IS-3397

ALGORITHMS FOR PROJECTION METHODS FOR SOLVING LINEAR SYSTEMS OF EQUATIONS

R. L. Wainwright
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-3397

ALGORITHMS FOR PROJECTION METHODS FOR SOLVING LINEAR SYSTEMS OF EQUATIONS

R. L. Wainwright 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

This paper develops a new projection method for any dimension which is equivalent to the conventional projection method. The new projection method requires less than half the arithmetic operations required by the old projection method per iteration step. In addition the number of operations per component per iterative step for the new method is shown to be independent of the dimension of the method used. Despite the additional overhead calculations required for the new projection method this study proves that the solution is obtained in fewer arithmetic operations than required by the conventional projection method. This is true for any initial vector, using any dimensional method, and for any nonsingular coefficient matrix.

Hybrid projection methods are also developed. They are shown to require less than half the arithmetic operations of the conventional method per iterative step. Operations required per component per iterative step are independent of the dimension of the method used.

A variety of other topics are discussed. The iteration matrix for the new projection method is easily determined and is presented for the one and two-dimensional cases. Two initial vector algorithms are developed for use with the new projection method. The new projection method may be subject to rounding error propagation, hence, an algorithm is presented for the prevention of rounding error propagation.

Finally, several test problems are included to demonstrate the superiority of the new and hybrid projection methods over the conventional projection methods.

I. AN INDEX TO SYMBOLS AND NOTATIONAL DEFINITIONS

1. a_i represents the i^{th} column of A
2. a_{ij} represents the inner product of the i^{th} and j^{th} column vectors of A, i.e., (a_i, a_j)
3. a_{ij}^k Equation 4.19a
4. A represents the matrix of coefficients - Equation 3.1
5. A_i represents the i^{th} row of A
6. b represents the constant vector of the linear system Equation 3.1
7. β_i represents the back substitution function using Gaussian elimination to obtain the i^{th} component of the solution vector - Equation 3.17
8. c_{ij}^k Equation 4.19b
9. $\cos \Theta_{ij}$ Equation 3.11b
10. d_i^k represents the change to the i^{th} component of the approximate solution vector at the k^{th} cycle
11. $\tilde{d}_j^n = \sum_{i=1}^{n} d_j^i$ Equation 4.3
12. D_{ijp} Equation 3.11a
13. e_i represents the i^{th} column of an n by n identity matrix

14. f_i Equations 6.4, 6.15, 6.28
15. g_{ij} Equation 4.13
16. G Equations 4.8, 6.12
17. H Equations 4.9, 6.31
18. K Equation 6.31
19. L represents a lower triangular matrix defined by 4.11, 4.21
20. M represents the iteration matrix
21. n represents the number of rows and columns in A
22. P Equation 6.5c
23. Q Equations 6.5c, 6.18
24. r^k represents the k^{th} residual vector defined by 3.2
25. R Equations 6.5c, 6.19
26. s_i Equation 3.9
27. t_i Equation 3.10
28. t_{ij} Equation 4.12
29. T Equation 6.31
30. U represents an upper triangular matrix defined by 4.11, 4.21
31. w represents the number of steps per cycle which is defined as $\lfloor (n + m - 1)/m \rfloor$ where m is the dimension of the projection method used

32. W_{ij} Equation 6.12

33. x represents the solution vector - Equation 3.1

34. x^k represents the approximate solution vector after the k^{th} cycle

35. Z_i^k Equations 6.3, 6.14, 6.27

II. INTRODUCTION

There currently exist many methods for solving systems of linear algebraic equations denoted by $Ax = b$, where A is an n by n nonsingular matrix and both x and b are n element column vectors. They can be classified into one of two categories: direct methods and iterative (indirect) methods. In direct methods the solution is obtained by performing a fixed number of arithmetic operations. There are many direct methods, but all seem to be mere variations of a basic few. The simplest and most widely used is the elimination method attributed to Gauss (3, chapter 3). Direct methods are known to be faster than iterative methods, but sometimes round-off errors can accumulate to a point where the solution is inaccurate and perhaps useless. Because of the considerable amounts of data and storage involved and the large number of arithmetic operations, direct methods are useful in practice on current computers for systems of up to about a hundred equations (5, p. 119). For large or illconditioned systems of equations direct methods may become iterative in nature. This is because the accumulated round-off errors in direct methods imply performing the direct computation repeatedly with a better approximation, hence it becomes iterative. One such example is an iterative process of Gaussian elimination (3, p. 143). In iterative

methods a solution is obtained by successive approximations. In these methods the matrix A can be written in the following iterative form: $x^{k+1} = Mx^k + c$. One begins with an initial approximation vector, x^0, either guessed or from a direct method, then an improvement is made to the old approximation x^k, to obtain a new approximation x^{k+1}, according to the above scheme. This process is repeated until the desired accuracy of the solution is achieved. M is called the iteration matrix, and given any initial guess, the iterative process will converge to a solution depending on the spectral radius of M (19, p. 13). Iterative methods are sometimes classified as stationary or nonstationary. If the function to obtain x^{k+1} from x^k is dependent on the step parameter, k, then the method is nonstationary and if the function is independent of k the method is said to be stationary.

The idea of solving systems of linear equations by iterative methods dates back at least to Gauss (1823) when he proposed the Method of Least Squares (1, p. 144; 19, p. 1). Gauss considered his method so simple that he did not explain it completely. He once wrote his pupil, Gerling: "The indirect procedure can be done while one is half asleep, or is thinking about other things" (1, p. 146). This method was further developed by Jacobi and later by one of Jacobi's pupils, Seidel. In Jacobi's method D, L,

and U are defined as $A = D + (L + U)$ and in the Gauss-Seidel method $A = (D + L) + U$, where D is a diagonal matrix and L and U are lower and upper triangular matrices, respectively. The iteration matrix, M, for the Jacobi method becomes $M = -D^{-1}(L + U)$ and for Gauss-Seidel, $M = -(D + L)^{-1}U$. Gauss-Seidel is a single-step method in that only one component of the solution vector is changed at each iteration (i.e., going from x^k to x^{k+1}). Jacobi, on the other hand, is a total step process. Each component of the approximation to the solution vector is changed at each iteration step.

Among the class of iterative methods is a class of methods called relaxation methods. Gauss' Method of Least Squares and Gauss-Seidel are two examples. In relaxation methods attention is given to the residuals, r_i where $r_i = b_i - A_i x$ and A_i is the i^{th} row of A and $A_i x$ is a vector product. Relaxation methods consist of changing the components in x in some fashion to reduce the magnitudes of residuals to negligible amounts, hence yielding a solution. Among the class of relaxation methods are the gradient methods. Gradient methods are those which minimize the sum of the squares of the residual vector $r = b - Ax$, i.e., the quantity $r'r = (b' - x'A')(b - Ax)$ is minimized at each iterative step. An extensive study of gradient methods for solving linear systems can be found in Gastinel (5), Fox (3), Varga (19)

and others. An extensive bibliography of references concerning all methods for solving systems of linear equations can be found in Householder (9) and Forsythe (2). Among the class of gradient methods are the methods of projection so named because the schemes have a geometric projection interpretation. There are many projection methods such as Kaczmarz's as found in (17, 1, 10, 5) and Cimmino's as found in (17, 10, 1, 5), and a method developed by the Raytheon Company as found in (17). However, the method developed by A. de la Garza (4), in 1951, has since become known as the projection method. Garza's projection method is so named because the approximate solution is improved by projecting the residual vector onto a subspace determined by one or more of the column vectors of the coefficient matrix.

Iterative methods are used mainly in those problems for which convergence is known to be rapid so the solution is ascertained with much less work than a direct method or for large systems when direct methods would be costly, inaccurate and storage dependent. A distinct advantage of iterative methods is that rounding errors do not accumulate, for they are restricted to the last operation. Among the disadvantages are that convergence may be very slow, or perhaps convergence will not occur at all unless the system of equations satisfies certain conditions. A further disadvantage is that some iterative methods such as Southwell's

Method of Relaxation (15) are designed to be done with human insight using hand calculators, and when implemented by computer programs the needed human insight goes away. Since the only requirement for convergence of the projection method is that the coefficient matrix be nonsingular, the concern is shifted to the rate of convergence, which is generally slow (a characteristic of gradient methods). Proof of convergence of the projection method can be found in (4) and (12). The rate of convergence of the projection method depends on the properties of the linear system as well as:

(1) the number of iteration steps. This is dependent on the dimension of the method used as well as which column(s) of the coefficient matrix to project the residual vector onto at each step.

(2) the number of arithmetic operations per step. This may or may not be a function of the dimension of the method used.

Much work has been done with respect to (1). Shen (14) proposed a two-dimensional algorithm for accelerating the one-dimensional projection method. Pyron (13) in his development of a two-dimensional projection method noted that the pairing of column vectors of the coefficient matrix based on the angles between the vectors significantly influenced the rate of convergence. Georg (6) developed some alternative bases for ordering the columns of A for two-

dimensional projection methods. Tokko (18) developed an a priori process for grouping the column vectors of the coefficient matrix in triples which significantly increased the rate of convergence. This dissertation takes a close look at (2) and develops the following methods and observations:

(a) A new projection method is developed for any dimension which is equivalent to the old projection method, hence has the same convergence conditions but requires less than half the number of arithmetic operations per iteration step.

(b) The number of operations per component per iterative step of the new projection method is independent of the dimension of the method used.

(c) The iteration matrix for the new projection method is easily ascertained where it is a virtual impossibility in the old projection method.

(d) Simple and multiple hybrid projection methods are also developed for any dimension from the new projection method. They too are equivalent to the old projection method and require half the number of arithmetic operations per iteration step.

(e) The number of operations for the simple hybrid projection method per component per iterative step is independent of the dimension of the method used.

(f) A technique for the prevention of round-off error propagation is developed for the new and hybrid projection methods.

III. THE PROJECTION METHOD

A. Basic Review

For a system of linear equations defined by

$$Ax = b \tag{3.1}$$

the one-dimensional projection method can be derived by minimizing the quadratic form (r^k, r^k) where r^k is the residual vector of the k^{th} iteration defined by

$$r^k = b - Ax^k \quad . \tag{3.2}$$

The approximation to the solution vector is modified one component at a time at each iteration by the following scheme

$$x^{k+1} = x^k + d_i^k e_i \tag{3.3}$$

where d_i^k is the change to the i^{th} component of x at the k^{th} iteration and e_i is the i^{th} column vector of an n by n identity matrix. Minimizing (r^{k+1}, r^{k+1}) results in

$$d_i^k = (r^k, a_i)/(a_i, a_i) \tag{3.4}$$

where a_i is the i^{th} column of A. A proof of this can be found in Fox (3, p. 205). The residual vector after the k^{th} step can be found by

$$r^{k+1} = r^k - d_i^k a_i \quad . \tag{3.5}$$

A two-dimensional projection method minimizes (r^{k+1}, r^{k+1}) while changing two components in the approximate solution vector. The results for the two-dimensional projection method for changing the i^{th} and j^{th} components at the k^{th} iteration are (13, p. 12).

$$d_i^k = \frac{(r^k, a_j)\, a_{ij} - (r^k, a_i)\, a_{jj}}{(a_{ij})^2 - a_{ii}\, a_{jj}} \tag{3.6a}$$

$$d_j^k = \frac{(r^k, a_i)\, a_{ij} - (r^k, a_j)\, a_{ii}}{(a_{ij})^2 - a_{ii}\, a_{jj}} \tag{3.6b}$$

where a_{ij} is the vector product of the i^{th} and j^{th} columns of the coefficient matrix (a_i, a_j). The residual vector after each step is computed by

$$r^{k+1} = r^k - d_i^k\, a_i - d_j^k\, a_j \quad . \tag{3.7}$$

A three-dimensional projection method minimizes (r^{k+1}, r^{k+1}) while changing three components in the approximate solution vector. The results of the three-dimensional projection method for changing the i^{th}, j^{th} and p^{th} components at the k^{th} iteration is given by (18, p. 11).

$$d_i^k = \frac{1}{D_{ijp}}[(r^k, a_i)\, s_i + (r^k, a_j)\, t_i + (r^k, a_p)\, t_j] \tag{3.8a}$$

$$d_j^k = \frac{1}{D_{ijp}}[(r^k, a_j)\ s_j + (r^k, a_i)\ t_i + (r^k, a_p)\ t_p] \tag{3.8b}$$

$$d_p^k = \frac{1}{D_{ijp}}[(r^k, a_p)\ s_p + (r^k, a_j)\ t_p + (r^k, a_i)\ t_j] \tag{3.8c}$$

where

$$s_i = (1 - \cos^2\theta_{jp})/a_{ii} \tag{3.9a}$$

$$s_j = (1 - \cos^2\theta_{ip})/a_{jj} \tag{3.9b}$$

$$s_p = (1 - \cos^2\theta_{ij})/a_{pp} \tag{3.9c}$$

$$t_i = (\cos\theta_{ip}\ \cos\theta_{jp} - \cos\theta_{ij})/(a_{ii}\ a_{jj})^{1/2} \tag{3.10a}$$

$$t_j = (\cos\theta_{ij}\ \cos\theta_{jp} - \cos\theta_{ip})/(a_{ii}\ a_{pp})^{1/2} \tag{3.10b}$$

$$t_p = (\cos\theta_{ij}\ \cos\theta_{ip} - \cos\theta_{jp})/(a_{jj}\ a_{pp})^{1/2} \tag{3.10c}$$

$$D_{ijp} = 1 + 2\ \cos\theta_{ij}\ \cos\theta_{ip}\ \cos\theta_{jp} - \cos^2\theta_{ij} - \cos^2\theta_{ip} - \cos^2\theta_{jp} \tag{3.11a}$$

and

$$\cos\theta_{ij} = \frac{(a_i, a_j)}{(a_i, a_i)^{1/2}(a_j, a_j)^{1/2}} \quad . \tag{3.11b}$$

The next residual vector after each step is computed by

$$r^{k+1} = r^k - d_i^k a_i - d_j^k a_j - d_p^k a_p \quad . \tag{3.12}$$

There are obviously more computations involved as the dimension of the method increases. This is primarily the reason why higher dimensional methods have not been studied in any great detail to date.

B. An m-Dimensional Projection Method

One can easily summarize the process involved in the development of an m-dimensional projection method by the following theorem:

Theorem 3.1

The changes to m components of the approximate solution vector at the k^{th} step of an m-dimensional projection method used to solve a linear system of n equations ($n \geq m$) are given as the solution to the following symmetric system of m linear equations.

$$a_{11} d_1^k + a_{12} d_2^k + a_{13} d_3^k + \ldots a_{1,m} d_m^k = (r^k, a_1)$$

$$a_{21} d_1^k + a_{22} d_2^k + a_{23} d_3^k + \ldots a_{2,m} d_m^k = (r^k, a_2)$$

$$\vdots$$

$$a_{m,1} d_1^k + a_{m,2} d_2^k + a_{m,3} d_3^k + \ldots a_{m,m} d_m^k = (r^k, a_m)$$

where $x_1, x_2, x_3, \ldots, x_m$ are m arbitrary components of the approximate solution vector.[1]

Theorem 3.1 is a well known theorem for projection methods and a proof can be found in Householder (8). Various properties of the m-dimensional projection method are listed below.

The residual vector, r^{k+1}, obtained after the k^{th} iteration step is orthogonal to each of the m columns used in the k^{th} step (8, p. 30). That is

$$(r^{k+1}, a_1) = 0$$

$$(r^{k+1}, a_2) = 0$$

$$\vdots$$

$$(r^{k+1}, a_{m-1}) = 0$$

$$(r^{k+1}, a_m) = 0 \quad . \tag{3.13}$$

The k + 1 approximation to the solution vector is obtained by

$$x^{k+1} = x^k + d_1^k e_1 + d_2^k e_2 + \ldots d_m^k e_m \tag{3.14}$$

hence

---

[1] A comma sometimes separates the subscripts of a. This has no special meaning and is used only for clarity, i.e., $a_{ij} = a_{i,j}$.

$$r^{k+1} = b - Ax^{k+1}$$

$$= b - Ax^k - Ad_1^k e_1 - Ad_2^k e_2 - \ldots Ad_m^k e_m$$

$$= r^k - d_1^k a_1 - d_2^k a_2 - \ldots d_m^k a_m \quad . \qquad (3.15)$$

By substituting Equation 3.15 into Equation 3.13 one obtains the following system of equations:

$$(r^k - d_1^k a_1 - d_2^k a_2 - \ldots d_m^k a_m, a_1) = 0$$

$$(r^k - d_1^k a_1 - d_2^k a_2 - \ldots d_m^k a_m, a_2) = 0$$

$$\vdots$$

$$(r^k - d_1^k a_1 - d_2^k a_2 - \ldots d_m^k a_m, a_m) = 0 \quad . \qquad (3.16)$$

Note that by expanding the inner products of Equation 3.16 the linear system of theorem 3.1 is obtained.

