\Delta_j x_i^k$.

The total change of the elements $x_i^k$ and $x_{i+1}^k$ is

$$\Delta x_i^k = \Delta_1 x_i^k + \Delta_2 x_i^k + \ldots \quad (III.1)$$

and

$$\Delta x_{i+1}^k = \Delta_1 x_{i+1}^k + \Delta_2 x_{i+1}^k + \ldots \quad (III.2)$$

Now, we are going to show that to calculate the above two series summations is a rather trivial matter.

At the beginning of the $k^{th}$ step of the acceleration process, we have the solution vector $x^k$ and the residual vector $r^k$. From Equations II.4, II.6, and II.7, we get the following results of the first few iterations within the $k^{th}$ step.

$$\Delta_1 x_{i+1}^k = (r^k, A_{i+1})/(A_i, A_{i+1}) \quad (III.3)$$

$$\Delta_1 r_{i+1}^k = \Delta_1 x_i^k A_{i+1} \quad (III.4)$$
\[ \Delta_1 x^k_i = [(r^k - \Delta_1 r^k_{i+1}), A_i]/(A_i, A_i) \]  
\[ = [(r^k, A_i) - (\Delta_1 r^k_{i+1}, A_i)]/(A_i, A_i). \] 

From lemma III.2, \( r^k \) is orthogonal to \( A_i \), hence, \( (r^k, A_i) = 0 \), and Equation III.5 is reduced to

\[ \Delta_1 x^k_i = -\Delta_1 r^k_{i+1}/(A_i, A_i). \]  

(III.6)

Substitute \( \Delta_1 r^k_{i+1} \) by III.4,

\[ \Delta_1 x^k_i = -\Delta_1 x^k_{i+1}(A_{i+1}, A_i)/(A_i, A_i). \]  

(III.7)

The corresponding change of the residual vector is

\[ \Delta_1 r^k_i = \Delta_1 x^k_i A_i. \]  

(III.8)

The resulting residual vector at this point is \( r^k - \Delta_1 r^k_{i+1} - \Delta_1 r^k_i \). The next iteration gives

\[ \Delta_2 x^k_{i+1} = [(r^k - \Delta_1 r^k_{i+1} - \Delta_1 r^k_i), A_{i+1}]/(A_{i+1}, A_{i+1}). \]

Since \( r^k - \Delta_1 r^k_{i+1} \) is orthogonal to \( A_{i+1} \) from lemma III.1, the above equation is simplified to

\[ \Delta_2 x^k_{i+1} = -\Delta r^k_{i+1}/(A_{i+1}, A_{i+1}). \]

Substitute III.8 for \( \Delta_1 r^k_i \), we get

---

1 This orthogonality condition does not hold for \( k = 0 \), because \( r^0 \) is obtained from \( x^0 \) which is arbitrarily chosen.
The change of residual vector is

\[ \Delta x_{i+1}^k = \Delta x_i^k A_{i+1}. \]  

(III.10)

The next iteration yields

\[ \Delta x_i^k = \left( r^k - \Delta r_{i+1}^k - \Delta r_i^k - \Delta x_{i+1}^k \right) (A_i, A_i) / (A_i, A_i) \]

(III.11)

\[ \Delta x_i^k = -\Delta x_{i+1}^k (A_{i+1}, A_i) / (A_i, A_i), \]

and

\[ \Delta x_i^k = \Delta x_{i+1}^k A_i. \]  

(III.12)

Substitute III.7 into III.9,

\[ \Delta x_{i+1}^k = \Delta x_i^k \left( (A_{i+1}, A_i) \right)^2 \left( (A_i, A_i) \right) \]

If \( \Delta x_{i+1}^k \) is different from zero, we may write

\[ \Delta x_{i+1}^k / \Delta x_{i+1}^k = (A_{i+1}, A_i)^2 / [(A_i, A_i)^2 (A_i, A_i)] \]

(III.13)

From Cauchy-Schwarz Inequality Theorem\(^1\),

\[ 0 \leq f_i < 1 \]  

(III.14)

\[^1\text{See, for example, Mostow and Sampson (10, p. 22).}\]
unless $A_{i+1}$ and $A_i$ are linearly dependent. In that case, $f_i = 1$. Similarly, we obtain the following equation by substituting III.9 into III.11,

$$\Delta_2 x_i^k / \Delta_1 x_i^k = f_i.$$  \hfill (III.15)

Here again we assume that $\Delta_1 x_i^k$ is non-zero. One can easily verify that the following equations are true.

$$\Delta_{j+1} x_i^{k+1} / \Delta_j x_i^k = f_i \text{ for } j = 1, 2, \ldots, \hfill (III.16)$$

and

$$\Delta_{j+1} x_i^k / \Delta_j x_i^k = f_i \text{ for } j = 1, 2, \ldots \hfill (III.17)$$

The only exception to Equation III.16 is when $k = 0, j = 1$. In that case, the residual vector $r^0$ computed from $r^0 = b - A x^0$ is in general not orthogonal to $A_1$. Equation III.6 does not apply and Equation III.5 must be used to compute $\Delta_1 x_1^1$. This invalidates Equation III.7 which, in turn, invalidates Equation III.13. But Equations III.8 through III.12 are not affected. Therefore, we have the following inequality

$$\Delta_2 x_2^0 / \Delta_1 x_2^0 \neq f_1.$$  \hfill (III.18)

Beyond this point, the orthogonality condition is established and equation similar to III.6 can be used to compute the subsequent changes of $x_i^k$.

From Equation III.1, the total change of the $i^{th}$ component of the solution vector in the $k^{th}$ step is
\[ \Delta x_i^k = \Delta_1 x_i^k + \Delta_2 x_i^k + \Delta_3 x_i^k + \ldots \]
\[ = \Delta_1 x_i^k (1 + \Delta_2 x_i^k/\Delta_1 x_i^k + \Delta_3 x_i^k/\Delta_1 x_i^k + \ldots) \]
\[ = \Delta_1 x_i^k [1 + \Delta_2 x_i^k/\Delta_1 x_i^k + (\Delta_3 x_i^k/\Delta_2 x_i^k)(\Delta_2 x_i^k/\Delta_1 x_i^k)] \]
\[ + \ldots]. \]

From Equation III.17, we get
\[ \Delta x_i^k = \Delta_1 x_i^k [1 + f_i + (f_i)^2 + \ldots] \]
\[ = \Delta_1 x_i^k \lim_{m \to \infty} [1 - (f_i)^m]/(1 - f_i). \]

Since \(0 < f_i < 1\) from Equation III.14, \(\lim_{m \to \infty} [1 - (f_i)^m] = 1\), hence
\[ \Delta x_i^k = \Delta_1 x_i^k/(1 - f_i). \] (III.19)

Similarly,
\[ \Delta x_{i+1}^k = \Delta_1 x_{i+1}^k/(1 - f_i). \] (III.20)

For \(k = 0, i = 1\), the ratio \(\Delta x_2^0/\Delta_1 x_2^0\) does not equal to \(f_1\) as indicated by Equation III.18. We may write Equation III.2 in the following form:
\[ \Delta x_2^0 = \Delta_1 x_2^0 + \Delta_2 x_2^0 (1 + \Delta_3 x_2^0/\Delta_2 x_2^0 + \Delta_4 x_2^0/\Delta_3 x_2^0 + \ldots) \]
\[ = \Delta_1 x_2^0 + \Delta_2 x_2^0 [1 + f_1 + (f_1)^2 + \ldots] \] (III.21)
\[ = \Delta_1 x_2^0 + \Delta_2 x_2^0/(1 - f_1). \]
We have assumed that $\Delta^1_{i+1} x^k, \Delta^1 x^k,$ and $\Delta^2 x^0$ are non-zero as we derived Equations III.13, III.15, and III.21, respectively. However, it can be checked easily that if $\Delta^1 x^k = 0,$ all subsequent changes of $x^k_i$ would be zero. The same is true for the other two cases. Therefore, Equations III.19, III.20, and III.21 are always valid.