C. Cost Analysis of the m-Dimensional Projection Method

By some a priori criteria which is independent of the work presented in this dissertation

$$w = \left\lfloor \frac{n+m-1}{m} \right\rfloor$$

groups of m columns of the coefficient matrix are determined

in such a way that each column is included in at least one group. $\lfloor y \rfloor$ represents the floor function as described by Iverson (11, p. 12) which is defined to be greatest integer less than or equal to y where y is any arithmetic expression. Every w iterative steps constitute a cycle, i.e., every element of approximate solution vector is changed at least once.

Each iterative step involves solving a symmetric system of m linear equations. Usually this system is solved by some direct method, such as Gaussian elimination where first the coefficient matrix is transformed into an upper triangular matrix and back substitution is performed to obtain an m element solution vector.

From cycle to cycle the same w linear systems are solved over and over with only the constant vector changing. Hence, triangularizing these matrices need only be done once.

Define β_i to be a back substitution function for Gaussian elimination to obtain d_i at any given iterative step. To emphasize that β_i depends only on the constant vector one can write

$$d_i^k = \beta_i [(r^k, a_i)] \tag{3.17}$$

where (r^k, a_i) is the i^{th} component of the constant vector at the k^{th} iterative step, and the coefficient matrix is that of theorem 3.1.

The following calculations are performed only once (fixed overhead costs). (See Fox (3, p. 176) for a description of the number of computations required for Gauss elimination.)

	Number of Additions	Number of Multiplications
For all combinations of i and j determine a_{ij} noting that $a_{ij} = a_{ji}$	$n^2(n + 1)/2$	$n^2(n + 1)/2$
Upper triangularizing w m by m coefficient matrices	$(\frac{m^3}{3} - \frac{m^2}{2} + \frac{m}{6})w$	$(\frac{m^3}{3} - \frac{m}{3})w$

After substituting $(n + m - 1)/m$ for w, the total number of fixed overhead operations to solve a system of n linear equations using an m-dimensional projection method is found to be bounded by

$$n^3 + n^2 + \frac{2m^3}{3} - \frac{7m^2}{6} + \frac{2m^2n}{3} + \frac{m}{3} - \frac{mn}{2} - \frac{n}{6} + \frac{1}{6} \quad . \quad (3.18)$$

Let $x_1, x_2, \ldots x_m$ be m arbitrary components of the solution vector modified by a given iterative step, then the following calculations are performed at each step in the order given:

	Number of Additions	Number of Multiplications
Calculations to determine the constant vector: $((r^k, a_1), (r^k, a_2) \ldots (r^k, a_m))$	mn	mn
Convert the constant vector as Gaussian elimination would do	$\frac{m^2 - m}{2}$	$\frac{m^2 - m}{2}$

Determine d_1^k, d_2^k,... d_m^k back substitution, i.e., evaluate β_i functions ($1 \leq i \leq m$)	$\frac{m^2 - m}{2}$	$\frac{m^2 + m}{2}$
Calculate x_1^{k+1}, x_2^{k+1},... x_m^{k+1} by Equation 3.14	m	0
Calculate r^{k+1} by Equation 3.15	mn	mn

The total number of operations performed at each step is

$$4mn + 2m^2 \quad . \tag{3.19}$$

Therefore, the number of operations required per component per iterative step to solve a system of n linear equations using an m-dimensional projection method is:

$$4n + 2m \quad . \tag{3.20}$$

The projection method presented in this chapter will be referred to in subsequent chapters as the old projection method or the conventional projection method.

IV. A NEW PROJECTION METHOD

A. A New One-Dimensional Projection Method

In solving a linear system of n equations using the old one-dimensional method (Equations 3.4 and 3.5) one obtains the following changes for the first cycle.

$$d_1^1 = (r^0, a_1)/a_{11}$$

$$d_2^1 = [(r^0, a_2) - d_1^1 a_{12}]/a_{22}$$

$$\vdots$$

$$d_n^1 = [(r^0, a_n) - d_1^1 a_{1,n} - d_2^1 a_{2,n} - \cdots d_{n-1}^1 a_{n-1,n}]/a_{nn} \quad . \tag{4.1}$$

Similarly for the second cycle

$$d_1^2 = [(r^0, a_1) - d_1^1 a_{11} - d_2^1 a_{12} \cdots - d_n^1 a_{1,n}]/a_{11}$$

$$d_2^2 = [(r^0, a_2) - \sum_{i=1}^{2} d_1^i a_{12} - d_2^1 a_{22} \cdots - d_n^1 a_{2,n}]/a_{22}$$

$$\vdots$$

$$d_n^2 = [(r^0, a_n) - \sum_{i=1}^{2} d_1^i a_{1,n} - \sum_{i=1}^{2} d_2^i a_{2,n} \cdots$$

$$- \sum_{i=1}^{2} d_{n-1}^i a_{n-1,n} - \sum_{i=1}^{1} d_n^i a_{nn}]/a_{nn} \quad . \tag{4.2}$$

Let

$$\tilde{d}_j^n = \sum_{i=1}^{n} d_j^i \tag{4.3}$$

then the changes to the approximate solution vector at the k^{th} cycle become

$$d_1^k = [(r^0, a_1) - \tilde{d}_1^{k-1} a_{11} - \tilde{d}_2^{k-1} a_{12} \cdots - \tilde{d}_n^{k-1} a_{1,n}]/a_{11}$$

$$d_2^k = [(r^0, a_2) - \tilde{d}_1^k a_{12} - \tilde{d}_2^{k-1} a_{22} - \tilde{d}_3^{k-1} a_{23} \cdots$$

$$- \tilde{d}_n^{k-1} a_{2,n}]/a_{22}$$

$$\vdots$$

$$d_n^k = [(r^0, a_n) - \tilde{d}_1^k a_{1,n} - \tilde{d}_2^k a_{2,n} \cdots - \tilde{d}_{n-1}^k a_{n-1,n}$$

$$- \tilde{d}_n^{k-1} a_{nn}]/a_{nn} \quad . \tag{4.4}$$

Since

$$d_i^k = [(r^0, a_i) - \tilde{d}_1^k a_{1,i} \cdots - \tilde{d}_{i-1}^k a_{i-1,i} - \tilde{d}_i^{k-1} a_{ii}$$

$$- \tilde{d}_{i+1}^{k-1} a_{i+1,i} \cdots - \tilde{d}_n^{k-1} a_{n,i}]/a_{ii}$$

$$= [(r^0, a_i) - \tilde{d}_1^k a_{1,i} \cdots - \tilde{d}_{i-1}^k a_{i-1,i} - \tilde{d}_{i+1}^{k-1} a_{i+1,i}$$

$$\cdots - \tilde{d}_n^{k-1} a_{n,i}]/a_{ii} - \tilde{d}_i^{k-1}$$

$$= \frac{(r^0, a_i)}{a_{ii}} - \tilde{d}_1^k \frac{a_{1,i}}{a_{ii}} \cdots - \tilde{d}_{i-1}^k \frac{a_{i-1,i}}{a_{ii}} - \tilde{d}_{i+1}^{k-1} \frac{a_{i+1,i}}{a_{ii}}$$

$$\cdots - \tilde{d}_n^{k-1} \frac{a_{n,i}}{a_{ii}} - \tilde{d}_i^{k-1} \quad . \tag{4.5}$$

And since

$$\tilde{d}_i^k = \tilde{d}_i^{k-1} + d_i^k \tag{4.6}$$

Equation 4.4 can be rewritten as

$$\tilde{d}_1^k = \frac{(r^0, a_1)}{a_{11}} - \tilde{d}_2^{k-1} \frac{a_{12}}{a_{11}} - \tilde{d}_3^{k-1} \frac{a_{13}}{a_{11}} \cdots - \tilde{d}_n^{k-1} \frac{a_{1,n}}{a_{11}}$$

$$\tilde{d}_2^k = \frac{(r^0, a_2)}{a_{22}} - \tilde{d}_1^k \frac{a_{12}}{a_{22}} - \tilde{d}_3^{k-1} \frac{a_{23}}{a_{22}} \cdots - \tilde{d}_n^{k-1} \frac{a_{2,n}}{a_{22}}$$

$$\vdots$$

$$\tilde{d}_n^k = \frac{(r^0, a_n)}{a_{nn}} - \tilde{d}_1^k \frac{a_{1,n}}{a_{nn}} - \tilde{d}_2^k \frac{a_{2,n}}{a_{nn}} \cdots \tilde{d}_{n-1}^k \frac{a_{n-1,n}}{a_{nn}} \quad . \tag{4.7}$$

The above proves the following lemma.

Lemma 4.1

The sum of changes to the i^{th} component of the approximate solution vector through k steps using a one-dimensional projection method in solving a linear system of n equations with a nonsingular coefficient matrix is given by

$$\tilde{d}_i^k = \frac{(r^0, a_i)}{a_{i,i}} - \tilde{d}_1^k \frac{a_{1,i}}{a_{i,i}} - \tilde{d}_2^k \frac{a_{2,i}}{a_{i,i}} - \ldots \tilde{d}_{i-1}^k \frac{a_{i-1,i}}{a_{i,i}}$$

$$- \tilde{d}_{i+1}^{k-1} \frac{a_{i+1,i}}{a_{i,i}} - \ldots \tilde{d}_n^{k-1} \frac{a_{n,i}}{a_{i,i}} \quad .$$

Therefore, by taking the old one-dimensional projection method and rearranging a few terms one developes a scheme where the change to a component of the approximate solution vector at a given step is a predetermined linear combination of the other n-1 components of the approximate solution vector. Equation 4.7 will be called the new one-dimensional projection method.

The iteration matrix for the new one-dimensional projection method can be obtained quite easily. Define

$$G_i = (r, a_i)/a_{ii} \tag{4.8}$$

$$H_{ij} = - a_{ij}/a_{ii} \tag{4.9}$$

then an iterative scheme for $\tilde{d}^k$ can be written in matrix notation as

$$L\tilde{d}^k = U\tilde{d}^{k-1} + G \tag{4.10}$$

where

$$L = \begin{bmatrix} 1 & 0 & 0 & \ldots & 0 & 0 \\ H_{21} & 1 & 0 & \ldots & 0 & 0 \\ H_{31} & H_{32} & 1 & \ldots & 0 & 0 \\ H_{41} & H_{42} & H_{43} & \ldots & 0 & 0 \\ \vdots & & & \ddots & & \\ H_{n-1,1} & H_{n-1,2} & H_{n-1,3} & \ldots & 1 & 0 \\ H_{n,1} & H_{n,2} & H_{n,3} & \ldots & H_{n,n-1} & 1 \end{bmatrix}$$

$$U = \begin{bmatrix} 0 & H_{12} & H_{13} & \cdots & H_{1,n} \\ 0 & 0 & H_{23} & \cdots & H_{2,n} \\ 0 & 0 & 0 & \cdots & H_{3,n} \\ \vdots & & & \ddots & \\ 0 & 0 & 0 & \cdots & H_{n-1,n} \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}$$

$$\tilde{d}^k = \begin{bmatrix} \tilde{d}_1^k \\ \tilde{d}_2^k \\ \vdots \\ \tilde{d}_n^k \end{bmatrix}, \quad \tilde{d}^{k-1} = \begin{bmatrix} \tilde{d}_1^{k-1} \\ \tilde{d}_2^{k-1} \\ \vdots \\ \tilde{d}_n^{k-1} \end{bmatrix}, \quad G = \begin{bmatrix} G_1 \\ G_2 \\ \vdots \\ G_n \end{bmatrix} \qquad (4.11)$$

Solving for $\tilde{d}^k$ in Equation 4.10 it is clear that

$$\tilde{d}^k = (L^{-1}U)\ \tilde{d}^{k-1} + L^{-1}G$$

and hence $L^{-1}U$ is the iteration matrix.

If one takes $x^0 \equiv 0$ then the above can be written as

$$x^k = (L^{-1}U)\ x^{k-1} + L^{-1}G$$

since x_i^k would equal $\tilde{d}_i^k$. Later in part C of Chapter V it

will be shown that the above substitution of x^k for $\tilde{d}^k$ can be done as long as $r^0 = b$ regardless of the choice for x^0, i.e., r^0 becomes b, x^0 is arbitrary and in general $r^0 \neq b - Ax^0$.

B. A New Two-Dimensional Projection Method

Define

$$t_{ij} = a_{ij}^2 - a_{ii}\, a_{jj} = t_{ji} \tag{4.12}$$

$$g_{ij} = [(r^0, a_j)a_{ij} - (r^0, a_i)a_{jj}]/t_{ij} \tag{4.13}$$

and assume that the column vectors of the coefficient matrix are arbitrarily paired (1,2), (3,4) ... (n-1,n) then using the old two-dimensional projection method (Equations 3.6 and 3.7) the changes for the first cycle are

$$d_1^1 = [(r^0, a_2)a_{12} - (r^0, a_1)a_{22}]/(a_{12}^2 - a_{11}a_{22}) = g_{12}$$

$$d_2^1 = g_{21}$$

$$d_3^1 = [((r^0, a_4) - d_1^1 a_{14} - d_2^1 a_{24})a_{34}$$

$$- ((r^0, a_3) - d_1^1 a_{13} - d_2^1 a_{23})a_{44}]/t_{34}$$

$$d_1^4 = [((r^0, a_3) - d_1^1 a_{13} - d_2^1 a_{23}) a_{34}$$

$$- ((r, a_4) - d_1^1 a_{14} - d_2^1 a_{24}) a_{33}]/t_{34}$$

$$\vdots$$

$$d_{n-1}^1 = [((r^0, a_n) - d_1^1 a_{1,n} - d_2^1 a_{2,n} \cdots$$

$$- d_{n-2}^1 a_{n-2,n}) a_{n-1,n} - ((r^0, a_{n-1}) - d_1^1 a_{1,n-1}$$

$$- d_2^1 a_{2,n-1} \cdots - d_{n-2}^1 a_{n-2,n-1}) a_{nn}]/t_{n-1,n}$$

$$d_n^1 = [((r^0, a_{n-1}) - d_1^1 a_{1,n-1} - d_2^1 a_{2,n-1} \cdots$$

$$- d_{n-2}^1 a_{n-2,n-1}) a_{n-1,n} - ((r^0, a_n) - d_1^1 a_{1,n}$$

$$- d_2^1 a_{2,n} \cdots - d_{n-2}^1 a_{n-2,n}) a_{n-1,n-1}]/t_{n-1,n}. \quad (4.14)$$

In general the changes to the approximate solution vector at the k^{th} cycle are

$$d_1^k = [((r^0, a_2) - \tilde{d}_1^{k-1} a_{12} - \tilde{d}_2^{k-1} a_{22} \cdots - \tilde{d}_n^{k-1} a_{2,n}) a_{12}$$

$$- ((r^0, a_1) - \tilde{d}_1^{k-1} a_{11} - \tilde{d}_2^{k-1} a_{12} \cdots$$

$$- d_n^{k-1} a_{1,n}) a_{22}]/t_{12}$$

$$d_2^k = [((r^0, a_1) - \tilde{d}_1^{k-1} a_{11} - \tilde{d}_2^{k-1} a_{12} \cdots - \tilde{d}_n^{k-1} a_{1,n}) a_{12}$$

$$- ((r^0, a_2) - \tilde{d}_1^{k-1} a_{12} - \tilde{d}_2^{k-1} a_{22} \cdots$$

$$- \tilde{d}_n^{k-1} a_{2,n}) a_{11}]/t_{12}$$

$$\vdots$$

$$d_{n-1}^k = [((r^0, a_n) - \tilde{d}_1^k a_{1,n} - \tilde{d}_2^k a_{2,n} \cdots - \tilde{d}_{n-2}^k a_{n-2,n}$$

$$- \tilde{d}_{n-1}^{k-1} a_{n-1,n} - \tilde{d}_n^{k-1} a_{nn}) a_{n-1,n}$$

$$- ((r^0, a_{n-1}) - \tilde{d}_1^k a_{1,n-1} - \tilde{d}_2^k a_{2,n-1} \cdots$$

$$- \tilde{d}_{n-2}^k a_{n-2,n-1} - \tilde{d}_{n-1}^{k-1} a_{n-1,n-1}$$

$$- \tilde{d}_n^{k-1} a_{n-1,n}) a_{nn}]/t_{n,n-1}$$

$$d_n^k = [((r^0, a_{n-1}) - \tilde{d}_1^k a_{1,n-1} - \tilde{d}_2^k a_{2,n-1} \cdots$$

$$- \tilde{d}_{n-2}^k a_{n-2,n-1} - \tilde{d}_{n-1}^{k-1} a_{n-1,n-1}$$

$$- \tilde{d}_n^{k-1} a_{n-1,n}) a_{n-1,n} - ((r^0, a_n) - \tilde{d}_1^k a_{1,n}$$

$$- \tilde{d}_2^k a_{2,n} \cdots - \tilde{d}_{n-2}^k a_{n-2,n} - \tilde{d}_{n-1}^{k-1} a_{n-1,n}$$

$$- \tilde{d}_n^{k-1} a_{nn}) a_{n-1,n-1}]/t_{n,n-1} \quad . \qquad (4.15)$$

No generality is lost if one assumed elements i and i+1 are paired then

$$d_i^k = [((r^0, a_{i+1}) - \tilde{d}_1^k a_{1,i+1} \cdots - \tilde{d}_{i-1}^k a_{i-1,i+1}$$

$$- \tilde{d}_i^{k-1} a_{i+1,i} - \tilde{d}_{i+1}^{k-1} a_{i+1,i+1} \cdots - \tilde{d}_n^{k-1} a_{i+1,n}) a_{i,i+1}$$

$$- ((r^o, a_i) - \tilde{d}_1^k a_{1,i} \cdots - \tilde{d}_{i-1}^k a_{i-1,i} - \tilde{d}_i^{k-1} a_{i,i}$$

$$- \tilde{d}_{i+1}^{k-1} a_{i,i+1} \cdots - \tilde{d}_n^{k-1} a_{i,n}) a_{i+1,i+1}]/t_{i,i+1} \qquad (4.16)$$

and rearranging terms

$$d_i^k = [(r^0, a_{i+1}) a_{i,i+1} - (r^0, a_i) a_{i+1,i+1}]/t_{i,i+1} \qquad (4.17a)$$

$$+ \tilde{d}_i^{k-1}[- a_{i,i+1}^2 + a_{ii} a_{i+1,i+1}]/t_{i,i+1} \qquad (4.17b)$$

$$+ \tilde{d}_{i+1}^{k-1}[- a_{i+1,i+1} a_{i,i+1} + a_{i+1,i+1} a_{i,i+1}]/t_{i,i+1} \qquad (4.17c)$$

$$+ [(- \tilde{d}_1^k a_{1,i+1} \cdots - \tilde{d}_{i-1}^k a_{i-1,i+1} - \tilde{d}_{i+2}^{k-1} a_{i+1,i+2}$$

$$\cdots - \tilde{d}_n^{k-1} a_{i+1,n}) a_{i,i+1} - (\tilde{d}_1^k a_{1,i} \cdots - \tilde{d}_{i-1}^k a_{i-1,i}$$

$$- \tilde{d}_{i+2}^{k-1} a_{i,i+2} \cdots - \tilde{d}_n^{k-1} a_{i,n}) a_{i+1,i+1}]/t_{i,i+1} \cdot \qquad (4.17d)$$

By substituting Equation 4.13 for line 4.17a and noting that line 4.17b = $-\tilde{d}_i^{k-1}$ and line 4.17c = 0 the following lemma is proved.

Lemma 4.2

$$d_i^k = g_{i,i+1} - \tilde{d}_i^{k-1} + \text{line } 4.17d \quad . \qquad (4.18)$$

Define

$$a_{ij}^{k} = a_{ik}a_{jj} - a_{jk}a_{ij} \tag{4.19a}$$

$$C_{ij}^{k} = a_{ij}^{k}/t_{ij} \tag{4.19b}$$

then substituting Equation 4.19 and applying Equation 4.6 to Equation 4.18 the final form for $\tilde{d}_i^k$ is obtained

$$\tilde{d}_i^k = g_{i,i+1} + \sum_{j=1}^{i-1} \tilde{d}_j^k \, C_{i,i+1}^j + \sum_{j=i+2}^{n} \tilde{d}_j^{k-1} \, C_{i,i+1}^j \tag{4.20a}$$

Similarly

$$\tilde{d}_{i+1}^k = g_{i+1,i} + \sum_{j=1}^{i-1} \tilde{d}_j^k \, C_{i+1,i}^j + \sum_{j=i+2}^{n} \tilde{d}_j^{k-1} \, C_{i+1,i}^j \tag{4.20b}$$

Therefore by taking the old two-dimensional projection method and rearranging a few terms a scheme is developed where the changes to two components of the approximate solution vector are a predetermined linear combination of the other n-2 components of the approximate solution vector. Equation 4.20 will be called the new two-dimensional projection method.

Following the same pattern as before with the new one-dimensional method with $x^0 \equiv 0$, the above method can be expressed in matrix notation as

$$L\, x^k = U\, x^{k-1} + G$$

where

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 \\ c_{34}^{1} & c_{34}^{2} & 1 & 0 & \cdots & 0 \\ c_{43}^{1} & c_{43}^{2} & 0 & 1 & \cdots & 0 \\ \vdots & & & & \ddots & \\ c_{n,n-1}^{1} & & & & \cdots & 1 \end{bmatrix}$$

$$U = \begin{bmatrix} 0 & 0 & c_{12}^{3} & c_{12}^{4} & \cdots & c_{12}^{n} \\ 0 & 0 & c_{21}^{3} & c_{21}^{4} & \cdots & c_{21}^{n} \\ 0 & 0 & 0 & 0 & \cdots & c_{34}^{n} \\ 0 & 0 & 0 & 0 & \cdots & c_{43}^{n} \\ \vdots & & & & \ddots & \\ 0 & 0 & 0 & 0 & \cdots & c_{n-2,n-3}^{n} \\ 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix}$$

$$x^k = \begin{bmatrix} x_1^k \\ x_2^k \\ \cdot \\ \cdot \\ \cdot \\ x_n^k \end{bmatrix}, \quad x^{k-1} = \begin{bmatrix} x_1^{k-1} \\ x_2^{k-1} \\ \cdot \\ \cdot \\ \cdot \\ x_n^{k-1} \end{bmatrix}, \quad G = \begin{bmatrix} g_{12} \\ g_{21} \\ g_{34} \\ g_{43} \\ \cdot \\ \cdot \\ \cdot \\ g_{n,n-1} \end{bmatrix} \quad .(4.21)$$

Thus

$$x^k = (L^{-1} U)x^{k-1} + L^{-1} G$$

and hence $L^{-1} U$ is the iteration matrix for the new two-dimensional projection method.

C. A New m-Dimensional Projection Method

A few observations concerning the β_i notation, Equation 3.17, are now in order.

Theorem 4.1

Let $x_1, x_2 \ldots x_m$ be m arbitrary components of the approximate solution vector to be modified during an iterative step of an m-dimensional projection method. Then for any $1 \leq i \leq m$ the following holds

$$\beta_i[(-\tilde{d}_1^{k-1}a_1 - \tilde{d}_2^{k-1}a_2 \ldots - \tilde{d}_i^{k-1}a_i \ldots - \tilde{d}_m^{k-1}a_m, a_i)]$$

$$= -\tilde{d}_i^{k-1} \quad .$$

Proof:

Applying the results of theorem 3.1 and Equation 3.17

$$\beta_i[(-\tilde{d}_1^{k-1}a_1 - \tilde{d}_2^{k-1}a_2 \ldots - \tilde{d}_i^{k-1}a_i \ldots - \tilde{d}_m^{k-1}a_m, a_i)]$$

(for i = 1 to m) is the solution vector, y, of the following system of m linear equations.

$$a_{11}y_1 + a_{12}y_2 + a_{13}y_3 + \cdots a_{1,n}y_n$$

$$= (-\tilde{d}_1^{k-1}a_1 - \tilde{d}_2^{k-1}a_2 \ldots - \tilde{d}_m^{k-1}a_m, a_1)$$

$$a_{21}y_1 + a_{22}y_2 + a_{23}y_3 + \cdots a_{2,n}y_n$$

$$= (-\tilde{d}_1^{k-1}a_1 - \tilde{d}_2^{k-1}a_2 \ldots - \tilde{d}_m^{k-1}a_m, a_2)$$

$$\vdots$$

$$a_{m,1}y_1 + a_{m,2}y_2 + a_{m,3}y_3 + \cdots a_{m,n}y_n$$

$$= (-\tilde{d}_1^{k-1}a_1 - \tilde{d}_2^{k-1}a_2 \ldots - \tilde{d}_m^{k-1}a_m, a_m) \quad .$$

The solution to the above system is

$$y_1 = -\tilde{d}_1^{k-1}$$

$$y_2 = -\tilde{d}_2^{k-1}$$

$$\vdots$$

$$y_m = -\tilde{d}_m^{k-1}$$

which is the desired result.

In order to develop the new m-dimensional projection method one needs to show that β_i is a distributive operator. The following theorem accomplishes this.

Theorem 4.2

Let $Cz = u$ represent an arbitrary linear system where C is a nonsingular coefficient matrix, z the solution vector and u the constant vector. Let p and q be column vectors such that $p+q = u$ hence $u_i = p_i + q_i$ and $Cz = p + q$. Let $\tilde{z}$ and $\overset{*}{z}$ be defined as follows

$$C\tilde{z} = p \quad , \qquad C\overset{*}{z} = q$$

then

$$C\tilde{z} + C\overset{*}{z} = p+q$$

and

$$C(\tilde{z} + \overset{*}{z}) = p+q$$

therefore $\tilde{z} + \overset{*}{z} = z$ and $\tilde{z}_i + \overset{*}{z}_i = z_i$.

By Equation 3.17 $z_i = \beta_i[u_i]$, $\tilde{z}_i = \beta_i[p_i]$ and $\overset{*}{z}_i = \beta_i[q_i]$ hence the desired result is achieved.

The change to the i^{th} component of the approximate solution vector at the k^{th} step for the old m-dimensional projection method is given by Equation 3.17. After substituting Equation 4.3 and generalizing Equation 3.7, Equation 3.17 can be rewritten as

$$d_i^k = \beta_i[(r^0 - \tilde{d}_1^{\bar{k}} a_1 - \tilde{d}_2^{\bar{k}} a_2 \ldots - \tilde{d}_i^{k-1} a_i - \ldots \tilde{d}_n^{\bar{k}} a_n, a_i)] \tag{4.22}$$

where $\bar{k}$ is k or k-1 depending if the element has been last changed in the k or k-1 cycle. Let $x_1, x_2 \ldots x_m$ be m arbitrary components of the approximate solution vector. Now an application of theorem 4.2 for $1 \leq i \leq m$ yields

$$d_i^k = \beta_i[(- \tilde{d}_1^{k-1} a_1 - \tilde{d}_2^{k-1} a_2 \ldots - \tilde{d}_m^{k-1} a_m, a_i)] + \beta_i[(r^0 - \tilde{d}_{m+1}^{k-1} a_{m+1} \ldots - \tilde{d}_n^{k-1} a_n, a_i)] \tag{4.23}$$

and applying theorem 4.1 one obtains

$$d_i^k = \beta_i[(r^0 - \tilde{d}_{m+1}^{k-1} a_{m+1} \ldots - \tilde{d}_n^{k-1} a_n, a_i)] - \tilde{d}_i^{k-1} \quad . \tag{4.24}$$

Finally, an application of Equation 4.6 yields this final form

$$\tilde{d}_i^k = \beta_i[(r^0 - \tilde{d}_{m+1}^{k-1} a_{m+1} \ldots - \tilde{d}_n^{k-1} a_n, a_i)] \quad . \qquad (4.25)$$

Let the new m-dimensional projection method be defined by Equation 4.25 then since Equation 4.25 is 3.17 with r^k replaced by an expression equal to it the new and old projection methods are equivalent.

V. THE NEW VERSUS OLD PROJECTION METHOD

A. Cost Analysis of the New m-Dimensional Projection Method

The steps involved in using the new m-dimensional projection method are summarized below.

(1) Choose by some criteria the dimension of the method to be used, m.

(2) Choose by some criteria the groupings of the columns of the coefficient matrix m at a time such that every column is in at least one group.

assumed $w = \lfloor \frac{m+n-1}{m} \rfloor$ such groups will be chosen since this is the minimum.

(3) The following fixed overhead operations are performed.

	Number of Additions	Number of Multiplications
(a) a_{ij} combinations	$n^2(n+1)/2$	$n^2(n+1)/2$
(b) Upper triangularize w m by m coefficient matrices	$[\frac{m^3}{3} - \frac{m^2}{2} + \frac{m}{6}]^w$	$[\frac{m^3}{3} - \frac{m}{3}]^w$
(c) (r^0, a_i)	n^2	n^2

The total number of fixed overhead operations (additions plus multiplications) is

$$n^3 + 3n^2 + \frac{2m^3}{3} - \frac{7m^2}{6} + \frac{2m^2n}{3} + \frac{m}{3} - \frac{mn}{2} - \frac{n}{6} + \frac{1}{6} \quad . \tag{5.1}$$

(4) At each iterative step the following calculations are performed.

	Number of Additions	Number of Multiplications
(a) Calculate the constant vector in Equation 4.25	$m(n-m)$	$m(n-m)$
(b) Convert the constant vector as Gaussian elimination would do	$(m^2-m)/2$	$(m^2-m)/2$
(c) Determine $x_1^k, x_2^k \ldots x_m^k$ by evaluating $\beta_1 \ldots \beta_m$	$(m^2-m)/2$	$(m^2+m)/2$

Thus, the total number of operations to be performed at each step is

$$2mn-m \tag{5.2}$$

which is

$$2n-1 \tag{5.3}$$

operations per component.

(5) The new projection methods bypass the use of the residual vector, thus termination of the iterative process is done by looking at successive approximate solution vectors after each cycle. If every corresponding component of two successive approximate solution vectors is within a predetermined

tolerance then the iterative process terminates. The cost is insignificant when done once a cycle. As in the old projection method one could, however, calculate (r^k, r^k) once a cycle to see if it is less than some tolerance, but the total cost in operations is $2n^2 + 3n$, which is defeating since it is more than the cost of the cycle itself!

In the preceding analysis it was assumed that w was chosen to minimize the number of groups of size m among n columns. This minimizes the fixed overhead operational costs and also the number of iterations per cycle. One can always increase w, the number of iterations per cycle, but one should be assured by some criteria that the added number of computations involved is offset by a faster rate of convergence. It was also assumed that m is a fixed integer but this need not be the case. By some criteria one may determine that the rate of convergence may be faster if some of the columns are grouped in pairs, some in triples, sextets or whatever. The groupings must be chosen on an a priori basis and once chosen cannot change from cycle to cycle.

B. The New Projection Method is Superior to the Old Projection Method

The definition of superior is indicated in the following theorem.

Theorem 5.1

For any system of n linear algebraic equations with a nonsingular coefficient matrix and for any dimensional projection method, m, and for any initial starting vector, x^0, the new projection method will obtain the solution in less number of arithmetic operations than the old projection method.

Proof:

From Equations 5.1 and 3.18 the following is obtained

	New Projection Method	Old Projection Method
Fixed overhead operations	$n^3+3n^2+2m^3/3$	$n^3+n^2+2m^3/3$
	$-7m^2/6+2m^2n/3+m/3$	$-7m^2/6+2m^2n/3+m/3$
	$-mn/2-n/6+1/6$	$-mn/2-n/6+1/6$
Step operations	$2mn-m$	$4mn+2m^2$

Let z be the step number at which point the new and the old projection methods cost the same (i.e., the breakeven point). z is determined by the following formula:

Fixed overhead costs for old projection + z [step costs for old projection] = Fixed overhead costs of the new projection + z [step costs for the new projection] .

Solving for z one obtains

$$z = 2n^2/(2m^2 + 2mn + m) \quad . \tag{5.4}$$

Finally, substituting (m+n-1)/m for w one obtains the following relationships:

$$2mn^2 < 2mn^2 + 2m^2n + mn$$

$$\frac{2n^2}{2m^2 + 2mn + m} < \frac{n}{m} < w \quad .$$

Therefore z < w and thus the breakeven point is somewhere in the first cycle. Since the iterative process must go at least one cycle the new projection method will always take less arithmetic operations than the old projection method to obtain the solution.

A few observations should be mentioned at this point. The new projection method solves for the new components of the approximate solution vector directly rather than for the changes to the component at each step. The step costs of the new method are less than half that of the old projection method and this is independent of the dimension of the method.

Corollary 5.1

The rate of convergence of the new projection method does not depend on the number of arithmetic operations per step since the number of operations performed per cycle is the same for any m. An exception to this is when n is not an even multiple of m. For example, let n = 10 then for m = 2, w = 5 and each component is changed once per cycle. For n = 10 with m = 3, w is 4 and two of the components are changed twice per cycle. Hence more operations are performed per cycle with m = 3 than m = 2. If one allows m to vary in the last step of the cycle in order that no component be changed twice in the same cycle then the number of operations performed per cycle is independent of m.

Proof:

The proof follows immediately since Equation 5.3 is independent of m.

Note that the additional overhead calculations incurred for the new method, $2n^2$, is also independent of the method used.

C. Round-Off Error Analysis

One advantage in general of an iterative method over a direct method is that round-off errors of an iterative method are not accumulative. This is certainly true of the

old projection method. At each step the changes to the approximate solution vector are derived based on the condition that the residual vector obtained will be orthogonal to the appropriate column vectors of A. After each step a new residual vector is calculated. In this way the inaccuracy of the approximate solution vector obtained in any one step has no effect on the final solution. The approximate solution vector is simply used as a guessed value for the next step. Other than this, each step is independent from the others. The computations at each step of the new projection method are also derived based on the condition that the residual vector obtained will be orthogonal to the appropriate column vectors of A. However, the residual vector is not directly calculated after each step as in Equation 3.2, but indirectly as in Equation 3.15. Hence the residual vector at each step is only as good as the previous one. This makes each iterative step dependent on the previous one, and perhaps rounding errors could accumulate to a point where the residual vector is not orthogonal to the appropriate column vectors of A. Thus errors may propagate and ruin the convergence.