We have illustrated how the two series summations III.1 and III.2 can be evaluated with ease. The savings in computational work is considerable. We will discuss this point further in Chapter V.

The algorithm of the acceleration process is summarized below.

1. Compute the total change $\Delta x^k_{i+1}.$ From Equations III.3 and III.20,
   \[
   \Delta x^k_{i+1} = \frac{(r^k, A^i_{i+1})}{[(A^i_{i+1}, A^i_{i+1})(1 - f^i)]}. \tag{III.22}
   \]

2. Compute the total change $\Delta x^k_i.$ From Equations III.7, III.19, and III.20,
   \[
   \Delta x^k_i = \frac{\Delta x^k_{i+1}}{(1 - f^i)} \tag{III.23}
   \]
   \[
   = - \frac{\Delta x^k_{i+1}(A^i_{i+1}, A^i_{i})}{[(A^i_{i}, A^i_{i})(1 - f^i)]}
   \]
   \[
   = - \Delta x^k_i (A^i_{i+1}, A^i_{i})/(A^i_{i}, A^i_{i}).
   \]

3. Compute the residual vector $r^{k+1}.$
   \[
   r^{k+1} = r^k - \Delta x^k_i A^i_{i} - \Delta x^k_{i+1} A^i_{i+1}. \tag{III.24}
   \]
4. Compute the solution vector $x^{k+1}$

\[ x_{i}^{k+1} = x_{i}^{k} + \Delta x_{i}^{k}. \]  

(III.25)

\[ x_{i+1}^{k+1} = x_{i+1}^{k} + \Delta x_{i+1}^{k}. \]  

(III.26)

\[ x_{j}^{k+1} = x_{j}^{k} \text{ for } j \neq i \text{ and } j \neq i+1. \]  

(III.27)

The above calculations form one step of the acceleration process. We call every $n$ steps an iteration cycle. The process is carried out in a cyclic order with $k = 0, 1, 2, \ldots$, and $i = k+1$ modulo $n$.

For $k = 0$, $i = 1$, the algorithm should be modified as follows.

1. Compute the change $\Delta_{1}x_{2}^{0}$ from Equation III.3.

2. Compute the change $\Delta_{1}r_{2}^{0}$ from Equation III.4.

3. Compute the change $\Delta_{1}x_{1}^{0}$ from Equation III.5.

4. Compute the total change $\Delta x_{1}^{0}$ from Equation III.19.

5. Compute the total change $\Delta x_{2}^{0}$. From Equations III.9 and III.21,

\[ \Delta x_{2}^{0} = \Delta_{1}x_{2}^{0} - \Delta_{1}x_{1}^{0}(A_{1}, A_{2})/[(A_{2}, A_{2})(1 - \alpha_{1})] \]  

(III.28)

\[ = \Delta_{1}x_{2}^{0} - \Delta x_{1}^{0}(A_{1}, A_{2})/(A_{2}, A_{2}). \]

6. Compute the residual vector $r_{1}^{1}$ from Equation III.24.

7. Compute the solution vector $x_{1}^{1}$ from Equations III.25, III.26 and III.27.
B. Convergence of Method

Theorem: The acceleration process of the projection method converges for all systems which possess a unique solution.

Proof: Suppose that the residual vector \( r^k \) is not a null vector. The residual vector at the end of the next step is, according to Equation III.24,

\[
r^{k+1} = r^k - \Delta x_i^k A_i - \Delta x_{i+1}^k A_{i+1}.
\]

Substitute \( \Delta x_i^k \) by Equation III.23,

\[
r^{k+1} = r^k + \Delta x_{i+1}^k [(A_{i+1} A_i) A_i / (A_i A_i) - A_{i+1}].
\]

The inner product of the above equation by itself is

\[
(r^{k+1}, r^{k+1}) = (r^k, r^k) + 2 \Delta x_{i+1}^k [(A_{i+1} A_i) (r^k, A_i) / (A_i A_i)]
\]

\[
- (r^k, A_{i+1}) + (\Delta x_{i+1}^k)^2 [(A_{i+1} A_{i+1}) - (A_i A_i)]
\]

Since \((r^k, A_i) = 0\) by lemma III.2, the above equation, after expressing \((r^k, A_{i+1})\) and \(\Delta x_{i+1}^k\) in terms of \(\Delta x_i^k\), is reduced to

\[
(r^{k+1}, r^{k+1}) = (r^k, r^k) - (\Delta x_i^k)^2 (A_i A_i)^2 (1 - f_i)
\]

\[
/ (A_{i+1} A_{i+1}).
\]

If \(\Delta x_i^k\) is expressed in terms of \(\Delta x_{i+1}^k\), we have

\[
(r^{k+1}, r^{k+1}) = (r^k, r^k) - (\Delta x_{i+1}^k)^2 (A_{i+1} A_{i+1}) (1 - f_i)
\]

Define two vectors \(c^k\) and \(d^k\) such that \(c^k\) has only one non-zero element \(c_i^k = \Delta x_i^k\) and \(d^k\) has only one non-zero element
\[ d_{i+1}^k = \Delta x_{i+1}^k. \] The above two equations may then be written as

\[ (r^{k+1}, r^{k+1}) = (r^k, r^k) - (Ac^k, Ac^k)g_k, \]  

(III.29a)

\[ (r^{k+1}, r^{k+1}) = (r^k, r^k) - (Ad^k, Ad^k)h_k, \]  

(III.29b)

where \( g_k = (A_i, A_i)(1 - f_i)/(A_{i+1}, A_{i+1}), \) \( h_k = 1 - f_i. \) Since \( g_k > 0, \) and \( h_k > 0, \) the above equations indicate that \( (r^{k+1}, r^{k+1}) \leq (r^k, r^k). \)

This result indicates that the length of the residual vector \( r^k \) is either reduced or not changed. The later case happens when \((r^k, A_{i+1}) = 0, \) i.e. \( r^k \) is orthogonal to \( A_{i+1}. \) Carrying out the next step, the length of \( r^{k+1} \) is again reduced or unchanged. If it remains unchanged, then \( r^{k+1} \) is orthogonal to \( A_{i+2}. \) Suppose \( r^k \) remains unchanged after \( n \) iterations, then \( r^k(= r^{k+1} = r^{k+2} = \ldots = r^{k+n-1}) \) is orthogonal to all the column vectors of \( A. \) Since \( A \) is non-singular, the \( n \) column vectors of \( A \) form a complete set of linearly independent vectors. The only vector which is orthogonal to such a set is the null vector. This contradicts the hypothesis that \( r^k \) is a non-zero vector. Thus, the length of the residual vector must be reduced by a certain amount in \( n \) steps (one cycle). This leads to the conclusion that the sequence \( \{(r^k, r^k)\} \) has a limit, because it is monotonic decreasing in a wider sense and is bounded below by zero. From Equations III.29a and III.29b, we see that
\[ \lim_{k \to \infty} c^k = \emptyset, \quad \lim_{k \to \infty} d^k = \emptyset. \]

From lemma III.2, \((A^i_{i+1}, r^k) = 0\) where \(i = k\) modulo \(n\). In order to simplify the notations, we choose \(k\) to be such that 
\((A^i_i, r^{k+i}) = 0\), for \(i = 1, 2, \ldots, n\). We may then write the following equations 
\[ A^T_i r^k = A^T_i (r^k - r^{k+i}), \text{ for } i = 1, 2, \ldots, n, \]
or,
\[ A^T r^k = (b - Ax^k) = u, \quad (III.30) \]

where the \(i^{th}\) element of the vector \(u\) is
\[ u_i = A^T_i (r^k - r^{k+i}) = A^T_i \Delta x^k + \Delta x^{k+1} + \ldots + \Delta x^{k+i-1} \]
\[ = A^T_i \sum_{j=k}^{k+i-1} \Delta x^j = A^T_i \sum_{j=k}^{i+i-1} (c^j + d^j) \]

Pre-multiplying Equation III.30 by \((A^T A)^{-1}\) and replacing \(r^k\) by \(b - Ax^k\), we have
\[ x^k = A^{-1} b - (A^T A)^{-1} u. \]

Take the limit of the above equation we get
\[ \lim_{k \to \infty} x^k = \lim_{k \to \infty} [A^{-1} b - (A^T A)^{-1} u] \]
\[ A^{-1}b - A^T A^{-1} \lim_{k \to \infty} u. \]

But \( \lim_{k \to \infty} u = 0 \), because \( \lim_{k \to \infty} \frac{1}{k} (c^j + d^j) = 0 \), for \( i = 1, 2, \ldots, n \). Therefore,

\[ \lim_{k \to \infty} x^k = A^{-1}b = x. \]

Thus complete the proof.