The easiest way to prevent the error from propagating is to treat each step independently from the others, i.e., directly calculate the residual vector after each step.

Before one determines how this should be done, a close look must be given to how x^0 and r^0 are determined.

Equation 4.25 defines the new m-dimensional projection method. When this process converges one must add the resulting vector of accumulated changes, $\tilde{d}$ to x^0 to obtain the solution. Let $x_1, x_2 \ldots x_m$ be m arbitrary components of the approximate solution vector to be modified at a given iteration step then there are two ways to choose x^0.

(1) Choose $x^0 \equiv 0$ then r^0 becomes b and the resulting vector of changes, $\tilde{d}$, becomes the solution itself, x. In this case one can depict Equation 4.25 for $1 \leq i \leq m$ as

$$x_i^k = \beta_i[(r^0 - x_{m+1}^k a_{m+1} - x_{m+2}^k a_{m+2} \ldots - x_n^k a_n, a_i)] \quad (5.5)$$

since the vector of total changes, $\tilde{d}^k$ is the approximate solution vector, x^k.

(2) Choose $x^0 \neq 0$ then since no changes to the approximate solution vector have been calculated Equation 4.25 when $k = 1$ becomes

$$d_i^1 = \beta_i[(r^0, a_i)] \quad (5.6)$$

where

$$r^0 = b - Ax^0 \quad .$$

Assume that x^0 is some nonzero vector of changes resulting from applying a new projection

method for k steps from an initial vector of zero. In this way Equation 5.6 using some $x^0 \neq 0$ is equivalent to Equations 4.25 and 5.5 using an initial vector of zero for some iterative process. Thus, regardless of the choice of x^0, Equation 5.5 defines the new m-dimensional projection method. Furthermore, regardless of x^0, r^0 can be chosen as if x^0 were identically zero, i.e., $r^0 = b$.

This opens the door for development of initial vector algorithms since a savings of $2n^2+n$ calculations have already occurred not having to calculate r^0 when $x^0 \neq 0$. Two such algorithms are presented below.

Initial vector algorithm I

Lemma 5.1

Let $x = \alpha \vec{1}$ be a constant vector where $x_i = \alpha$. Define

$$y_i = \sum_{j=1}^{n} A_{ij} \quad \text{(the row sum of A)}$$

and let $r = b - Ax$ then the value of α to minimize (r,r) is given by $(b,y)/(y,y)$ where b is the constant vector of the linear system, $Ax = b$, and $(y,y) \neq 0$.

Proof:

$$(r,r) = (b - Ax,\ b - Ax)$$
$$= (b - \alpha y,\ b - \alpha y)$$

$$= (b,b) - 2\alpha(b,y) + \alpha^2(y,y)$$

taking the first derivative of both sides with respect to α and equating to zero one obtains

$$-2(b,y) + 2\alpha(y,y) = 0$$

hence

$$\alpha = (b,y)/(y,y).$$

Choose $x^0 = \alpha\vec{1}$ where α is defined in lemma 5.1. The total number of calculations required, i.e., for y, (b,y), (y,y) and $r^0 = b - \alpha y$ is $n^2 + 6n$.

Initial vector algorithm II

For each i let $\bar{x}^0 = \alpha_i \vec{1}$ such that d_i^1 using a one-dimensional projection method is zero, i.e., $d_i^1 = (\bar{r}^0, a_i)/(a_i, a_i) = 0$ where $\bar{r}^0 = b - A\bar{x}^0$. Call α_i the Initial Equilibrium Number for x_i.

Theorem 5.2

The Initial Equilibrium Number for x_i is given by $\alpha_i = (b, a_i)/(a_i, y)$ where y is as in lemma 5.1, the row sum of A and $(a_i, y) \neq 0$.

Proof:

$$d_i^1 = (\bar{r}^0, a_i)/(a_i, a_i) \text{ which implies } (\bar{r}^0, a_i) = 0$$

$$(\bar{r}^0, a_i) = (b - A\bar{x}^0, a_i)$$

$$= (b - \alpha_i y, a_i)$$

$$= (b, a_i) - \alpha_i (y, a_i)$$

solving for α_i where $(\bar{r}^0, a_i)$ is 0 yields the desired result. Define the Equilibrium Vector Point as $(\alpha_1, \alpha_2, \ldots, \alpha_n)$ where α_i is the Initial Equilibrium Number for x_i. Let this vector be the initial vector, x^0. The total number of calculations required, i.e., for y, (b, a_i), (y, a_i) and α_i is $5n^2 + n$. If rounding error propagation correction is employed, then only $3n^2 + n$ calculations are required since (b, a_i) would already have been calculated. The success of using initial vector algorithms is reflected in Chapter VII.

Returning to the problem of rounding error propagation, assume that one wishes to recalculate the residual vector after k steps of the iterative process defined by Equation 5.5 and then continue processing. To do this the following steps are performed.

(1) Add x^k to a vector, y. (y is initially zero.)

(2) Calculate (r^k, a_i) for all i. This becomes the

new (r^0, a_i). This is calculated by using the following identity:

$$(r^k, a_i) = (b - Ay, a_i)$$

$$= (b, a_i) - y_1 a_{1,i} - y_2 a_{2,i} \cdots - y_n a_{n,i} \quad .(5.7)$$

(3) The iterative process is continued but not using Equation 5.5. Instead Equation 4.25 defines the iterative process because now $\tilde{d}^k \neq x^k$.

One can recalculate a new residual vector as often as desired using the above steps. When the process converges, one must add the resulting vector of changes, $\tilde{d}$, to the accumulated y vector in order to obtain the solution.

A summary of the number of calculations involved to correct for rounding error propagation is given below.

	Number of Additions	Number of Multiplications
Additional overhead calculations required		
(b, a_i)	n^2	n^2
Calculations performed for the next cycle		
(a) Add $\tilde{d}^k$ to y	n	0
(b) Calculate (r^k, a_i) by Equation 5.7	n^2	n^2

Calculations performed at each step of the next cycle		
(a) Calculate the constant vector in Equation 4.25 (the number of computations varies from 0 to m(n-m) from step to step. The average number is given).	$m(n-m)/2$	$m(n-m)/2$
(b) Convert the constant vector as Gaussian elimination would do.	$(m^2-m)/2$	$(m^2-m)/2$
(c) Determine the next change vector by evaluating $\beta_1 \cdots \beta_m$	$(m^2-m)/2$	$(m^2+m)/2$

Thus the average number of operations to be performed at each step of the next cycle after correcting for rounding error propagation is $3nm+m^2$ which is $3n+m$ operations per component. Thus to correct for rounding error propagation using the new projection method one must endure an additional overhead cost of $2n^2$ calculations. Each component per step for rounding error correction cycles will require $3n+m$ calculations while uncorrected cycles will require the usual $2n-1$ calculations. Even if rounding error corrections are performed every cycle, this still requires significantly less number of calculations than the old projection method.

Recall the number of calculations for the old method per component per step was $4n+2m$.

VI. HYBRID PROJECTION METHODS

This section deals with an alternative approach for the development of an m-dimensional projection method by using two or more lower dimensional new projection methods as a basis. In all cases the sum of the lower dimensions used must add to m. Methods based on two lower dimensional methods are called simple hybrid projection methods and those based on more than two are called multiple hybrid projection methods.

A. A Four-Dimensional Simple Hybrid Projection Method

This method is developed by using two new two-dimensional projection methods. For convenience Equation 4.20 which defines the new two-dimensional projection method is repeated below

$$x_i^k = g_{i,i+1} + \sum_{j=1}^{i-1} x_j^k \, C_{i,i+1}^j + \sum_{j=i+2}^{n} x_j^{k-1} \, C_{i,i+1}^j \qquad (6.1a)$$

$$x_{i+1}^k = g_{i+1,i} + \sum_{j=1}^{i-1} x_j^k \, C_{i+1,i}^j + \sum_{j=i+2}^{n} x_j^{k-1} \, C_{i+1,i}^j \qquad (6.1b)$$

where columns i and i+1 of the coefficient matrix are paired together. In order for Equation 4.20 and Equations 6.1 to be equivalent it is necessary for $x_i^k = \tilde{d}_i^k$. Thus as in the new projection method, hybrid projection methods

require $r^0 = b$. There is still no restriction on x^0.

Let columns (1,2) and (3,4) be paired together, then at the k^{th} step Equations 6.1 yield

$$x_1^k = g_{12} + x_3^{k-1} C_{12}^3 + x_4^{k-1} C_{12}^4 + \sum_{j=5}^{n} x_j^{k-1} C_{12}^j$$

$$x_2^k = g_{21} + x_3^{k-1} C_{21}^3 + x_4^{k-1} C_{21}^4 + \sum_{j=5}^{n} x_j^{k-1} C_{21}^j$$

$$x_3^k = g_{34} + x_1^k C_{34}^1 + x_2^k C_{34}^2 + \sum_{j=5}^{n} x_j^{k-1} C_{34}^j$$

$$x_4^1 = g_{43} + x_1^k C_{43}^1 + x_2^k C_{43}^2 + \sum_{j=5}^{n} x_j^{k-1} C_{43}^j \quad . \qquad (6.2)$$

Define

$$Z_1^{k-1} = \sum_{j=5}^{n} x_j^{k-1} C_{12}^j$$

$$Z_2^{k-1} = \sum_{j=5}^{n} x_j^{k-1} C_{21}^j$$

$$Z_3^{k-1} = \sum_{j=5}^{n} x_j^{k-1} C_{34}^j$$

$$Z_4^{k-1} = \sum_{j=5}^{n} x_j^{k-1} C_{43}^j \qquad (6.3)$$

$$f_1(x_3^k, x_4^k) = g_{12} + x_3^k\, c_{12}^3 + x_4^k\, c_{12}^4$$

$$f_2(x_3^k, x_4^k) = g_{21} + x_3^k\, c_{21}^3 + x_4^k\, c_{21}^4$$

$$f_3(x_1^k, x_2^k) = g_{34} + x_1^k\, c_{34}^1 + x_2^k\, c_{34}^2$$

$$f_4(x_1^k, x_2^k) = g_{43} + x_1^k\, c_{43}^1 + x_2^k\, c_{43}^2 \quad . \qquad (6.4)$$

Then substituting Equations 6.3 and 6.4 in Equations 6.2 one obtains the following condensed form for the new two-dimensional projection method.

$$x_1^k = f_1(x_3^{k-1}, x_4^{k-1}) + z_1^{k-1}$$

$$x_2^k = f_2(x_3^{k-1}, x_4^{k-1}) + z_2^{k-1}$$

$$x_3^k = f_3(x_1^k, x_2^k) + z_3^{k-1}$$

$$x_4^k = f_4(x_1^k, x_2^k) + z_4^{k-1} \quad . \qquad (6.5a)$$

Repeating the projections on pairs (1,2), (3,4), (1,2), (3,4) ... etc. p times one obtains

$$x_1^{k+p} = f_1(x_3^{k+p-1}, x_4^{k+p-1}) + z_1^{k-1}$$

$$= f_1(f_3(x_1^{k+p-1}, x_2^{k+p-1}) + z_3^{k-1},\ f_4(x_1^{k+p-1}, x_2^{k+p-1}) + z_4^{k-1})$$

$$+ z_1^{k-1}$$

.
.
.

$$x_4^{k+p} = f_4(x_1^{k+p}, x_2^{k+p}) + z_4^{k-1}$$

$$= f_4(f_1(x_3^{k+p-1}, x_4^{k+p-1}) + z_1^{k-1},\ f_2(x_3^{k+p-1}, x_4^{k+p-1}) + z_2^{k-1})$$

$$+ z_4^{k-1}$$

which expanded is

$$x_1^{k+p} = f_1(g_{34} + x_1^{k+p-1}c_{34}^1 + x_2^{k+p-1}c_{34}^2 + z_3^{k-1},\ g_{43}$$

$$+ x_1^{k+p-1}c_{43}^1 + x_2^{k+p-1}c_{43}^2 + z_4^{k-1}) + z_1^{k-1}$$

$$= g_{12} + c_{12}^3[g_{34} + x_1^{k+p-1}c_{34}^1 + x_2^{k+p-1}c_{34}^2 + z_3^{k-1}]$$

$$+ c_{12}^4[g_{43} + x_1^{k+p-1}c_{43}^1 + x_2^{k+p-1}c_{43}^2 + z_4^{k-1}] + z_1^{k-1}$$

$$x_1^{k+p} = g_{12} + x_1^{k+p-1}[c_{12}^3c_{34}^1 + c_{12}^4c_{43}^1] + x_2^{k+p-1}[c_{12}^3c_{34}^2$$

$$+ c_{43}^2c_{12}^4] + c_{12}^3g_{34} + c_{12}^4g_{43} + c_{12}^3z_3^{k-1} + c_{12}^4z_4^{k-1} + z_1^{k-1}$$

.
.
.

$$x_4^{k+p} = f_4(g_{12} + x_3^{k+p-1}C_{12}^3 + x_4^{k+p-1}C_{12}^4 + z_1^{k-1},\ g_{21}$$

$$+ x_3^{k+p-1}C_{21}^3 + x_4^{k+p-1}C_{21}^4 + z_2^{k-1}) + z_4^{k-1}$$

$$= g_{43} + C_{43}^1[g_{12} + x_3^{k+p-1}C_{12}^3 + x_4^{k+p-1}C_{12}^4 + z_1^{k-1}]$$

$$+ C_{43}^2[g_{21} + x_3^{k+p-1}C_{21}^3 + x_4^{k+p-1}C_{21}^4 + z_2^{k-1}] + z_4^{k-1}$$

$$x_4^{k+p} = g_{43} + x_3^{k+p-1}[C_{43}^1C_{12}^3 + C_{43}^2C_{21}^3] + x_4^{k+p-1}[C_{43}^1C_{12}^4$$

$$+ C_{43}^2C_{21}^4] + C_{43}^1g_{12} + C_{43}^1z_1^{k-1} + C_{43}^2g_{21} + C_{43}^2z_2^{k-1} + z_4^{k-1}$$

(6.5b)

Define for the general quadruple i, j, u, v paired (i,j) (u,v)

$$P_i = C_{ij}^uC_{uv}^i + C_{ij}^vC_{vu}^i$$

$$P_j = C_{ji}^uC_{uv}^i + C_{ji}^vC_{vu}^i$$

$$P_u = C_{uv}^iC_{ij}^u + C_{uv}^jC_{ji}^u$$

$$P_v = C_{vu}^iC_{ij}^u + C_{vu}^jC_{ji}^u$$

$$Q_i = C_{ij}^uC_{uv}^j + C_{vu}^jC_{ij}^v$$

$$Q_j = C_{ji}^uC_{uv}^j + C_{ji}^vC_{vu}^j$$

$$Q_u = C_{uv}^iC_{ij}^v + C_{uv}^jC_{ji}^v$$

$$Q_v = C^i_{vu} C^v_{ij} + C^j_{vu} C^v_{ji}$$

$$R_i = C^u_{ij}[g_{uv} + Z_u] + C^v_{ij}[g_{vu} + Z_v] + Z_i + g_{ij}$$

$$R_j = C^u_{ji}[g_{uv} + Z_u] + C^v_{ji}[g_{vu} + Z_v] + Z_j + g_{ji}$$

$$R_u = C^i_{uv}[g_{ij} + Z_i] + C^j_{uv}[g_{ji} + Z_j] + Z_u + g_{uv}$$

$$R_v = C^i_{vu}[g_{ij} + Z_i] + C^j_{vu}[g_{ji} + Z_j] + Z_v + g_{vu} \quad . \qquad (6.5c)$$

Then Equations 6.5a become

$$x_1^{k+p} = P_1 x_1^{k+p-1} + Q_1 x_2^{k+p-1} + R_1$$

$$x_2^{k+p} = P_2 x_1^{k+p-1} + Q_2 x_2^{k+p-1} + R_2$$

$$x_3^{k+p} = P_3 x_3^{k+p-1} + Q_3 x_4^{k+p-1} + R_3$$

$$x_4^{k+p} = P_4 x_3^{k+p-1} + Q_4 x_4^{k+p-1} + R_4 \quad . \qquad (6.6)$$

The sequence of norms squared of the residual vectors from step to step is nonincreasing (18, p. 14), i.e.,

$$(r^k, r^k) - (r^{k+1}, r^{k+1}) \geq 0 \quad . \qquad (6.7)$$

There exists a lower limit, namely zero, for the norm squared of any vector hence as the number of iterative steps increases without limit the sequence of norms squared of the residual vectors will converge to a limit, q, i.e.,

$$\lim_{p \to \infty} (r^{k+p}, r^{k+p}) = q \quad . \tag{6.8}$$

Let x_1, x_2, x_3, x_4 be the values of the first four components of the approximate solution vector when $p \to \infty$ on Equation 6.6. Since the norm squared of the next residual vector can be reduced no more the approximate solution vector does not change either and Equation 6.6 in the limit becomes the following fixed point equations.

$$\begin{aligned} x_1 &= P_1 X_1 + Q_1 x_2 + R_1 \\ x_2 &= P_2 x_1 + Q_2 x_2 + R_2 \\ x_3 &= P_3 x_3 + Q_3 x_4 + R_3 \\ x_4 &= P_4 x_3 + Q_4 x_4 + R_4 \quad . \end{aligned} \tag{6.9}$$

Solving the above two 2 by 2 equations one obtains

$$x_1 = ([Q_2 - 1]R_1 - R_2Q_1)/(P_2Q_1 - [P_1 - 1][Q_2 - 1])$$

$$x_2 = (P_2R_1 - R_2[P_1 - 1])/([Q_2 - 1][P_1 - 1] - Q_1P_2)$$

$$x_3 = ([Q_4 - 1]R_3 - R_4Q_3)/(P_4Q_3 - [P_3 - 1][Q_4 - 1]$$

$$x_4 = (P_4R_3 - R_4[P_3 - 1])/([Q_4 - 1][P_3 - 1] - Q_3P_4)$$

and for the general quadruple i, j, u, v the final form becomes

$$x_i = ([Q_j - 1]R_i - R_jQ_i)/(P_jQ_i - [P_i - 1][Q_j - 1])$$

$$x_j = (P_jR_i - R_j[P_i - 1])/(Q_j - 1][P_i - 1] - Q_iP_j)$$

$$x_u = (Q_v - 1]R_u - R_vQ_u)/(P_vQ_u - [P_u - 1][Q_v - 1]$$

$$x_v = (P_vR_u - R_v[P_u - 1])/([Q_v - 1][P_u - 1] - Q_uP_v) \quad .$$

(6.10)

Thus by performing the new two-dimensional projection method cyclically on two pairs at a time a closed form for changing four components at a time is developed. Equation 6.10 is defined to be the simple four-dimensional hybrid projection method. For the four-dimensional simple hybrid projection method the fixed overhead and step costs are broken down as follows:

		Number of Additions	Number of Multiplications
Fixed Overhead Costs			
$n(n-2)$	c_{ij}^k	$n(n-2)$	$3n(n-2)$
n	g_{ij}	n	$3n$
$n/2$	t_{ij}	$n/2$	n
for all i	(r^0, a_i)	n^2	n^2
for all i,j	a_{ij}	$n^2(n+1)/2$	$n^2(n+1)/2$

up to n+3	P_i	n+3	2(n+3)
up to n+3	Q_i	n+3	2(n+3)

Step Costs

	Z_i, Z_j, Z_u, Z_v	4(n-4)	4(n-4)
	R_i, R_j, R_u, R_v	20	8
new	x_i, x_j, x_u, x_v	12	20

The total number of fixed overhead operations required is $n^3 + 7n^2 + 7n/2 + 18$ and the number of computations required per step is $8n + 28$ which is $2n + 7$ operation per component. The following questions immediately come to mind. Does the four-dimensional simple hybrid projection method resemble the new or old projection methods? What advantage or disadvantage does it possess over the other forms of projection methods? The former question is answered in this next theorem.

Theorem 6.1

Given the same initial approximate solution vector, x^0, the four-dimensional simple hybrid projection method, Equation 6.10 and the old four-dimensional projection method will generate the same sequence of approximate solution vectors, hence they are equivalent methods.

Proof:

Let x^k be the approximate solution vector at the k^{th} step and let components i,j,u,v be the four components to be altered at the k+1 step. Using the old four-dimensional projection method one determines unique values d_i^{k+1}, d_j^{k+1}, d_u^{k+1}, d_v^{k+1} by solving

$$(r^{k+1}, a_i) = 0$$

$$(r^{k+1}, a_j) = 0$$

$$(r^{k+1}, a_u) = 0$$

$$(r^{k+1}, a_v) = 0 \qquad (6.11)$$

which is the same as minimizing (r^{k+1}, r^{k+1}) (see theorem 3.1).

Let $\bar{r}^{k+1}$ be the residual vector and $\bar{d}_i^{k+1}$, $\bar{d}_j^{k+1}$, $\bar{d}_u^{k+1}$ and $\bar{d}_v^{k+1}$ be the changes to the four components at the k+1 step using the simple hybrid projection method. Note that $(\bar{r}^{k+1}, a_i) = (\bar{r}^{k+1}, a_j) = (\bar{r}^{k+1}, a_u) = (\bar{r}^{k+1}, a_v) = 0$. Define

$$d^{k+1} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ d_i^{k+1} \\ d_j^{k+1} \\ d_u^{k+1} \\ d_v^{k+1} \\ \vdots \\ 0 \end{bmatrix}, \quad \bar{d}^{k+1} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \bar{d}_i^{k+1} \\ \bar{d}_j^{k+1} \\ \bar{d}_u^{k+1} \\ \bar{d}_v^{k+1} \\ \vdots \\ 0 \end{bmatrix}, \quad d_q = d_q^{k+1} - \bar{d}_q^{k+1}$$

then

$$\begin{aligned} \bar{r}^{k+1} - r^{k+1} &= b - A(x^k + \bar{d}^{k+1}) - b + A(x^k + d^{k+1}) \\ &= A(d^{k+1} - \bar{d}^{k+1}) \\ &= a_i d_i + a_j d_j + a_u d_u + a_v d_v \quad . \end{aligned}$$

Taking the inner product of both sides of the above equation with each of a_i, a_j, a_u, a_v results in the following homogeneous system of equations.

$$0 = a_{ii} d_i + a_{ij} d_j + a_{iu} d_u + a_{iv} d_v$$

$$0 = d_{ji} d_i + a_{jj} d_j + a_{ju} d_u + a_{jv} d_v$$

$$0 = d_{ui} d_i + a_{uj} d_j + a_{uu} d_u + a_{uv} d_v$$

$$0 = d_{vi}d_i + a_{vj}d_j + a_{vu}d_u + a_{vv}d_v \quad .$$

The coefficient matrix is the same as that used in the old four-dimensional projection method (see theorem 3.1). According to Householder (8, p. 52) the coefficient matrix is nonsingular thus the unique solution is $d_i = d_j = d_u = d_v = 0$. Therefore the changes to the approximate solution vector at each step for the old four-dimensional projection method and the four-dimensional simple hybrid projection method are the same, which is the desired result.

B. An m-Dimensional Simple Hybrid Projection Method

The m-dimensional hybrid projection method is generated by a new p and a new q-dimensional projection method, with $(p+q = m)$. The new projection method is defined by Equation 4.25 which is simply a predetermined linear combination of the other components of the approximate solution vector that are not modified in the iterative step. Let the predetermined linear combination equivalent of Equation 4.25 be denoted as follows:

$$x_1^k = G_1 + \sum_{j=m+1}^{n} x_j^k W_{1,j}$$

$$x_2^k = G_2 + \sum_{j=m+1}^{n} x_j^k W_{2,j}$$

$$\vdots$$

$$x_m^k = G_m + \sum_{j=m+1}^{n} x_j^k W_{m,j} \qquad (6.12)$$

where columns 1,2,...m are grouped together and the values of G and W are predetermined by the β_i functions.

Let columns (1,2...p) and (p+1, p+2, p+q = m) be grouped together then at the k^{th} step Equation 6.12 yields

$$x_1^k = G_1 + x_{p+1}^{k-1} W_{1,p+1} \cdots + x_m^{k-1} W_{1,m} + \cdots x_n^{k-1} W_{1,n}$$

$$\vdots$$

$$x_p^k = G_p + x_{p+1}^{k-1} W_{p,p+1} \cdots + x_m^{k-1} W_{p,m} + \cdots x_n^{k-1} W_{p,n}$$

$$x_{p+1}^k = G_{p+1} + x_1^k W_{p+1,1} \cdots + x_p^k W_{p+1,p} + x_{m+1}^{k-1} W_{p+1,m+1}$$

$$\cdots + x_n^{k-1} W_{p+1,n}$$

$$\vdots$$

$$x_m^k = G_m + x_1^k W_{m,1} \cdots + x_p^k W_{m,p} + x_{m+1}^{k-1} W_{m,m+1} \cdots$$

$$+ x_n^{k-1} W_{m,n} \quad . \tag{6.13}$$

Define

$$z_i^{k-1} = \sum_{j=m+1}^{n} x_j^{k-1} W_{i,j} \tag{6.14}$$

$$f_i(x_{p+1}^k, \ldots x_m) = G_i + \sum_{j=p+1}^{m} x_j^k W_{i,j} \quad 1 \leq i \leq p \tag{6.15a}$$

$$f_i(x_1^k, \ldots x_p) = G_i + \sum_{j=1}^{p} x_j^k W_{i,j} \quad p+1 \leq i \leq m \quad . \tag{6.15b}$$

Then substituting Equations 6.15 and 6.14 in Equation 6.13 one obtains this condensed form of applying a new p followed by a new q-dimensional projection method at the k^{th} step

$$x_1^k = f_1(x_{p+1}^{k-1}, \ldots x_m^{k-1}) + z_1^{k-1}$$

$$\vdots$$

$$x_p^k = f_p(x_{p+1}^{k-1}, \ldots x_m^{k-1}) + z_p^{k-1}$$

$$x_{p+1}^{k} = f_{p+1}(x_1^k, \ldots x_p^k) + z_{p+1}^{k-1}$$

$$\vdots$$

$$x_m^k = f_m(x_1^k, \ldots x_p^k) + z_m^{k-1} \quad . \tag{6.16}$$

Repeating the projections on columns (1,2...p) (p+1,...m) cyclicly v times one obtains

$$x_1^{k+v} = f_1(x_{p+1}^{k+v-1}, \ldots x_m^{k+v-1}) + z_1^{k-1}$$

$$= f_1(f_{p+1}(x_1^{k+v-1}, \ldots x_p^{k+v-1}) + z_{p+1}^{k-1},$$

$$\ldots f_m(x_1^{k+v-1}, \ldots x_p^{k+v-1}) + z_m^{k-1}) + z_1^{k-1}$$

$$\vdots$$

$$x_p^{k+v} = f_p(f_{p+1}(x_1^{k+v-1}, \ldots x_p^{k+v-1}) + z_{p+1}^{k-1},$$

$$\ldots f_m(x_1^{k+v-1}, \ldots x_p^{k+v-1}) + z_m^{k-1}) + z_p^{k-1}$$

$$x_{p+1}^{k+v} = f_{p+1}(f_1(x_{p+1}^{k+v-1}, \ldots x_m^{k+v-1}) + z_1^{k-1},$$

$$\ldots f_p(x_{p+1}^{k+v-1},\ldots x_m^{k+v-1}) + z_p^{k-1}) + z_{p+1}^{k-1}$$

$$\vdots$$

$$x_m^{k+v} = f_m(f_1(x_{p+1}^{k+v-1},\ldots x_m^{k+v-1}) + z_1^{k-1},$$

$$\ldots f_p(x_{p+1}^{k+v-1},\ldots x_m^{k+v-1}) + z_p^{k-1}) + z_m^{k-1}$$

which expanded becomes

$$x_1^{k+v} = G_1 + W_{1,p+1}[f_{p+1}(x_1^{k+v-1},\ldots x_p^{k+v-1}) + z_{p+1}^{k-1}]$$

$$+ W_{1,p+2}[f_{p+2}(x_1^{k+v-1},\ldots x_p^{k+v-1}) + z_{p+2}^{k-1}]$$

$$\ldots + W_{1,m}[f_m(x_1^{k+v-1},\ldots x_p^{k+v-1}) + z_m^{k-1}] + z_1^{k-1}$$

$$= G_1 + z_1^{k-1}$$

$$+ W_{1,p+1}[G_{p+1} + x_1^{k+v-1}W_{p+1,1} \ldots + x_p^{k+v-1}W_{p+1,p}$$

$$+ z_{p+1}^{k-1}]$$

$$+ \ldots W_{1,m}[G_m + x_1^{k+v-1}W_{m,1} \ldots + x_p^{k+v-1}W_{m,p}$$

$$+ z_m^{k-1}]$$

$$\vdots$$

$$x_m^{k+v} = G_m + W_{m,1}[f_1(x_{p+1}^{k+v-1},\ldots x_m^{k+v-1}) + Z_1^{k-1}]$$

$$+ \ldots W_{m,p}[f_p(x_{p+1}^{k+v-1},\ldots x_m^{k+v-1}) + Z_p^{k-1}] + Z_m^{k-1}$$

$$= G_m + Z_m^{k-1}$$

$$+ W_{m,1}[G_1 + x_{p+1}^{k+v-1}W_{1,p+1} + \ldots x_m^{k+v-1}W_{1,m} + Z_1^{k-1}]$$

$$+ \ldots W_{m,p}[G_p + x_{p+1}^{k+v-1}W_{p,p+1} + \ldots x_m^{k+v-1}W_{p,m}$$

$$+ Z_p^{k-1}] \quad . \tag{6.17}$$

Define

$$Q_{i,j,u,v} = \sum_{q=u}^{v} W_{i,q}\, W_{q,j} \tag{6.18}$$

$$R_{i,j,u} = G_i + Z_i^{k-1} + \sum_{q=j}^{u} W_{i,q}\ [G_q + Z_q^{k-1}] \tag{6.19}$$

then Equations 6.17 become

$$x_1^{k+v} = Q_{1,1,p+1,m}x_1^{k+v-1} + Q_{1,2,p+1,m}x_2^{k+v-1}$$

$$+ \ldots Q_{1,p,p+1,m}x_p^{k+v-1} + R_{1,p+1,m}$$

$$\vdots$$

$$x_{p+1}^{k+v} = Q_{p+1,p+1,1,p}x_{p+1}^{k+v-1} + Q_{p+1,p+2,1,p}x_{p+2}^{k+v-1}$$

$$+ \ldots Q_{p+1,m,1,p} x_m^{k+v-1} + R_{p+1,1,p}$$

$$\vdots$$

$$x_m^{k+v} = Q_{m,p+1,1,p} x_{p+1}^{k+v-1} + Q_{m,p+2,1,p} x_{p+2}^{k+v-1}$$

$$+ \ldots Q_{m,m,1,p} x_m^{k+v-1} + R_{m,1,p} \quad . \qquad (6.20)$$

It was shown earlier in this chapter that as v increases without limit the above equations become the following fixed point equations.

$$x_1 = Q_{1,1,p+1,m} x_1 + Q_{1,2,p+1,m} x_2$$

$$+ \ldots Q_{1,p,p+1,m} x_p + R_{1,p+1,m}$$

$$\vdots$$

$$x_p = Q_{p,1,p+1,m} x_1 + Q_{p,2,p+1,m} x_2$$

$$+ \ldots Q_{p,p,p+1,m} x_p + R_{p,p+1,m}$$

$$x_{p+1} = Q_{p+1,p+1,1,p} x_{p+1} + Q_{p+1,p+2,1,p} x_{p+2}$$

$$+ \ldots Q_{p+1,m,1,p} x_m + R_{p+1,1,p}$$

$$\vdots$$

$$x_m = Q_{m,p+1,1,p}x_{p+1} + Q_{m,p+2,1,p}x_{p+2}$$

$$+ \ldots Q_{m,m,1,p}x_m + R_{m,1,p} \quad . \qquad (6.21)$$

Thus by performing the new p and new q-dimensional methods cyclicly on two groups of columns of A one obtains a closed form for changing m components at a single iteration step. This involves solving the above systems of p and q equations at each iterative step. Note that all Q values can be predetermined since W is predetermined. The only values that change from step to step are the constant vectors of the systems, R. Thus before the iterative process begins one can transform the two coefficient matrices of Equations 6.21 to an upper triangular form. This is done for each group of m columns that will be used in the iterative process. In this way the β_i functions (back substitution functions for Gaussian elimination) can be employed at each iterative step.

Theorem 6.1 can now be generalized to show that the m-dimensional simple hybrid projection method is equivalent to the new m-dimensional projection method.

Corollary 6.1

Given the same initial approximate solution vector, x^0, the m-dimensional simple hybrid projection method, Equation 6.21 and the old m-dimensional projection method

will generate the same sequence of approximate solution vectors, hence they are equivalent methods.

Proof:

A proof can be obtained by generalizing the proof given for theorem 6.1.

The advantage or disadvantage of the hybrid method still remains to be seen. To begin to determine this, one must first take a close look at the number of computations required for the simple hybrid projection method.

C. Cost Analysis of the Simple Hybrid Projection Methods

The steps involved in using the simple hybrid projection method are summarized below.

(1) Choose by some criteria the dimension of the method to be used--m.

(2) Choose by some criteria the groupings of the columns of the coefficient matrix m at a time such that every column is in at least one group. Also divide each group of m columns into two subgroups of p and q. Although it is not necessary, it is assumed p, q, m are constant in every group and that $w = \lfloor \frac{m+n-1}{m} \rfloor$ such groups of m are chosen since this is the minimum.

(3) The following fixed overhead operations are performed.