Next, we are going to establish a linear relationship between the residual vectors of successive iteration cycles.

From Equations III.22, III.23, and III.24,

\[ r^{k+1} = r^k + (r^k, A_{i+1}) [(A_{i+1}, A_{i}) A_i / (A_i, A_i) - A_{i+1}] / [(A_{i+1}, A_{i+1})(1 - f_i)]. \]

We define

\[ c_i = 1 / [(A_{i+1}, A_{i+1})(1 - f_i)], \]  \hspace{1cm} \text{(III.31)}

\[ d_i = c_i (A_i, A_{i+1}) / (A_i, A_i), \]  \hspace{1cm} \text{(III.32)}

then,

\[ r^{k+1} = r^k + (r^k, A_{i+1})(d_i A_i - c_i A_{i+1}) \]

\[ = r^k + (d_i A_i - c_i A_{i+1}) A_{i+1}^T r^k. \]

Let \( B_i = (d_i A_i - c_i A_{i+1}) A_{i+1}^T \),

\[ \text{Let } B_i = (d_i A_i - c_i A_{i+1}) A_{i+1}^T. \]  \hspace{1cm} \text{(III.33)}
then,

$$r^{k+1} = (I + B_1)r^k,$$  \hspace{1cm} (III.34)

where $I$ is the identity matrix. It follows from III.34 that after a complete iteration cycle, the resulting residual vector is,

$$r^{k+n} = (I + B_n)(I + B_{n-1}) \ldots (I + B_1)r^k.$$

Again, we use $p$ to indicate the $p^{th}$ cycle and $p+1$ to indicate the next cycle,

$$r^{p+1} = (I + B_n)(I + B_{n-1}) \ldots (I + B_1)r^p = Br^p \hspace{1cm} (III.35)$$

where $B = (I + B_n)(I + B_{n-1}) \ldots (I + B_1)$. Equation III.35 is useful in the discussion of rate of convergence and in the development of the so-called second acceleration process.

If the system does not have a unique solution, then $A$ must be singular. If $f_i = 1$ for at least one $i$, we will recognize immediately that $A$ is singular. Question arises when $f_i \neq 1$ for all $i$ and $A$ is singular. The outcome of applying the acceleration process to such a system is unpredictable. The process may not converge but may also converge to any one of the infinite set of solutions depending on the initial choice of the solution vector $x^0$. Suppose $r^k$ remains unchanged for a complete iteration cycle, then $r^k$ is orthogonal to all the $A$ column vectors as we proved in the preceding paragraph. But,
these A vectors do not form a complete independent set, therefore, the fact that \( r^k \) is not a null vector does not constitute a contradiction. So, the acceleration process cannot converge to a solution in this case. On the other hand, there is no reason why \( r^k \) cannot be reduced to a null vector. For example, it is certainly possible that \( r^k \) is parallel to \( A_{i+1} \), say \( r^k = cA_{i+1} \), then from Equation III.22,

\[
\Delta x^k_{i+1} = (r^k, A_{i+1})/[(A_{i+1}, A_{i+1})(1 - f_i)]
\]

\[
= c(A_{i+1}, A_{i+1})/[(A_{i+1}, A_{i+1})(1 - f_i)]
\]

\[
= c/(1 - f_i).
\]

Since \( r^k \) is orthogonal to \( A_i \), so \( A_{i+1} \) is orthogonal to \( A_i \). From lemma III.2 and Equation III.13, \( (r^k, A_i) = c(A_{i+1}, A_i) = 0 \) and \( f_i = 0 \). Therefore, \( \Delta x^k_i = 0 \) from Equation III.23. The residual vector \( r^{k+1} \) is, from Equation III.24,

\[
r^{k+1} = r^k - cA_{i+1}/(1 - f_i)
\]

\[
= cA_{i+1} - cA_{i+1}
\]

\[
= 0.
\]

So the process converges. Computationally we must be concerned with this case. Because for a nearly singular system (ill-conditioned), the solution obtained may be far away from the true solution even though the process seems to have converged.
C. Rate of Convergence

A general linear iterative method can be expressed in the following form,

$$x^{k+1} = Mx^k + g$$  \hspace{1cm} (III.36)

where $M$ is a matrix of dimension $n$, $g$ is a vector, and $k$ is the iteration cycle index. Ultimately, Equation III.36 has to be satisfied by the true solution $x$, that is

$$x = Mx + g.$$  \hspace{1cm} (III.37)

Subtracting Equation III.36 from Equation III.37, we have

$$x - x^{k+1} = M(x - x^k).$$

We define the vector $E^i = x - x^i$ as the error vector associated with the $i^{th}$ iteration cycle. The above equation may be written as

$$E^{k+1} = ME^k, \quad k \geq 0.$$

From this equation, we get

$$E^m = ME^{m-1} = M^2E^{m-2} = \ldots = M^m E^0, \quad m \geq 0.$$

Taking norms of both side,

$$\| E^m \| = \| M^m E^0 \| \leq \| M^m \| \| E^0 \|, \quad m \geq 0.$$  \hspace{1cm} (III.38)

The above inequality relation does not help to predict how many iterations will be needed to get the solution of a given problem, because the quantity $\| E^0 \|$ is related to the true
solution, hence, cannot be determined in advance. But, it
does give an upper bound of the rate of convergence in terms
of the reduction of $\| E^0 \|$ for $m$ iterations. Therefore, $\| M^m \|
may be used as a basis of comparison of different methods.

The average rate of convergence for $m$ iterations of the
matrix $M$ is defined as

$$R(M^m) = -\ln((\| M^m \|)^{1/m}) = -\ln(\| M^m \|/m). \quad (III.39)$$

If $R(A^m) < R(B^m)$, then $B$ is iteratively faster for $m$ iterations
than $A$. In $m$ iterations, the norm of the error vector is re­
duced from $\| E^0 \|$ to $\| E^m \|$, so the average reduction factor per
iteration for the successive norms is

$$s = (\| E^m \| / \| E^0 \|)^{1/m}. \quad (III.38)$$

From Equation (III.38),

$$s \leq (\| M^m \|)^{1/m}. \quad (III.38)$$

From Equation (III.39),

$$ (\| M^m \|)^{1/m} = e^{-R(M^m)},$$

where $e$ is the base of the natural logarithm. It follows that

$$s \leq e^{-R(M^m)},$$

or,

$$s (R(M^m))^{-1} \leq 1/e.$$  

From the definition of $s$, it is clear that $1/R(M^m)$ is the num-
ber of iterations required to reduce the quantity \( \| E^0 \| \) by a factor of \( e \).

The following theorem\(^1\) relates the average rate of convergence of a matrix \( M \) to its spectral radius \( \rho(M) \).

**Theorem III.1** For all \( m \) sufficiently large, the average rate of convergence for \( m \) iterations \( R(M^m) \) satisfies

\[
\lim_{m \to \infty} R(M^m) = -\ln \rho(M) \geq R_{\infty}(M). 
\] (III.40)

\( R_{\infty}(M) \) is called the asymptotic rate of convergence. Since \( \| M^m \| \geq [\rho(M)]^m \) for all \( m \geq 1 \), it follows that

\[
R_{\infty}(M) \geq R(M^m) 
\] (III.41)

for any positive integer \( m \) for which \( \| M^m \| < 1 \). \( R_{\infty}(M) \) is usually used as a measure of the rate of convergence.

Our next job is to establish the matrix \( M \) of the acceleration process. From Equation III.35 and the definition of residual vector, we have

\[
b - Ax^{p+1} = B(b - Ax^p). \] (III.42)

This equation is ultimately satisfied by the true solution vector \( x \), so

\[
b - Ax = B(b - Ax). \]

---

\(^1\)For the proof of this theorem and further discussions on the average rate of convergence, see Varga (12, pp. 61-68).
Taking the difference of the last two equations, we have

$$A(x - x^{P+1}) = B A(x - x^P),$$
or,

$$A E^{P+1} = B A E^P,$$
or,

$$E^{P+1} = A^{-1} B A E^P.$$  \hfill (III.43)

The matrix $A^{-1} B A$ is then equivalent to the matrix $M$ defined in III.36. It is, of course, not possible to establish this matrix without knowing $A^{-1}$. We may, however, measure the rate of convergence with respect to $\| r^1 \|$. This is done as follows.

From Equation III.35,

$$r^{P+1} = B r^P = B^2 r^{P-1} = \ldots = B^P r^1,$$ \hfill (III.44)

$$\| r^{P+1} \| \leq \| B^P \| \| r^1 \|.$$  

So, $R(B^P)^{-1}$ is the number of iteration cycles required to reduce $\| r^1 \|$ by a factor of $e$. Since $r^1$ is a known vector, we may actually find the upper bound of the number of iteration cycles required to reduce $\| r^1 \|$ to a desired minimum$^1$.

We might point out here that what we have discussed in this section is merely for the interest of theoretical studies. In practice, it is much easier to perform a few iterations to see how the convergence behaves rather than to construct the matrix $B$.

\[1\text{Notice that } r^1 \neq B r^0 \text{ because Equation III.22 does not apply for } k = 0, i = 1.\]
matrix and to find its spectral radius. The same is true for the matrix \( M \) which may not be difficult to construct, but, its spectral radius, in general, cannot be easily determined.

D. Improvement of Numerical Accuracy

In the previous sections, we proved the theoretical aspect of the convergence of the acceleration process. Now, let us turn our attention to the problem of accuracy. Like any other method, iterative or direct, the acceleration process also faces the difficulties caused by numerical inaccuracy. How to overcome such difficulties is a subject of interest by itself, and is beyond the scope of this study. However, some numerical problems which are peculiar to the acceleration process deserve a discussion.

As discussed in section A, the acceleration process speeds up the convergence by replacing the summation of an infinite series by a closed formula. From Equations III.19 and III.20, one can see that any error existed in \( \Delta x_i^k \) and \( \Delta x_{i+1}^k \) will be magnified by the factor \( 1/(1 - f_i) \). If \( f_i \) is very close to unity, i.e. \( A_i \) and \( A_{i+1} \) are nearly parallel to each other, the accuracy of \( \Delta x_i^k \) and \( \Delta x_{i+1}^k \) would be seriously affected.

One way to eliminate this error is, of course, to compute \( \Delta x_i^k \) and \( \Delta x_{i+1}^k \) as accurately as possible. Beside this, we can rearrange the column vectors of \( A \) so that the angle between two neighboring column vectors would not be too close to zero or \( \pi \).
This is equivalent to saying that no \( f_i \) is close to unity, because \( f_i \) is actually the square of the cosine of the angle between \( A_i \) and \( A_{i+1} \). This technique is somewhat similar to the pivotal search scheme applied to the Gauss elimination method. One should realize that this technique is not foolproof. It behaves similar to the pivotal search which sometimes merely delays rather than eliminates the appearance of errors. It is specially so when \( A \) is nearly singular. As we pointed out in section B, the acceleration process may converge to a solution even when the system is singular. A nearly singular system is no different from a singular system from a numerical point of view. The correctness of the solution obtained for such a system is highly unreliable. Therefore, this technique should not be used as a way to solve ill-conditioned systems but rather as a way to improve the rate of convergence for a not-so-ill-conditioned problem. This point will be clarified later in this section.

One advantage of an iterative method over a direct method is that the rounding errors of an iterative method are not accumulative. The inaccuracy of the approximate solution vector obtained in one step has no effect on the final solution, because the intermediate solution vector is used merely as a guessed value for the next step. Other than this, one step is completely independent from the other steps. It is not quite so in the acceleration process. In section A, we have shown that Equation III.6 is derived based on the condition that the
residual vector, \( r^k \), obtained in the preceding step is orthogonal to the column vector \( A_i \). Since \( r^k \) is directly related to \( \Delta x_i \) and \( \Delta x_{i-1} \), the error in \( \Delta x_i \) and \( \Delta x_{i-1} \) will cause the residual vector \( r^k \) from being orthogonal to \( A_i \). If this is the case, the equations derived in section A are then no longer valid. The error may propagate and ruin the convergence of the process.

The easiest way to prevent the error from propagating is to treat each step identically as what is done for the first step of the first cycle. In the first step of the first cycle, one change of \( x_1 \) and two changes of \( x_2 \) are computed due to the fact that the initial residual vector \( r^0 \) is not necessarily orthogonal to \( A_1 \).

The modified algorithm is as follows.

\[
\Delta x^k_{i+1} = \frac{(r^k, A_{i+1})}{(A_{i+1}, A_{i+1})}
\]

\[
\Delta r^k_{i+1} = \Delta x^k_{i+1} A_{i+1}
\]

\[
\Delta x^k_i = \frac{(r^k - \Delta r^k_{i+1}, A_i)}{[(A_i, A_i)(1 - f_i)]}
\]

\[
\Delta x^k_{i+1} = \Delta x^k_i - \Delta x^k_{i+1}(A_i, A_{i+1})/(A_{i+1}, A_{i+1})
\]

\[
r^{k+1} = r^k - \Delta x^k_i A_i - \Delta x^k_{i+1} A_{i+1}
\]

\[
x^{k+1}_i = x^{k+1}_i + \Delta x^k_{i+1}
\]

\[
x^{k+1}_i = x^k_i + \Delta x^k_i
\]
\[ x_{j}^{k+1} = x_{j}^{k} \text{ for } j \neq i \text{ and } j \neq i+1. \]

The above calculations are again carried out in a cyclic order with \( k = 0, 1, 2, \ldots \), and \( i = k+1 \) modulo \( n \).

This algorithm makes each step completely independent from other steps. Thus, the errors in the results of one step will not affect the solution of the next step. However, if the value of \( f_{i} \) is close to unity, the errors introduced in \( x_{i}^{k} \) and \( x_{i+1}^{k} \) may be very large. This is equivalent to making a bad guess vector for the next step and, hence, slows down the convergence. The selection of a better set of \( (f_{i}) \) is then helpful to avoid this undesirable phenomenon.
IV. SECOND ACCELERATION PROCESS

In this chapter we are going to show that for certain systems, a second acceleration process\(^1\) may be applied to speed up the rate of convergence of the basic acceleration process described in Chapter III.