	Number of Additions	Number of Multiplications
(a) a_{ij} (all combinations)	$n^2(n+1)/2$	$n^2(n+1)/2$
(b) (r^0, a_i) for all i	n^2	n^2
(c) upper triangularize in preparation for determining G and W	$wp[\frac{1}{3}p^2 - \frac{1}{2}p + \frac{1}{6} + \frac{1}{2}(n-p+1)(p-1)] + wq[\frac{1}{3}q^2 - \frac{1}{2}q + \frac{1}{6} + \frac{1}{2}(n-q+1)(q-1)]$	$wp[(\frac{1}{3}p^2 - \frac{1}{3}) + \frac{1}{2}(n-p+1)(p-1)] + wq[(\frac{1}{3}q^2 - \frac{1}{3}) + \frac{1}{2}[(n-q+1)(q-1)]$
(d) back substitute to determine G and W values	$w\frac{p(p-1)}{2}(n-p+1) + w\frac{q(q-1)}{2}(n-q+1)$	$w\frac{p(p+1)}{2}(n-p+1) + w\frac{q(q+1)}{2}(n-q+1)$
(e) all Q values	$w[p^2q + qp^2]$	$w[p^2q + qp^2]$
(f) upper triangularize w systems of p and q equations (see Equation 6.17)	$w[\frac{p^3}{3} - \frac{p^2}{2} + \frac{p}{6} + \frac{q^3}{3} - \frac{q^2}{2} + \frac{q}{6}]$	$w[\frac{p^3}{3} - \frac{p}{3} + \frac{q^3}{3} - \frac{q}{3}]$

Substituting $w = (m+n-1)/m$ and recalling that $p+q = m$, the total number of fixed overhead operations is

$$[\frac{n+m-1}{m}][-\frac{2}{3}p^3 - \frac{2}{3}q^3 + 2p^2 + 2q^2 + 2p^2n + 2q^2n + 2p^2q + 2q^2p]$$

$$+ [n+m-1][-\frac{4}{3} - n] + n^3 + 3n^2 \quad . \tag{6.22}$$

(4) At each iterative step the following calculations are performed:

	Number of Additions	Number of Multiplications
(a) Z_i	$m(n-m)$	$m(n-m)$
(b) $Z_i + G_i$	m	0
(c) $R_{i,j,u}$	$p(q+1) + q(p+1)$	$2pq$
(d) convert the constant vector, R, as Gaussian elimination would do	$(p^2-p+q^2-q)/2$	$(p^2-p+q^2-q)/2$
(e) determine the m new approximations by back substitution	$(p^2-p+q^2-q)/2$	$(p^2+p+q^2+q)/2$

Thus the total number of operations performed at each step is

$$2mn - 2m^2 + m + 2p^2 + 2q^2 + 4pq$$

and substituting $p+q = m$ this becomes

$$2mn + m \tag{6.23}$$

which is

$$2n + 1 \tag{6.24}$$

operations per component per iterative step.

(5) The hybrid projection methods bypass the use of the residual vector. Thus termination of the iterative process is done the same as for the new projection method, i.e., by looking at successive approximate solution vectors after each cycle.

In the preceding analysis it was assumed that w was chosen to minimize the number of groups of size m among n columns. It was also assumed that m, p, q once chosen would remain fixed for all groups. As discussed for the new projection method in Chapter V this need not be the case. One can increase the number of iterations per cycle, w, if it is believed to increase the rate of convergence. One can vary as desired the group size, m, and p, q within each group. However, the groupings are chosen a priori and cannot change from cycle to cycle.

The following is a comparison of the arithmetic operations involved in the various projection methods (assume p, q, m constant).

Method	Number of Overhead Operations	Number of Step Operations Per Component
Old m	$n^3 + n^2 + 2m^3/3 - 7m^2/6$ $+ 2m^2n/3 + m/3 - mn/2 - n/6 + 1/6$	$4n + 2m$
New m	$n^3 + 3n^2 + 2m^3/3 - 7m^2/6$ $+ 2m^2n/3 + m/3 - mn/2 - n/6 + 1/6$	$2n - 1$

$$\text{Simple Hybrid}_{m} \quad \left[\frac{n+m-1}{m}\right]\left[-\frac{2}{3}(p^3+q^3) + 2(p^2+q^2) + 2n(p^2+q^2) + 2(p^2q+q^2p) - \frac{4m}{3} - mn\right] \qquad 2n+1$$

$$+ n^3 + 3n^2 \quad .$$

It is clear that the costs per component per iterative step is about the same for the simple hybrid and new projection methods. In fact the cost is independent of the choice for m in both methods and independent of p and q in the simple hybrid projection method. This means that like the new projection method the rate of convergence of the simple hybrid projection methods does not depend on the number of operations performed per cycle since it is the same for all methods. Thus the rate of convergence depends on the system itself, the dimension of the method used, m, along with p and q (which may vary from iterative step to step) and on what p and q-dimensional subspaces to project the residual vector onto at each step.

It is easy to show that the number of fixed overhead operations in Equation 6.22 is minimized when p = q. Hence, given m for a simple hybrid projection method one should choose p and q in the absence of any other criteria to be as equal as possible. Figures 1 and 2 compare the fixed overhead costs of the three types of projection methods.

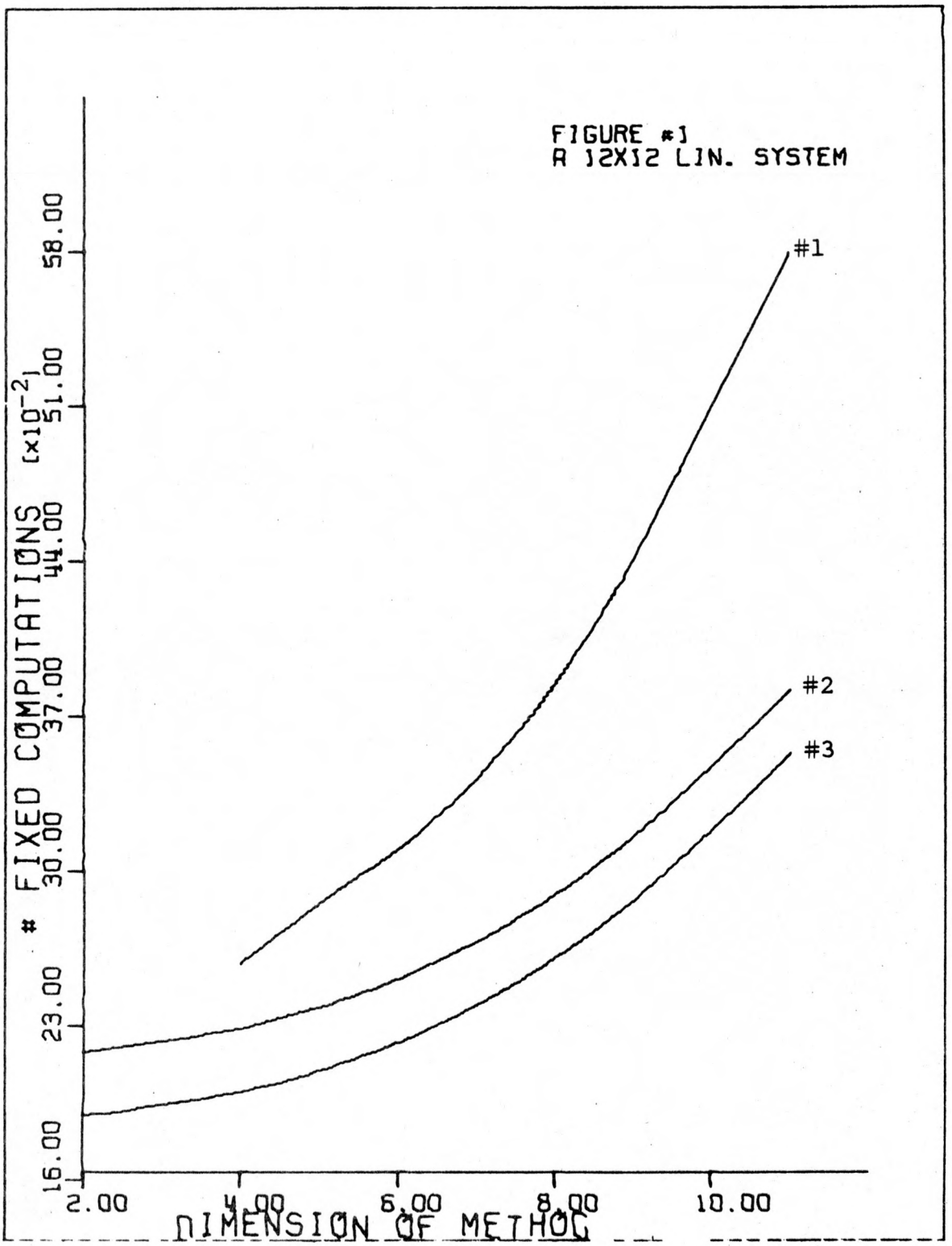

Figure 1. The number of fixed computations vs the dimension of the method used for all three types of projection methods on a 12 by 12 linear system

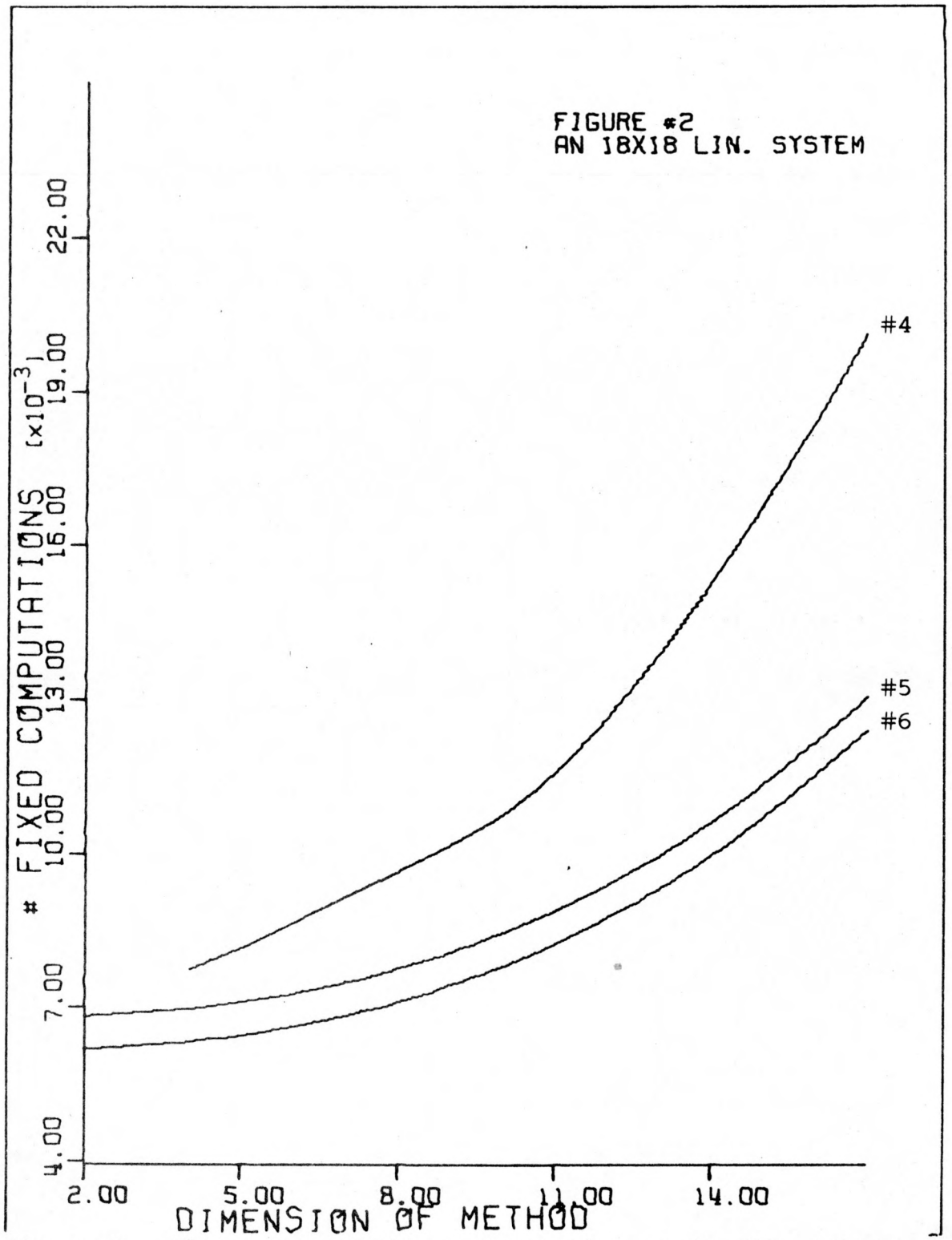

Figure 2. The number of fixed computations vs the dimension of the method used for all three types of projection methods on an 18 by 18 linear system

Figure 1 is for a 12 by 12 linear system and Figure 2 is for an 18 by 18 linear system. The similarity in the figures indicates that the same relative position of each curve would be the same regardless of what size linear system was used. Curves #1 and #4 depict the simple hybrid projection method (p=q), and curves #2 and #5 depict the new projection method. Curves #3 and #6 represent the old projection method. These figures indicate that the simple hybrid projection method becomes more costly at an increasing rate compared to the other two methods as the dimension of the method used increases. At the same time, the difference between the new and old projection methods decrease. This suggests that the simple hybrid projection methods are competitive when the dimension of the method used is small, i.e., according to the figures at most n/2.

It should be noted that the simple hybrid projection method like the new projection method is subject to rounding error propagation. One can easily prevent this by employing the same technique that was used for the new projection method (see Chapter V, part C).

One possible advantage of the simple hybrid projection method over the new projection method is the fact that different operations are performed in achieving the same result. Because of this it is possible for a given system that the simply hybrid projection method may not generate round-off errors to the extent that the new projection

method would or vice versa. It is suggested that a new projection method first be employed to solve a given system. If this should prove unsatisfactory a simple hybrid projection method should be attempted. Only after both methods have proved unsatisfactory due to rounding errors should the rounding error prevention technique be used. Another possible advantage may be in the selection of what m-dimensional subspaces to choose to project the residual vector onto at each iterative step. It may be easier to look at smaller dimensional subspaces p and q in order to obtain the best a priori groupings. One side effect of hybrid methods is its use in proving the following theorem.

Theorem 6.2

Let the iterative process for solving a linear system with a nonsingular coefficient matrix be at the k^{th} step using any of the projection methods. Assume one has a choice of using a p or q-dimensional projection method at the k+1 step ($p > q$). Let the columns of the coefficient matrix used by the q-dimensional method be a subset of those used by the p-dimensional method. Let (r^k, r^k), (r_p^{k+1}, r_p^{k+1}) and (r_q^{k+1}, r_q^{k+1}) represent the norms squared of the residual vector after the k^{th} step, k+1 step using a p-dimensional method and the k+1 step using a q-dimensional method respectively. The following relationship then holds:

$$(r_p^{k+1}, r_p^{k+1}) \leq (r_q^{k+1}, r_q^{k+1}) \leq (r^k, r^k).$$

Proof:

Since the projection method is nonnorm increasing from one step to the next (18, p. 14), it is clear that (r^k, r^k) is no less than either of the other two quantities. Let $w = p-q$. A generalization of theorem 6.1 indicates that a p-dimensional projection method is equivalent to a p-dimensional simple hybrid projection method using w and q-dimensional projection methods as a basis. The p-dimensional simple hybrid method is obtained by performing a q followed by a w-dimensional method cyclicly until the norm squared of the residual vector can be reduced no more. Since the norm is nonincreasing and the q-dimensional method is only one of the iterative steps involved in developing a p-dimensional method the desired result is true.

D. Multiple Hybrid Projection Methods

Thus far the development of an m-dimensional hybrid projection method has been based on two lower dimensional new projection methods. This section looks at the possibility of developing an m-dimensional multiple hybrid projection method based on three lower dimensional new projection methods, p, q, s where $p+q+s = m$.

The example below develops a six-dimensional multiple

hybrid projection method where p = q = s = 2. For convenience Equation 4.20 which defines the new two-dimensional projection method is repeated below.

$$x_i^k = g_{i,i+1} + \sum_{j=1}^{i-1} x_j^k C_{i,i+1}^j + \sum_{j=i+2}^{n} x_j^{k-1} C_{i,i+1}^j$$

$$x_{i+1}^k = g_{i+1,i} + \sum_{j=1}^{i-1} x_j^k C_{i+1,i}^j + \sum_{j=i+2}^{n} x_j^{k-1} C_{i+1,i}^j \qquad (6.25)$$

where columns i and i+1 are paired together.