From Equation III.44,

\[ r^{p+1} = B^p \, r^1 \]  

(IV.1)

where the index \( p+1 \) refers to the \((p+1)\)th iteration cycle (to be distinguished from the step number). We also have, from Equation III.43,

\[ E^{p+1} = x - x^{p+1} = A^{-1}BA^p = A^{-1}B^2A^p - 1 = \ldots = A^{-1}B^pA^1. \]

Since \( E^1 = x - x^1 = x - A^{-1}(b - r^1) = A^{-1}r^1 \), therefore,

\[ E^{p+1} = A^{-1}B^p r^1. \]  

(IV.2)

From Equation III.42,

\[ x^{p+2} = A^{-1}[b - B(b - Ax^{p+1})] \]

\[ = A^{-1}(I - B)b + A^{-1}BAx^{p+1}, \]

and

\[ x^{p+1} = A^{-1}(I - B)b + A^{-1}BAx^p. \]

\(^1\)The idea of this second acceleration process is nothing new. It is originated from the Aitken's acceleration process (1). We merely establish here the theoretical justification for applying it to the accelerated projection method.
Taking the difference of these two equations, we get

\[ \Delta x^{p+1} = x^{p+2} - x^{p+1} = A^{-1}BA(x^{p+1} - x^p) = A^{-1}BA\Delta x^p \]

\[ = A^{-1}B^2A\Delta x^{p-1} = \ldots = A^{-1}BPA\Delta x^1 \]

\[ = A^{-1}BPA(x^2 - x^1) = A^{-1}BPA\left[A^{-1}\left[(b - r^2) - (b - r^1)\right]\right] \]

\[ = A^{-1}B^p(r^1 - r^2). \]

From Equation IV.1, \( r^2 = Br^1 \), so,

\[ \Delta x^{p+1} = A^{-1}B^p(I - B)r^1. \quad \text{(IV.3)} \]

Suppose the matrix \( B \) has \( n \) distinct eigenvectors, \( y^i \) for \( i = 1, 2, \ldots, n \), and the corresponding eigenvalues, \( \lambda_i \), are arranged in the order of

\[
\begin{vmatrix}
\lambda_1 \\
\lambda_2 \\
\vdots \\
\lambda_n
\end{vmatrix}.
\]

We may express the vector \( r^1 \) uniquely as a linear combination of this set of eigenvectors. If we let \( r^1 = \sum_{i=1}^{n} c_i y^i \), then,

\[ x^{p+1} = B^p r^1 = B^p \sum_{i=1}^{n} c_i y^i = \sum_{i=1}^{n} c_i (\lambda_i)^p y^i, \quad \text{(IV.4)} \]

\[ \Delta x^{p+1} = A^{-1}B^p(I - B)r^1 = A^{-1} \sum_{i=1}^{n} c_i (\lambda_i)^p (1 - \lambda_i) y^i, \quad \text{(IV.5)} \]

\[ e^{p+1} = A^{-1}B^p r^1 = A^{-1} \sum_{i=1}^{n} c_i (\lambda_i)^p y^i. \quad \text{(IV.6)} \]
The summation
\[ \sum_{i=1}^{n} c_i (\lambda_i)^p y^i = (\lambda_1)^p [c_1 y^1 + c_2 (\lambda_2/\lambda_1)^p y^2 + c_3 (\lambda_3/\lambda_1)^p y^3 + \ldots + c_n (\lambda_n/\lambda_1)^p y^n] \]
approaches to \( c_1 (\lambda_1)^p y^1 \) for sufficient large \( p \), because \( (\lambda_i/\lambda_1) \) is less than unity. Similarly, the summation
\[ \sum_{i=1}^{n} c_i (\lambda_i)^p (1 - \lambda_i) y^i \]
approaches to \( c_1 (\lambda_1)^p (1 - \lambda_1)y^1 \) for sufficient large \( p \). Thus, ultimately Equations IV.4, IV.5, and IV.6 will be reduced to
\[ r^{p+1} = c_1 (\lambda_1)^p y^1, \quad (IV.7) \]
\[ \Delta x^{p+1} = c_1 (\lambda_1)^p (1 - \lambda_1)A^{-1} y^1, \quad (IV.8) \]
\[ E^{p+1} = c_1 (\lambda_1)^p A^{-1} y^1. \quad (IV.9) \]
From these equations, we deduce that
\[ r^{p+1} = \lambda_1 r^p, \quad (IV.10) \]
\[ \Delta x^{p+1} = \lambda_1 \Delta x^p, \quad (IV.11) \]
\[ E^{p+1} = \lambda_1 E^p. \quad (IV.12) \]
Substituting \( \lambda_1 \), which we can evaluate from either Equation IV.10 or Equation IV.11, into Equation IV.12, we have
\[ E^{P+1} = x - x^{P+1} = \lambda_1 (x - x^P). \]

From this equation we solve for \( x \),

\[
x = \frac{(x^{P+1} - \lambda_1 x^P) / (1 - \lambda_1)}{(x^P + \Delta x^P - \lambda_1 x^P) / (1 - \lambda_1)}
\]

\[ = x^P + \Delta x^P / (1 - \lambda_1). \]

Notice that this equation is similar to the basic acceleration process equations III.19 and III.20 with \( \lambda_1 \) replaced by \( f_1 \).

In deriving the above equations, we have assumed that there is only one single dominant \( |\lambda_1| \). If \( |\lambda_1| \) is close to \( |\lambda_2| \), but \( |\lambda_2| > |\lambda_3| > \ldots > |\lambda_n| \), then for even quite large \( p \), we will have the following equations instead of equations IV.7, IV.8 and IV.9,

\[
r^{P+1} = c_1 (\lambda_1)^P y^1 + c_2 (\lambda_2)^P y^2,
\]

\[
\Delta x^{P+1} = c_1 (\lambda_1)^P (1 - \lambda_1) A^{-1} y^1 + c_2 (\lambda_2)^P (1 - \lambda_2) A^{-1} y^2,
\]

\[
E^{P+1} = c_1 (\lambda_1)^P A^{-1} y^1 + c_2 (\lambda_2)^P A^{-1} y^2.
\]

The linear relationship between successive vectors defined by Equations IV.10, IV.11, and IV.12 does not hold for this case. It is still possible to solve for \( x \) from five successive \( r \) (or \( x \)) vectors, but the work involved is too much to be considered worthwhile, unless we know in advance that the matrix \( B \) has a
set of eigenvalues satisfying the condition of this case.

Our second acceleration process is restricted to systems whose matrix $B$ has only a single dominant eigenvalue. The algorithm of this process is summarized below.

1. Apply the first acceleration algorithm for one complete cycle and save the last residual vector $r^{1}$.

2. Carry out the next iteration cycle and check to see if the current residual vector $r^{P}$ and the previous saved $r^{P-1}$ satisfying the condition specified by Equation IV.10.

3. If the condition is not satisfied, save the current residual vector $r^{P}$ and repeat step 2.

4. If the condition is satisfied, compute $x$ by adding to the current solution vector $x^{P}$ the amount $\Delta x^{P}/(1 - \lambda_{1})$, where $\lambda_{1}$ is determined from Equation IV.10.

The usefulness of the second acceleration process depends entirely on how fast the limit IV.10 can be reached. In practice, the condition IV.10 does not have to be satisfied exactly. We may apply Equation IV.13 to compute $x$ when the two successive vectors are approximately parallel to each other. The solution vector thus obtained usually meets the desired accuracy. If necessary, we can always continue the iteration process to revise the results. Test problem no. 3 given in Chapter V is solved by the second acceleration process.
V. COMPARISON OF METHOD

A. Measure of Computational Work

In order to compare the efficiency of various methods for solving a given problem, it is necessary to know the number of arithmetic operations involved. In this section, we will establish this information for the acceleration process and the projection method. The number of operations required for the Gauss-Seidel method and the Jacobi method, which will be used later for comparison, will also be given.

The computational work is usually measured in terms of multiplications (or divisions) and additions (or subtractions) required for one iteration cycle. Let’s look at the acceleration process described by Equations III.22 through III.27 first. The expression \((A_{i+1},A_i+1)(1 - f_i) = c_i\) in Equation III.22, and \((A_{i+1},A_i)/(A_i,A_i) = d_i\) in Equation III.23 are constant values and need be evaluated only once before the iteration starts. Therefore, the amount of operations required to establish these constants will not be included in the number of operations for each cycle.
<table>
<thead>
<tr>
<th>No. of multiplications</th>
<th>No. of additions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta x_{i+1}^k = (r^k, A_i + 1)/(c_i)$</td>
<td>$n + 1$</td>
</tr>
<tr>
<td>$\Delta x_i^k = -\Delta x_{i+1}^k d_i$</td>
<td>1</td>
</tr>
<tr>
<td>$r^{k+1} = r^k - \Delta x_i^k A_i - \Delta x_{i+1}^k A_{i+1}$</td>
<td>$2n$</td>
</tr>
<tr>
<td>$\Delta x_{i+1}^{k+1} = x_i^{k+1} + \Delta x_i^k$</td>
<td>1</td>
</tr>
<tr>
<td>$x_{i+1}^{k+1} = x_{i+1}^k + \Delta x_i^{k+1}$</td>
<td>1</td>
</tr>
</tbody>
</table>

Total no. of operations per step: $3n + 2$  
Total no. of operations per cycle: $3n^2 + 2n$  

The number of operations for the projection method is:

<table>
<thead>
<tr>
<th>No. of multiplications</th>
<th>No. of additions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_i^k = (r^k, A_i)/(A_i, A_i)$</td>
<td>$n + 1$</td>
</tr>
<tr>
<td>$r^{k+1} = r^k - \Delta x_i^k A_i$</td>
<td>$n$</td>
</tr>
<tr>
<td>$x_i^{k+1} = x_i^k + \Delta x_i^k$</td>
<td>1</td>
</tr>
</tbody>
</table>

Total no. of operations per step: $2n + 1$  
Total no. of operations per cycle: $2n^2 + n$  

The number of operations for the Gauss-Seidel method as well as for the Jacobi method are $n^2 - n$ multiplications and $n^2 - n$ additions per cycle. (Assume that the diagonal elements of the coefficient matrix $A$ are normalized initially.)
From these numbers, we may conclude that as far as the computational work is concerned, one cycle of the acceleration process is equivalent to 3/2 cycles of the projection method, and 3 cycles of the Gauss-Seidel method or the Jacobi method.

The Gauss elimination method requires \( n^3/3 + n^2/2 - n/3 \) multiplications and \( n^3/3 + n^2 - 5n/6 \) additions to obtain a solution. Taking into account the extra operations required for initialization by the acceleration process, we may say that the Gauss method is equivalent to \( n/9 \) cycles of the acceleration process.

B. Theoretical Comparison

The average rate of convergence and the asymptotic rate of convergence defined in Chapter III, section C provide means for comparison of different iterative methods. But, these can only be used for comparing methods for a particular problem. There is no way to judge the performance of methods in general. (This is probably the major reason why people are reluctant to use an iterative method. It is hard to predict how much effort is needed to solve a problem by an iterative method while one can always tell with full confidence how much work is involved to get a solution, good or bad, by a direct method). It is for this reason, we are excused from providing a general comparison of the acceleration process with other iterative methods.

---

\(^1\) See, for example, Fox (3, p. 176).
Since the acceleration process is supposed to accelerate the projection method, we would like to see if we can say something about the rate of convergence of these two methods in general.

From the results given in Chapter III, section A, one can verify that in one step of the acceleration process, the residual vector is reduced by the amount \( \frac{\Delta r_i^k + \Delta r_{i+1}^k}{1 - f_i} \).

If we want to accomplish exactly the same thing by the projection method, we have to repeat the \( i^{th} \) and the \( (i+1)^{th} \) steps of the projection method an infinite number of times, because,

\[
\frac{\Delta r_i^k}{1 - f_i} = \Delta r_i^k + \Delta_2 r_i^k + \ldots
\]

\[
= \Delta r_i^k \lim_{m \to \infty} [1 + f_i + (f_i)^2 + \ldots + (f_i)^m]
\]

and

\[
\frac{\Delta r_{i+1}^k}{1 - f_i} = \Delta r_{i+1}^k \lim_{m \to \infty} [1 + f_i + (f_i)^2 + \ldots + (f_i)^m].
\]

In practice, there is no need to carry on the operations beyond the point where \( (f_i)^m \) is much smaller than unity. If we terminate the series at, say, \( (f_i)^m = .0001 \), for an average value of \( f_i = .5 \), \( m = \ln .0001/\ln .5 \approx 13 \) approximately. Hence, one step of the acceleration process is equivalent to repeating a pair of steps of the projection method thirteen times. As far as numerical work is concerned, it is equivalent to \( 2 \times 13 \times \frac{2}{3} = 17 \) steps. Since \( m \) is related to \( \Delta r_i^k \) instead of the residual vector \( r_i^k \) itself, if \( \Delta r_i^k \) is less than \( r_i^k \), we may want to cut off the series at a point earlier than that determined by
m. For the exceptional case, \( f_i = 0 \), i.e. \( A_i \) is orthogonal to \( A_{i+1} \), the acceleration process is identical to the projection method. We do admit that the above discussion is valid only when the residual vector at the beginning of each step is the same for both methods. If \( \Delta r^k_i \) or \( \Delta r^k_{i+1} \) of the projection step is much larger than that of the acceleration step, the acceleration process may converge slower in that particular step. We doubt this can happen in so many steps as to make the projection method favorable over the acceleration process. This statement is backed up by the many sample problems tested. So far, we have not encountered a single such case. It seems safe to say that the rate of convergence of the acceleration process is, in general, faster than that of the projection method.

C. Numerical Comparison

The available methods for solving systems of linear equations are numerous. It is impossible to compare the acceleration process with each one of them. Besides, as we stated earlier in section B the computational work involved in an iterative method depends entirely on the property of the system to be solved. It is meaningless to draw any definite conclusions from some test problems. In this section, we present some numerical results only in the attempt to show the merit of the acceleration process for the problems tested. All the results back up the claim that the acceleration process indeed accelerates the rate of convergence of the basic projection
method considerably. The results of the first two test problems show that the basic acceleration process converges faster than all the other methods tested while the rate of convergence of the projection method falls behind all of them. The last test problem shows that the basic acceleration process converges slower than the Gauss-Seidel method and the Jacobi method. But the second acceleration process can be applied in this case and gives a very satisfactory result. By observing the numerical data given in the following tables, one might get an impression that the acceleration process is not much better than the Gauss-Seidel method. It should be reminded that we purposely selected these test problems which are known to be very well-conditioned for the Gauss-Seidel method.
Test problem 1.
\[
\begin{bmatrix}
3 & 2 & 0 \\
0 & 2 & 1 \\
1 & 0 & 2
\end{bmatrix}
\begin{bmatrix}
1 \\
1 \\
x
\end{bmatrix}
=
\begin{bmatrix}
5 \\
3 \\
b
\end{bmatrix}
\]

Table 1. Length of residual vector

<table>
<thead>
<tr>
<th>Method</th>
<th>Number of iteration cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3</td>
</tr>
<tr>
<td>Acceleration process</td>
<td>.0000 (^a)</td>
</tr>
<tr>
<td>Projection</td>
<td>.2162</td>
</tr>
<tr>
<td>Jacobi</td>
<td>1.0929</td>
</tr>
<tr>
<td>Gauss-Seidel</td>
<td>.5122</td>
</tr>
</tbody>
</table>

\(^a\)The underlined numbers are results obtained by the various methods with approximately the same number of operations.

Table 2. Solution vector

<table>
<thead>
<tr>
<th>Method</th>
<th>Spectral radius of iteration matrix</th>
<th>No. of cycles</th>
<th>(x_1)</th>
<th>(x_2)</th>
<th>(x_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acceleration process</td>
<td>.011</td>
<td>3</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>Projection(^a)</td>
<td>.43</td>
<td>5</td>
<td>1.1765</td>
<td>1.3056</td>
<td>.7926</td>
</tr>
<tr>
<td>Jacobi</td>
<td>.55</td>
<td>9</td>
<td>1.0046</td>
<td>1.0046</td>
<td>1.0046</td>
</tr>
<tr>
<td>Gauss-Seidel</td>
<td>.41</td>
<td>9</td>
<td>1.0005</td>
<td>1.0004</td>
<td>.9997</td>
</tr>
</tbody>
</table>

\(^a\)The iteration matrix \(B\) of the projection method is defined by \(r^{k+1} = Br^k\). The construction of \(B\) is given by Fox (3, p. 314).
### Table 3. Length of residual vector