Let columns (1,2), (3,4), (5,6) be grouped together then at the k^{th} cycle one has the following:

$$x_1^k = g_{12} + x_3^{k-1}C_{12}^3 + x_4^{k-1}C_{12}^4 + x_5^{k-1}C_{12}^5 + x_6^{k-1}C_{12}^6 + \sum_{j=7}^{n} x_j^{k-1}C_{12}^j$$

$$x_2^k = g_{21} + x_3^{k-1}C_{21}^3 + x_4^{k-1}C_{21}^4 + x_5^{k-1}C_{21}^5 + x_6^{k-1}C_{21}^6 + \sum_{j=7}^{n} x_j^{k-1}C_{21}^j$$

$$x_3^k = g_{34} + x_1^kC_{34}^1 + x_2^kC_{34}^2 + x_5^{k-1}C_{34}^5 + x_6^{k-1}C_{34}^6 + \sum_{j=7}^{n} x_j^{k-1}C_{34}^j$$

$$x_4^k = g_{43} + x_1^kC_{43}^1 + x_2^kC_{43}^2 + x_5^{k-1}C_{43}^5 + x_6^{k-1}C_{43}^6$$

$$+ \sum_{j=7}^{n} x_j^{k-1} C_{43}^j$$

$$x_5^k = g_{56} + x_1^k C_{56}^1 + x_2^k C_{56}^2 + x_3^k C_{56}^3 + x_4^k C_{56}^4 + \sum_{j=7}^{n} x_j^{k-1} C_{56}^j$$

$$x_6^k = g_{65} + x_1^k C_{65}^1 + x_2^k C_{65}^2 + x_3^k C_{65}^3 + x_4^k C_{65}^4 + \sum_{j=7}^{n} x_j^{k-1} C_{65}^j \tag{6.26}$$

Define

$$Z_{ip} = \sum_{j=7}^{n} x_j^{k-1} C_{ip}^j \tag{6.27}$$

$$f_{ij}(x_p^k, x_t^k, x_y^k, x_z^k) = g_{ij} + x_p^k C_{ij}^p + x_t^k C_{ij}^t + x_y^k C_{ij}^y + x_z^k C_{ij}^z \tag{6.28}$$

where (i,j), (p,t) and (y,z) are the three pairs forming the group of six columns.

Substituting Equations 6.27 and 6.28 into Equation 6.26 one obtains

$$x_1^k = f_{12}(x_3^{k-1}, x_4^{k-1}, x_5^{k-1}, x_6^{k-1}) + Z_{12}$$

$$x_2^k = f_{21}(x_3^{k-1}, x_4^{k-1}, x_5^{k-1}, x_6^{k-1}) + Z_{21}$$

$$x_3^k = f_{34}(x_1^k, x_2^k, x_5^{k-1}, x_6^{k-1}) + Z_{34}$$

$$x_4^k = f_{43}(x_1^k, x_2^k, x_5^{k-1}, x_6^{k-1}) + Z_{43}$$

$$x_5^k = f_{56}(x_1^k, x_2^k, x_3^k, x_4^k) + Z_{56}$$

$$x_6^k = f_{65}(x_1^k, x_2^k, x_3^k, x_4^k) + Z_{65} \quad . \tag{6.29}$$

Repeating iterations on pairs (1,2), (3,4), (5,6) cyclicly v times one obtains

$$x_1^{k+v} = f_{12}(x_3^{k+v-1}, x_4^{k+v-1}, x_5^{k+v-1}, x_6^{k+v-1}) + z_{12}$$

$$= f_{12}[f_{34}(x_1^{k+v-1}, x_2^{k+v-1}, x_5^{k+v-2}, x_6^{k+v-2}) + z_{34} ,$$

$$f_{43}(x_1^{k+v-1}, x_2^{k+v-1}, x_5^{k+v-2}, x_6^{k+v-2}) + z_{43} ,$$

$$f_{56}(x_1^{k+v-1}, x_2^{k+v-1}, x_3^{k+v-1}, x_4^{k+v-1}) + z_{56} ,$$

$$f_{65}(x_1^{k+v-1}, x_2^{k+v-1}, x_3^{k+v-1}, x_4^{k+v-1}) + z_{65}] + z_{12}$$

$$x_1^{k+v} = f_{12}[g_{34} + x_1^{k+v-1}C_{34}^1 + x_2^{k+v-1}C_{34}^2 + x_5^{k+v-2}C_{34}^5$$

$$+ x_6^{k+v-2}C_{34}^6 + z_{34} ,$$

$$g_{43} + x_1^{k+v-1}C_{43}^1 + x_2^{k+v-1}C_{43}^2 + x_5^{k+v-2}C_{43}^5$$

$$+ x_6^{k+v-2}C_{43}^6 + z_{43} ,$$

$$g_{56} + x_1^{k+v-1}C_{56}^1 + x_2^{k+v-1}C_{56}^2 + x_3^{k+v-1}C_{56}^3$$

$$+ x_4^{k+v-1}C_{56}^4 + z_{56} ,$$

$$g_{65} + x_1^{k+v-1}C_{65}^1 + x_2^{k+v-1}C_{65}^2 + x_3^{k+v-1}C_{65}^3$$

$$+ x_4^{k+v-1}C_{65}^4 + z_{65}] + z_{12}$$

$$x_1^{k+v} \equiv f_{12}[g_{34} + x_1^{k+v-1}C_{34}^1 + x_2^{k+v-1}C_{34}^2$$

$$+ C_{34}^5(g_{56} + x_1^{k+v-2}C_{56}^1 + x_2^{k+v-2}C_{56}^2$$

$$+ x_3^{k+v-2}C_{56}^3 + x_4^{k+v-2}C_{56}^4 + Z_{56})$$

$$+ C_{34}^6(g_{65} + x_1^{k+v-2}C_{65}^1 + x_2^{k+v-2}C_{65}^2$$

$$+ x_3^{k+v-2}C_{65}^3 + x_4^{k+v-2}C_{65}^4 + Z_{65}) + Z_{34} ,$$

$$g_{43} + x_1^{k+v-1}C_{43}^1 + x_2^{k+v-1}C_{43}^2$$

$$+ C_{43}^5(g_{56} + x_1^{k+v-2}C_{56}^1 + x_2^{k+v-2}C_{56}^2$$

$$+ x_3^{k+v-2}C_{56}^3 + x_4^{k+v-2}C_{56}^4 + Z_{56})$$

$$+ C_{43}^6(g_{65} + x_1^{k+v-2}C_{65}^1 + x_2^{k+v-2}C_{65}^2$$

$$+ x_3^{k+v-2}C_{65}^3 + x_4^{k+v-2}C_{65}^4 + Z_{65}) + Z_{43} ,$$

$$g_{56} + x_1^{k+v-1}C_{56}^1 + x_2^{k+v-1}C_{56}^2 + x_3^{k+v-1}C_{56}^3$$

$$+ x_4^{k+v-1}C_{56}^4 + Z_{56} ,$$

$$g_{65} + x_1^{k+v-1}C_{65}^1 + x_2^{k+v-1}C_{65}^2 + x_3^{k+v-1}C_{65}^3$$

$$+ x_4^{k+v-1}C_{65}^4 + Z_{65}] + Z_{12}$$

$$
\begin{aligned}
x_1^{k+v} = {} & g_{12} \\
& + C_{12}^{3}(g_{34} + x_1^{k+v-1}C_{34}^{1} + x_2^{k+v-1}C_{34}^{2}) \\
& + C_{12}^{3}C_{34}^{5}(g_{56} + x_1^{k+v-2}C_{56}^{1} + x_2^{k+v-2}C_{56}^{2} \\
& \qquad + x_3^{k+v-2}C_{56}^{3} + x_4^{k+v-2}C_{56}^{4} + z_{56}) \\
& + C_{12}^{3}C_{34}^{6}(g_{65} + x_1^{k+v-2}C_{65}^{1} + x_2^{k+v-2}C_{65}^{2} \\
& \qquad + x_3^{k+v-2}C_{65}^{3} + x_4^{k+v-2}C_{65}^{4} + z_{65}) \\
& + C_{12}^{3}z_{34} \\
& + C_{12}^{4}(g_{43} + x_1^{k+v-1}C_{43}^{1} + x_2^{k+v-1}C_{43}^{2}) \\
& + C_{12}^{4}C_{43}^{5}(g_{56} + x_1^{k+v-2}C_{56}^{1} + x_2^{k+v-2}C_{56}^{2} \\
& \qquad + x_3^{k+v-2}C_{56}^{3} + x_4^{k+v-2}C_{56}^{4} + z_{56}) \\
& + C_{12}^{4}C_{43}^{6}(g_{65} + x_1^{k+v-2}C_{65}^{1} + x_2^{k+v-2}C_{65}^{2} \\
& \qquad + x_3^{k+v-2}C_{65}^{3} + x_4^{k+v-2}C_{65}^{4} + z_{65}) \\
& + C_{12}^{4}z_{43}
\end{aligned}
$$

$$+ C_{12}^{5}(g_{56} + x_1^{k+v-1}C_{56}^{1} + x_2^{k+v-1}C_{56}^{2}$$

$$+ x_3^{k+v-1}C_{56}^{3} + x_4^{k+v-1}C_{56}^{4} + Z_{56})$$

$$+ C_{12}^{6}(g_{65} + x_1^{k+v-1}C_{65}^{1} + x_2^{k+v-1}C_{65}^{2}$$

$$+ x_3^{k+v-1}C_{65}^{3} + x_4^{k+v-1}C_{65}^{4} + Z_{65}) + Z_{12}.$$

For large v $x_i^{k+v-1} = x_i^{k+v-2}$ and thus after collecting terms one obtains the following form:

$$x_1^{k+v} = g_{12} + Z_{12} + C_{12}^{3}g_{34} + C_{12}^{4}g_{43} + C_{12}^{3}Z_{34} + C_{12}^{4}Z_{43}$$

$$+ (g_{56} + Z_{56})(C_{12}^{3}C_{34}^{5} + C_{12}^{4}C_{43}^{5} + C_{12}^{5})$$

$$+ (g_{65} + Z_{65})(C_{12}^{3}C_{34}^{6} + C_{12}^{4}C_{43}^{6} + C_{12}^{6})$$

$$+ x_1^{k+v-1}[C_{12}^{3}C_{34}^{1} + C_{12}^{3}C_{34}^{5}C_{56}^{1} + C_{12}^{3}C_{34}^{6}C_{65}^{1} + C_{12}^{4}C_{43}^{1}$$

$$+ C_{12}^{4}C_{43}^{5}C_{56}^{1} + C_{12}^{4}C_{43}^{6}C_{65}^{1} + C_{12}^{5}C_{56}^{1} + C_{12}^{6}C_{65}^{1}]$$

$$+ x_2^{k+v-1}[C_{12}^{3}C_{34}^{2} + C_{12}^{3}C_{34}^{5}C_{56}^{2} + C_{12}^{3}C_{34}^{6}C_{65}^{2} + C_{12}^{4}C_{43}^{2}$$

$$+ C_{12}^{4}C_{43}^{5}C_{56}^{2} + C_{12}^{4}C_{43}^{6}C_{65}^{2} + C_{12}^{5}C_{56}^{2} + C_{12}^{6}C_{65}^{2}]$$

$$+ x_3^{k+v-1}[C_{12}^{3}C_{34}^{5}C_{56}^{3} + C_{12}^{3}C_{34}^{6}C_{65}^{3} + C_{12}^{4}C_{43}^{5}C_{56}^{3}$$

$$+ C_{12}^{4}C_{43}^{6}C_{65}^{3} + C_{12}^{5}C_{56}^{3} + C_{12}^{6}C_{65}^{3}]$$

$$+ x_4^{k+v-1}[C_{12}^3 C_{34}^5 C_{56}^4 + C_{12}^3 C_{34}^6 C_{65}^4 + C_{12}^4 C_{43}^5 C_{56}^4$$

$$+ C_{12}^4 C_{43}^6 C_{65}^4 + C_{12}^5 C_{56}^4 + C_{12}^6 C_{65}^4] \qquad (6.30)$$

Now let Equation 6.30 be represented in the following form:

$$x_1^{k+v} = H_{11} x_1^{k+v-1} + H_{12} x_2^{k+v-1} + H_{13} x_3^{k+v-1} + H_{14} x_4^{k+v-1}$$

$$+ K_1 + T_1 \qquad (6.31)$$

where H_{11}, H_{12}, H_{13}, H_{14} represent the coefficients of x_1^{k+v-1}, x_2^{k+v-1}, x_3^{k+v-1}, x_4^{k+v-1} respectively in Equation 6.30. K_1 represents the constant terms and T_1 the terms involving Z.

A similar development which led to Equation 6.30 for x_1^{k+v-1} can be performed for

x_2^{k+v-1} with x_5^{k+v-1}, x_6^{k+v-1} terms omitted

x_3^{k+v-1} with x_5^{k+v-1}, x_6^{k+v-1} terms omitted

x_4^{k+v-1} with x_5^{k+v-1}, x_6^{k+v-1} terms omitted

x_3^{k+v-1} with x_1^{k+v-1}, x_2^{k+v-1} terms omitted

x_4^{k+v-1} with x_1^{k+v-1}, x_2^{k+v-1} terms omitted

x_5^{k+v-1} with x_1^{k+v-1}, x_2^{k+v-1} terms omitted

x_6^{k+v-1} with x_1^{k+v-1}, x_2^{k+v-1} terms omitted .

These eight equations can be represented in Equation 6.31 form as

$$x_1^{k+v} = H_{11}x_1^{k+v-1} + H_{12}x_2^{k+v-1} + H_{13}x_3^{k+v-1} + H_{14}x_4^{k+v-1}$$

$$+ K_1 + T_1$$

$$x_2^{k+v} = H_{21}x_1^{k+v-1} + H_{22}x_2^{k+v-1} + H_{23}x_3^{k+v-1} + H_{24}x_4^{k+v-1}$$

$$+ K_2 + T_2$$

$$x_3^{k+v} = H_{31}x_1^{k+v-1} + H_{32}x_2^{k+v-1} + H_{33}x_3^{k+v-1} + H_{34}x_4^{k+v-1}$$

$$+ K_3 + T_3$$

$$x_4^{k+v} = H_{41}x_1^{k+v-1} + H_{42}x_2^{k+v-1} + H_{43}x_3^{k+v-1} + H_{44}x_4^{k+v-1}$$

$$+ K_4 + T_4$$

$$x_5^{k+v} = H_{53}x_3^{k+v-1} + H_{54}x_4^{k+v-1} + H_{55}x_5^{k+v-1} + H_{56}x_6^{k+v-1}$$

$$+ K_5 + T_5$$

$$x_6^{k+v} = H_{63}x_3^{k+v-1} + H_{64}x_4^{k+v-1} + H_{65}x_5^{k+v-1} + H_{66}x_6^{k+v-1}$$

$$+ K_6 + T_6$$

$$x_3^{k+v} = H_{73}x_3^{k+v-1} + H_{74}x_4^{k+v-1} + H_{75}x_5^{k+v-1} + H_{76}x_6^{k+v-1}$$

$$+ K_7 + T_7$$

$$x_4^{k+v} = H_{83}x_3^{k+v-1} + H_{84}x_4^{k+v-1} + H_{85}x_5^{k+v-1} + H_{86}x_6^{k+v-1}$$

$$+ K_8 + T_8 \quad . \tag{6.32}$$

It has already been shown that if one repeats iterations on pairs (1,2), (3,4), (5,6) cyclicly v times and lets v approach infinity that the following two fixed point systems of four equations can be formed.

$$x_1 = H_{11}x_1 + H_{12}x_2 + H_{13}x_3 + H_{14}x_4 + K_1 + T_1$$

$$x_2 = H_{21}x_1 + H_{22}x_2 + H_{23}x_3 + H_{24}x_4 + K_2 + T_2$$

$$x_3 = H_{31}x_1 + H_{32}x_2 + H_{33}x_3 + H_{34}x_4 + K_3 + T_3$$

$$x_4 = H_{41}x_1 + H_{42}x_2 + H_{43}x_3 + H_{44}x_4 + K_4 + T_4$$

$$x_5 = H_{53}x_3 + H_{54}x_4 + H_{55}x_5 + H_{56}x_6 + K_5 + T_5$$

$$x_6 = H_{63}x_3 + H_{64}x_4 + H_{65}x_5 + H_{66}x_6 + K_6 + T_6$$

$$x_3 = H_{73}x_3 + H_{74}x_4 + H_{75}x_5 + H_{76}x_6 + K_7 + T_7$$

$$x_4 = H_{83}x_3 + H_{84}x_4 + H_{85}x_5 + H_{86}x_6 + K_8 + T_8$$

The H, K and T values have been explicitly defined for only x_1. The remaining definitions will not be given for it soon will become clear that this method is too costly to implement compared to the other projection methods.

For the above six-dimensional case from Equations 6.32 and 6.30 one can determine the amount of fixed overhead and step computations required without actually knowing what they are.

Assuming $w = \lfloor \frac{n+5}{6} \rfloor$ groups of six are formed then the fixed overhead costs are:

	Number of Additions	Number of Multiplications
a_{ij}	$n^2(n+1)/2$	$n^2(n+1)/2$
(r^0, a_i)	n^2	n^2
t_{ij}	$n/2$	n
C_{ij}^k	$n(n-2)$	$3n(n-2)$
g_{ij}	n	$3n$
H	$8w(24)$	$8w(44)$
K	$8w(6)$	$8w(6)$
upper triangularizing $2w$ 4 by 4 coefficient matrices	$2w(14)$	$2w(20)$

the total number of overhead operations after substituting for w is

$$n^3 + 7n^2 + 121n + 616 \tag{6.33}$$

At each iterative step the following calculations are performed.

	Number of Additions	Number of Multiplications
Z	6(n-6)	6(n-6)
calculate the constant vectors K + T	48	32
convert the constant vectors as Gaussian elimination would do	12	12
determine $x_1^k, \ldots x_6^k$	12	20

The total number of operations performed at each step is

$$12n + 64$$

which is

$$2n + 11 \tag{6.34}$$

operations per component.