<table>
<thead>
<tr>
<th>Method</th>
<th>Number of iteration cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4</td>
</tr>
<tr>
<td>Acceleration process</td>
<td>.0000</td>
</tr>
<tr>
<td>Projection</td>
<td>.9870</td>
</tr>
<tr>
<td>Jacobi</td>
<td>2.5905</td>
</tr>
<tr>
<td>Gauss-Seidel</td>
<td>.2585</td>
</tr>
</tbody>
</table>

### Table 4. Solution vector

<table>
<thead>
<tr>
<th>Method</th>
<th>Spectral radius of iteration matrix</th>
<th>No of cycles</th>
<th>x₁</th>
<th>x₂</th>
<th>x₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acceleration process</td>
<td>.049</td>
<td>4</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>Projection</td>
<td>.63</td>
<td>6</td>
<td>1.2348</td>
<td>1.1365</td>
<td>.7002</td>
</tr>
<tr>
<td>Jacobi</td>
<td>.75</td>
<td>12</td>
<td>.9581</td>
<td>.9812</td>
<td>.9718</td>
</tr>
<tr>
<td>Gauss-Seidel</td>
<td>.41</td>
<td>12</td>
<td>1.00001</td>
<td>1.0000</td>
<td>.9999</td>
</tr>
</tbody>
</table>
Test problem 3.

\[
\begin{bmatrix}
4 & -1 & 2 \\
-1 & 2 & 0 \\
3 & 0 & 5 \\
\end{bmatrix}
\begin{bmatrix}
2 \\
3 \\
4 \\
\end{bmatrix}
= 
\begin{bmatrix}
13 \\
4 \\
26 \\
\end{bmatrix}
\]

\[
A \quad \begin{array}{c} x \\ \end{array} \quad \begin{array}{c} b \\
\end{array}
\]

Table 5. Length of residual vector

<table>
<thead>
<tr>
<th>Method</th>
<th>Number of iteration cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3</td>
</tr>
<tr>
<td>Acceleration</td>
<td>.0000&lt;sup&gt;a&lt;/sup&gt;</td>
</tr>
<tr>
<td>process</td>
<td></td>
</tr>
<tr>
<td>Projection</td>
<td>3.7695</td>
</tr>
<tr>
<td>Jacobi</td>
<td>5.6412</td>
</tr>
<tr>
<td>Gauss-Seidel</td>
<td>0.5193</td>
</tr>
</tbody>
</table>

<sup>a</sup>This result is obtained by the second acceleration process. The basic acceleration process converges in 14 cycles.

Table 6. Solution vector

<table>
<thead>
<tr>
<th>Method</th>
<th>Spectral radius of iteration matrix</th>
<th>No. of cycles</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acceleration</td>
<td>.38</td>
<td>3</td>
<td>2.0000</td>
<td>3.0000</td>
<td>4.0000</td>
</tr>
<tr>
<td>process</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Projection</td>
<td>.85</td>
<td>5</td>
<td>3.3905</td>
<td>4.1860</td>
<td>2.9790</td>
</tr>
<tr>
<td>Jacobi</td>
<td>.65</td>
<td>9</td>
<td>2.0407</td>
<td>2.9674</td>
<td>4.0392</td>
</tr>
<tr>
<td>Gauss-Seidel</td>
<td>.43</td>
<td>9</td>
<td>2.0013</td>
<td>3.0007</td>
<td>3.9992</td>
</tr>
</tbody>
</table>
VI. COMPUTER PROGRAM IMPLEMENTATION

The basic acceleration process and the second acceleration process described in Chapters III and IV, respectively, are implemented into a FORTRAN subroutine program SPEEDY. It has been tested on an IBM/360/65 computer.

The convergence criterion is based upon the length of the residual vector at the end of an iteration cycle. This is, of course, an arbitrary choice. One may, for example, check on the change of the solution vector and terminate the iteration process when the change is negligible.

The second acceleration process is applied when all the following conditions are satisfied.

1. The user requests the program to apply the second acceleration process when it is possible by specifying a parameter in the argument list.

2. If we let \( r^{k-2}, r^{k-1}, \) and \( r^k \) be the residual vectors obtained at the end of three successive iteration cycles, then the difference between the two ratios \( \frac{\| r^{k-1} \|}{\| r^{k-2} \|} \) and \( \frac{\| r^k \|}{\| r^{k-1} \|} \) must be less than or equal to a predefined tolerance, TOLRB.

3. The two residual vectors must satisfy the condition

\[
\sum_{i=1}^{n} \left( r_i^{k-1} - r_i^k \right)^2 \leq TOLRB,
\]

where \( n \) is the dimension of the vector \( r \).
Conditions 2 and 3 are not checked if condition 1 is not satisfied. Likewise, condition 3 is not checked if conditions 1 and 2 are not satisfied. Condition 2 alone is not sufficient to guarantee that Equation IV.10 is satisfied, but it is a necessary condition. Since the lengths of the residual vectors are used to check convergence and are readily available, we check condition 2 first instead of checking condition 3 directly for the reason of saving computer time.

The arguments used in the subroutine are described below:

A: Coefficient matrix $A$.

B: Right-hand vector $b$.

X: At the beginning, it contains the initial solution vector $x^0$ passed from the calling program. Subsequently, it contains the current solution vector $x^k$. At the end, it returns the final solution vector to the calling program.

XX: The solution vector obtained at the immediate preceding cycle.

RC: The current residual vector $r^k$.

RP: The preceding residual vector $r^{k-1}$.

C: The constants $1/[(A_i^{T} A_i + 1)(1 - f_i)]$.

D: The constants $A_i^{T} A_i / [(A_i^{T} A_i + 1) A_i^{T} A_i (1 - f_i)]$.

TOLRA: The iteration process terminates when the length of the residual vector is less than or equal to TOLRA.

TOLRB: See discussion in the previous paragraph.
ITER: The iteration process terminates when the number of iteration cycles performed equals to ITER.

N: The dimension of the system to be solved.

M: The data set number for output of intermediate results.

NN: Defines the size of all arrays (NN must be greater than or equal to N).

COND: It is set to zero if the process converges, otherwise, it is set to one.

IGO: It is set to one by the calling program if the second acceleration process is desired, otherwise, it is set to zero.

A list of the subroutine SPEEDY and a sample calling program is given in the Appendix.
VII. SUMMARY

The acceleration process of the projection method presented in this thesis, in the author's opinion, is not just another iterative method among the many; it is a method worth the consideration for solving systems of linear equations. It, of course, should not be used indiscriminately. For a general system, a direct method such as the Gauss elimination method usually requires much less computational work than any iterative method one can name. It is in those occasions where a direct method is undesirable that an iterative method is useful. For example, when the coefficient matrix is sparse, the number of operations per cycle of an iterative method will be reduced considerably, and what is more important is the savings in storage usage. The drop in speed due to insufficient high speed core storage for solving a large system by a direct method cannot be ignored. Among the iterative methods themselves, we do not recommend to use the acceleration process blindly either. For a system whose property is well known to be suitable for, say, the Gauss-Seidel method, there is no reason why one should have to try the acceleration process. On the other hand, when it is in doubt that the system can be solved efficiently by any particular method, the acceleration process then should be included in the consideration. Specially for those systems whose convergence cannot be guaranteed by many well-known iterative methods, the acceleration process is highly
recommended. Because, iterative methods which do not impose restrictions on the systems to be solved are usually as slow as the projection method, and we are almost one-hundred percent sure that the acceleration process can do a better job than the projection method.

As far as future research is concerned, we feel that the following items are worthwhile to study:

1. A more concise method for determining the rate of convergence of the acceleration process than the one discussed in Chapter III, section C. It is preferable to determine the rate of convergence from the property of the coefficient matrix \( A \) rather than from the iteration matrix \( B \) (III.35). This problem may not be solved easily if it can be solved at all. But, we should try to see if something can be derived for certain special systems such as the diagonally dominant systems for example.

2. A similar study on the applicability of the second acceleration process related to the matrix \( A \).

3. When the matrix \( A \) is symmetric and positive-definite, the projection method can be simplified\(^1\). The acceleration process, however, cannot be applied to the simplified method. It seems that one should be able to develop an acceleration process for that case also.

\(^1\)See Fox (3, pp. 207-208).
VIII. LITERATURE CITED


IX. ACKNOWLEDGMENT

The author wishes to express his deepest gratitude to Dr. Roy Keller for his generous guidance and encouragement during the preparation of this dissertation.

A part of the research is supported by the author's employer, Simpson Gumpertz and Heger, Inc. in Cambridge, Massachusetts, which is gratefully acknowledged.
X. APPENDIX

C SAMPLE CALLING PROGRAM
C
IMPLICIT REAL*8 (A-H,O-Z)
INTEGER COND
DIMENSION A(10,10),B(10),X(10),XX(10),RC(10),C(10),D(10),*
RP(10)
1000 FORMAT(3I5,2F10.0)
1100 FORMAT(8F10.0)
2000 FORMAT('1 ITER=',I5,' TOLRA=',D16.6,' TOLRB=',D16.6)
2100 FORMAT(8D16.6)
2200 FORMAT(' VECTOR B'/(8D16.6))
2300 FORMAT('0 INITIAL SOLUTION VECTOR X'/(8D16.6))
2400 FORMAT('0 PROCESS FAILED TO CONVERGE')
2500 FORMAT('0 PROCESS CONVERGED')
2600 FORMAT('0 FINAL SOLUTION VECTOR'/(8D16.6))
2700 FORMAT('0 FINAL RESIDUAL VECTOR'/(8D16.6))
2800 FORMAT('0 MATRIX A')
NN=10
M=3
10 READ(1,1000,END=50) N,ITER, IGO,TOLRA,TOLRB
WRITE(3,2000) ITER,TOLRA,TOLRB
WRITE(3,2800)
DO 20 I=1,N
READ(1,1100) (A(I,J),J=1,N)
20 WRITE(3,2100) (A(I,J),J=1,N)
READ(1,1100) (B(I),I=1,N)
WRITE(3,2200) (B(I),I=1,n)
READ(1,1100) (X(I),I=1,N)
CALL SPEEDYC.,
               B,X,XX,RC,RP,C,D,TOLRA,TOLRB,ITER,N,M,NN,
               COND,IGO)
IF(COND.EQ.0) GO TO 30
WRITE(3,2400)
GO TO 40
30 WRITE(3,2500)
40 WRITE(3,2600) (X(I),I=1,N)
WRITE(3,2700) (RC(I),I=1,N)
GO TO 10
50 STOP
END
SUBROUTINE SPEEDY(A,B,X,XX,RC,RP,C,D,TOLRA,TOLRB,ITER,N, *
* M,NN,COND,IGO)
IMPLICIT REAL*8 (A-H,O-Z)
DIMENSION A(NN,NN),B(NN),X(NN),XX(NN),RC(NN),RP(NN),C *
* (NN),D(NN)
INTEGER CYCLE,STEP ,COND
1000 FORMAT('CONSTANTS F(I) REF.EQ. III. '/(8D16.6))
1100 FORMAT('CYCLE',14,' STEP',IV9X,'X VECTOR=',6D18.6 /
* (20X,6D18.6))
1200 FORMAT(9X,'LENGTH OF R VECTOR=',6D18.6/9X,'R VECTOR', *
* 6D18.6/(20X,6D18.6))
1300 FORMAT('PROCESS FAILED TO CONVERGE IN',I4,' CYCLES')
COND=0
C
C INITIALIZATION
C
DO 20 I=1,N
J=I+1
IF(I.EQ.N) J=1
AII=0.DO
AIJ=0.DO
DO 10 K=1,N
AII=AII+A(K,I)**2
10 AIJ=AIJ+A(K,J)*A(K,I)
RC(I)=AII
20 XX(I)=AIJ
DO 30 I=1,N
J=I+1
IF(I.EQ.N) J=1
30 C(I)=XX(I)**2/(RC(I)*RC(J))
WRITE(M,1000) (C(I),I=1,N)
CI=1.DO=(RC(1)*C(1))/DO-C(1))
D1=XX(1)/RC(2)
RC2=RC(2)
DO 40 I=1,N
J=I+1
IF(I.EQ.N) J=1
CI=1.DO/(RC(J))*C(1))
40 D(I)=XX(I)/RC(I)
C
C C(I)=1/A(I+1)'A(I+1)(1-F(I)  REF. EQ. III.23
C D(I)=A(I+1)'A(I)/A(I)'A(I)  REF. EQ. III.24
C
MCYCLE=0
50 MCYCLE=MCYCLE+1
CYCLE=0 ,
STEP=0
C
C COMPUTE INITIAL RESIDUAL VECTOR AND IT'S LENGTH
C
R1=0.DO
DO 70 I=1,N
RI=B(I)
DO 60 J=1,N
60 RI=RI-A(I,J)*X(J)
RC(I)=RI
70 RL=RL+RI*RI
RL=DSQRT(RL)
WRITE(M,1100) MCYCLE,STEP,(X(I),I=1,N)
WRITE(M,1200) RL,(RC(I),I=1,N)
IF(RL.LE.TOLRA) RETURN

C
C FIRST ITERATION STEP OF THE FIRST CYCLE
C
CYCLE=1
STEP=1
J=2
DXJ=0.DO
DO 80 I=1,N
80 DXJ=DXJ+RC(I)*A(I,2)
DXJ=DXJ/RC2
DXI=0.DO
DO 90 I=1,N
90 DXI=(RC(I)-DXJ*A(I,2))*A(I,1)+DXI
DXI=DXI*C1
DXJ=DXJ-DXI*D1
GO TO 130

C
C REGULAR ITERATION STEP
C
100 CYCLE=CYCLE+1
MCYCLE=MCYCLE+1
IF(MCYCLE.GT.ITER) GO TO 230
STEP=0
110 STEP=STEP+1
J=STEP+1
IF(STEP.EQ.N) J=1
DXJ=0.DO
DO 120 K=1,N
120 DXJ=DXJ+RC(K)*A(K,J)
DXJ=DXJ*C(STEP)
DXI=-DXJ*D(STEP)
130 DO 140 K=1,N
140 RC(K)=RC(K)-DXJ*A(K,J)-DXI*A(K,STEP)
X(J)=X(J)+DXJ
X(STEP)=X(STEP)+DXI
IF(STEP.LT.N) GO TO 110

C
C CHECK TO SEE IF THE SECOND ACCELERATION PROCESS COULD
C BE APPLIED.
C
RR=0.DO
DO 150 K=1,N
150 RR=RR+ RC(K)**2
RR=DSQRT(RR)
WRITE(M,1100) MCYCLE,STEP,(X(I),I=1,N)
WRITE(M,1200) RR,(RC(I),I=1,N)
IF(RR.LT.TOLRA) RETURN
IF(IGO.EQ.0) GO TO 100
IF(CYCLE.GT.L) GO TO 160
R1=RR
GO TO 100
160 IF(CYCLE.GT.2) GO TO 170
R2=RR
GO TO 210
C
C CHECK TO SEE IF THE SECOND ACCELERATION PROCESS CAN BE
C APPLIED
C
170 RATIO=RR/R2
IF(DABS(RATIO-R2/R1).GT.TOLRB) GO TO 200
DD=0.DO
CC=0.DO
DO 180 K=1,N
CC=CC+DABS(RC(K))
180 DD=(RC(K)-RP(K)*RATIO)**2+DD
DD=DSQRT(DD)/CC
IF(DD.GT.TOLRB) GO TO 200
C
C APPLY SECOND ACCELERATION PROCESS
C
RATIO=1.DO/(1.DO-RATIO)
DO 190 I=1,N
J=I+1
IF(I.EQ.N) J=1
190 X(I)=XX(I)+(X(I)-XX(I))*RATIO
GO TO 50
200 R1=R2
R2=RR
210 DO 220 I=1,N
RP(I)=RC(I)
220 XX(I)=X(I)
GO TO 100
230 WRITE(M,1300) MCYCLE
COND=1
RETURN
END