Generalizing all of the above one concludes that multiple hybrid projection methods involve more computations than simple hybrid methods. This is because two systems of 2m/3 equations need to be solved at each step compared to two systems of m/2 equations for the simple hybrid methods. Because of this, the total overhead

costs are more than any of the other projection methods. Furthermore, the operational cost per component is dependent on the dimension of the method used and is more than either the new or simple hybrid projection methods. In conclusion, the simple hybrid projection method is competitive with the new projection method and there are an unlimited number of possible multiple hybrid projection methods that exist but their advantage if any remains to be seen.

VII. TEST PROBLEMS AND COMPARISONS

The ten test problems presented in this chapter include linear systems of order six through ten. For each problem the coefficient matrix, constant vector and final approximate solution vector are given. The comparisons include C.P.U. time used by each program, the number of iterative steps and the number of cycles required. In each case the iterative process was terminated when every corresponding element of the approximate solution vector from two successive cycles was within 0.000005. The C.P.U. time recorded is in seconds. The number of steps is indicated by Iter and the number of cycles by Cycles.

The eight different programs used on each problem are identified in the comparison tables by the following notation.

1. G-E is the direct method of Gauss elimination using the largest pivotal divisor
2. G-S is the single-step method of Gauss-Seidel
3. OLD2 is the old two-dimensional projection method
4. OLD3 is the old three-dimensional projection method
5. HYB4 is the four-dimensional simple hybrid projection method
6. NEW2 is the new two-dimensional projection method
7. NEW3 is the new three-dimensional projection method

8. NEWM is a general new projection method algorithm. By passing a parameter, m, to the algorithm it will solve a given system of n equations using an m-dimensional projection method ($2 \leq m \leq n$). For values of 2 and 3 NEWM is not as fast as the special purpose algorithms, NEW2 and NEW3. Thus in the following tables the results of NEWM are indicated for values of m from four to n. The dimension of the method used is indicated in parenthesis.

Each program was written in PL/1 compiled by the PL/1 optimizing compiler and placed onto a library. Each program uses integer variables as FIXED BINARY (15) and real variables as double precision, i.e., FLOAT DEC (16). Each program was executed on an I.B.M. 360/65 computer under HASP while nothing else was in the system. Each test case used an initial starting vector of zero and rounding error propagation correction was not used. The only I/O performed was to read the various parameters and to print the final approximate solution vector. In each case the ordering of the column vectors of the coefficient vector was consecutive. For example, for n=8 and m=2 the groupings are (1,2), (3,4), (5,6), (7,8) and for m=3 the groupings become (1,2,3), (4,5,6), (6,7,8). Note that given the criteria for the termination of the iterative process two cycles is minimum.

None of the test problems were ill-conditioned enough to cause Gauss elimination to fail. Hence the C.P.U. time given for Gauss elimination should represent a lower bound. Gauss-Seidel failed in four of the ten cases. In test problem #9

Test Problem #1 n=8

$$A = \begin{bmatrix} 1 & 2 & 6 & -2 & 8 & -2 & -9 & 3 \\ 2 & 1 & 4 & -1 & 7 & 9 & 8 & 2 \\ 5 & 3 & -3 & 0 & -1 & -3 & 7 & 7 \\ 3 & 5 & 2 & 1 & 0 & 4 & 2 & -1 \\ -1 & 2 & 9 & 9 & 4 & -6 & -4 & 0 \\ 0 & 1 & -6 & -2 & 2 & 1 & -8 & -2 \\ 1 & 0 & 1 & 4 & -3 & 0 & 1 & 1 \\ 7 & -1 & 3 & 3 & 9 & 2 & 1 & 4 \end{bmatrix} \quad b = \begin{bmatrix} 18 \\ 9 \\ 16 \\ 12 \\ 26 \\ 5 \\ -2 \\ 14 \end{bmatrix}$$

$x' = (-2.136107, 4.685665, -1.280805, 2.697152, 3.467498, -1.677675, .861985, .168814)$

Method		Cycles	Iter	Time
1.	G-E	--	--	.59
2.	G-S	FAILED	--	--
3.	OLD2	109	436	1.47
4.	OLD3	133	399	1.57
5.	HYB4	108	216	.97
6.	NEW2	109	436	.99
7.	NEW3	133	399	.87
8.	NEWM(4)	108	216	2.02
9.	NEWM(5)	42	84	1.29
10.	NEWM(6)	40	80	1.35
11.	NEWM(7)	11	22	1.09
12.	NEWM(8)	2	2	.97

Test Problem #2 n=9

$$A = \begin{bmatrix} 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 4 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 4 & -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 4 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 4 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 \end{bmatrix} \quad b = \begin{bmatrix} 500 \\ 1000 \\ 500 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

x' = (247.023809, 373.511905, 247.023809, 114.583333, 166.666607, 114,583333, 44.6428571, 63.988095, 44.642857)

Method		Cycles	Iter	Time
1.	G-E	--	--	.55
2.	G-S	--	198	.74
3.	OLD2	51	255	1.27
4.	OLD3	34	102	.99
5.	HYB4	35	105	.80
6.	NEW2	51	255	.99
7.	NEW3	34	102	.64
8.	NEWM(4)	35	105	1.52
9.	NEWM(5)	23	46	1.15
10.	NEWM(6)	9	18	.99
11.	NEWM(7)	8	16	.99
12.	NEWM(8)	6	12	1.10
13.	NEWM(9)	2	2	.87

Test Problem #3 n=6

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

x' = (1.000077, 0.999817, 1.000004, 1.000072, 0.999889, 1.000063)

Method	Cycles	Iter	Time
1. G-E	--	--	.44
2. G-S	FAILED	--	--
3. OLD2	2184	6552	7.94
4. OLD3	3778	7556	11.45
5. HYB4	222	444	.94
6. NEW2	2184	6552	4.25
7. NEW3	3778	7556	6.12
8. NEWM(4)	222	444	2.59
9. NEWM(5)	31	62	1.19
10. NEWM(6)	2	2	.79

Test Problem #4 n=6

$$A = \begin{bmatrix} 5 & 3 & 4 & 7 & 2 & 1 \\ 2 & 10 & 6 & 5 & 8 & 3 \\ 5 & 7 & 10 & 2 & 5 & 6 \\ 6 & 8 & 9 & 10 & 3 & 4 \\ 4 & 6 & 8 & 9 & 10 & 2 \\ 3 & 5 & 7 & 9 & 3 & 10 \end{bmatrix} \qquad b = \begin{bmatrix} 22 \\ 34 \\ 35 \\ 40 \\ 39 \\ 37 \end{bmatrix}$$

x' = (0.999991, 1.000000, 1.000007, 1.000002, 0.999997, 0.999997)

Method		Cycles	Iter	Time
1.	G-E	--	--	.50
2.	G-S	--	228	.95
3.	OLD2	800	2400	3.62
4.	OLD3	232	464	1.54
5.	HYB4	37	74	.70
6.	NEW2	800	2400	1.90
7.	NEW3	232	464	.95
8.	NEWM(4)	37	74	.99
9.	NEWM(5)	28	56	1.04
10.	NEWM(6)	2	2	.82

Test Problem #5 n=6

$$A = \begin{bmatrix} 101 & 3 & 6 & 4 & 7 & 1 \\ 1 & 96 & 3 & 2 & 4 & 6 \\ 3 & 2 & 201 & 4 & 1 & 3 \\ 1 & 2 & 3 & 176 & 1 & 1 \\ 2 & 2 & 1 & 2 & 93 & 6 \\ 1 & 2 & 3 & 7 & 2 & 316 \end{bmatrix} \qquad b = \begin{bmatrix} 122 \\ 112 \\ 214 \\ 184 \\ 106 \\ 331 \end{bmatrix}$$

x' = (1.000000, 1.000000, 1.000000, 1.000000, 1.000000, 1.000000)

Method		Cycles	Iter	Time
1.	G-E	--	--	.47
2.	G-S	--	30	.59
3.	OLD2	5	15	.59
4.	OLD3	5	10	.50
5.	HYB4	5	10	.64
6.	NEW2	5	15	.65
7.	NEW3	5	10	.62
8.	NEWM(4)	5	10	.79
9.	NEWM(5)	3	6	.74
10.	NEWM(6)	2	2	.72

Test Problem #6 n=8

$$A = \begin{bmatrix} 10 & 5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 10 & 5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 5 & 10 & 5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5 & 10 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 10 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 10 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5 & 10 & 5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 5 & 10 \end{bmatrix} \qquad b = \begin{bmatrix} 200 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 200 \end{bmatrix}$$

x' = (31.111111, -22.222222, 13.333333, -4.444444,
-4,444444, 13.333333, -22.222222, 31.111111)

Method		Cycles	Iter	Time
1.	G-E	--	--	.64
2.	G-S	--	816	1.10
3.	OLD2	522	2088	3.47
4.	OLD3	377	1131	2.24
5.	HYB4	255	510	1.14
6.	NEW2	522	2088	2.14
7.	NEW3	377	1131	1.62
8.	NEWM(4)	255	510	3.25
9.	NEWM(5)	53	106	1.42
10.	NEWM(6)	24	48	1.09
11.	NEWM(7)	17	34	.97
12.	NEWM(8)	2	2	.60

Test Problem #7 n=9

$$A = \begin{bmatrix} 8 & 3 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 8 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 3 & 8 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 8 & 3 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 8 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 3 & 8 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 8 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 8 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 3 & 8 \end{bmatrix} \qquad b = \begin{bmatrix} 14 \\ 14 \\ 15 \\ 15 \\ 14 \\ 15 \\ 15 \\ 14 \\ 14 \end{bmatrix}$$

x' = (1.000001, 1.000001, 0.999998, 1.000006, 0.999998,
0.999997, 1.000001, 1.000000, 1.000000)

Method		Cycles	Iter	Time
1.	G-E	--	--	.62
2.	G-S	--	108	.64
3.	OLD2	27	135	.84
4.	OLD3	8	24	.70
5.	HYB4	24	72	.87
6.	NEW2	27	135	.74
7.	NEW3	8	24	.52
8.	NEWM(4)	24	72	1.07
9.	NEWM(5)	9	18	.74
10.	NEWM(6)	4	8	.60
11.	NEWM(7)	3	6	.79
12.	NEWM(8)	3	6	.94
13.	NEWM(9)	2	2	.60

Test Problem #8 n=10

$$A = \begin{bmatrix} 6 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 1.9 & .1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & .1 & 9 & 9 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 3.9 & .1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & .1 & -7 & -7 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -3 & -2.9 & .1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & .1 & 9 & 9 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 2.1 & .1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & .1 & 5 & 5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 10.1 \end{bmatrix} \quad b = \begin{bmatrix} 120 \\ 23 \\ 45 \\ 72 \\ 38 \\ 30 \\ 36 \\ 72 \\ 144 \\ 29 \end{bmatrix}$$

x' = (-2699.187341, 2719.187341, 2549.187341, -2574.400534,
-845.872827, 803.667108, -1769.833862, 1764.909127,
-946.316437, 939.818255)

Test Problem #8 (Continued) n=10

Method		Cycles	Iter	Time
1.	G-E	--	--	.75
2.	G-S	FAILED	--	--
3.	OLD2	606	3030	5.12
4.	OLD3	1250	5001*	10.22
5.	HYB4	201	603	1.55
6.	NEW2	606	3030	3.27
7.	NEW3	1250	5001*	5.65
8.	NEWM(4)	201	603	3.89
9.	NEWM(5)	2500	5001**	37.09
10.	NEWM(6)	52	104	1.60
11.	NEWM(7)	48	96	1.79
12.	NEWM(8)	26	52	1.50
13.	NEWM(9)	26	52	1.30
14.	NEWM(10)	2	2	.72

---

*The step limit was reached. The residual norm squared was reduced to 8332.

**The step limit was reached. The residual norm squared was reduced to 2119.

Test Problem #9 n=10

$$A = \begin{bmatrix} 60 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 200 & 60 & 0 & 0 & 0 & & 0 & 0 & 0 & 0 \\ 175 & 25 & 60 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 50 & 100 & 50 & 60 & 0 & 0 & 0 & 0 & 0 & 0 \\ 75 & 25 & 75 & 25 & 60 & 0 & 0 & 0 & 0 & 0 \\ 50 & 50 & 25 & 15 & 60 & 60 & 0 & 0 & 0 & 0 \\ 20 & 60 & 20 & 80 & 5 & 15 & 60 & 0 & 0 & 0 \\ 10 & 50 & 10 & 30 & 50 & 35 & 15 & 60 & 0 & 0 \\ 5 & 15 & 25 & 55 & 25 & 50 & 10 & 5 & 60 & 0 \\ 10 & 5 & 15 & 30 & 40 & 40 & 30 & 15 & 15 & 60 \end{bmatrix} \quad b = \begin{bmatrix} 100 \\ 2300 \\ 4500 \\ 7600 \\ 1200 \\ 8500 \\ 7200 \\ 5800 \\ 2200 \\ 3000 \end{bmatrix}$$

x' = (1.666667, 32,777778, 56.481481, 23.580247, -76.167695, 159.701646, 2.821073, 17.478888, -120.090762, -10.366622)

Methods		Cycles	Iter	Time
1.	G-E	--	--	.70
2.	G-S	--	20	.59
3.	OLD2	1294	6470	10.64
4.	OLD3	1920	7680	15.52
5.	HYB4	596	1788	3.32
6.	NEW2	1294	6470	6.64
7.	NEW3	1920	7680	9.15
8.	NEWM(4)	596	1788	11.70
9.	NEWM(5)	555	1110	8.72
10.	NEWM(6)	463	926	8.95
11.	NEWM(7)	170	340	4.25
12.	NEWM(8)	159	318	4.39
13.	NEWM(9)	6	12	.80
14.	NEWM(10)	2	2	.80

Test Problem #10 n=7

$$A = \begin{bmatrix} 100 & 245 & 630 & -75 & -896 & 0 & 0 \\ 147 & -25 & 300 & 756 & 0 & 13 & 0 \\ 0 & 123 & 452 & -1 & 753 & 249 & 0 \\ 12 & -568 & 30 & 21 & -78 & 0 & 85 \\ 0 & 0 & -23 & 563 & 752 & 843 & -1 \\ 145 & -20 & 1 & 7 & 30 & 6 & 0 \\ -1 & 0 & 43 & 8023 & 63 & 0 & 3 \end{bmatrix} \qquad b = \begin{bmatrix} 100 \\ 350 \\ -496 \\ -7002 \\ 143 \\ 579 \\ 3045 \end{bmatrix}$$

x' = (4.435283, 3.141906, -1.777156, 0.412942, -0.041611, 0.190498, -61.520275)

Method		Cycles	Iter	Time
1.	G-E	--	--	.75
2.	G-S	FAILED	--	--
3.	OLD2	809	3236	4.20
4.	OLD3	684	2052	3.80
5.	HYB4	685	1370	2.07
6.	NEW2	809	3236	2.45
7.	NEW3	684	2052	2.15
8.	NEWM(4)	685	1370	7.40
9.	NEWM(5)	688	1366	9.05
10.	NEWM(6)	26	52	1.04
11.	NEWM(7)	2	2	.60

using a lower triangular coefficient matrix it proved superior, however, in the remaining cases the C.P.U. time was competitive with one or more of NEW2, NEW3 or HYB4 methods. In all cases the new projection methods were shown superior to the old projection methods as expected. In test problems #3, #4, #8, #9, and #10 where the number of iterations required is large the C.P.U. times for OLD2 and OLD3 versus NEW2 and NEW3 approach the theoretical limit of two to one. In test problem #5 where the number of iterations required is near minimal the C.P.U. times are statistically no different given the accuracy of the C.P.U. clock.

The C.P.U. times for NEWM are given for the sake of completeness. As stated earlier due to the expense of generality the NEWM algorithm for a given dimension is more costly in C.P.U. time than a special purpose new projection algorithm of the same dimension. This is quite evident when comparing NEWM(4) and HYB4. HYB4 should require about the same C.P.U. time as NEWM(4). This shows if nothing else that if a subroutine package of projection methods is to be developed an individual algorithm for each dimension should be used.

Each test problem was run again using initial vector algorithms. In test problems #3, #4, #5, and #7 where the solution is a constant vector initial vector algorithms I and II guessed the solution immediately. These, however, are special cases. In the other test problems both initial

vector algorithms reduced the number of steps required and C.P.U. time slightly in only a few cases. Most of the time the number of steps was slightly increased as was C.P.U. time compared to using an initial vector of zero. The conclusion reached by the author is that initial vector algorithms I or II do not possess any particular advantage in selecting an initial vector over an initial vector of one's choice. The author's experience on this matter has been that regardless of the initial vector the projection method converges rapidly to the general area of the solution and then converges very slowly to the actual solution. With this observation in mind the initial vector used does not seem to be of any major concern.

Each test problem was run again using rounding error propagation correction every five cycles. In only one case the solution obtained differed in the third decimal place. In the remainder of cases the solutions were the same to the fourth or fifth decimal place. It is hard to say how much of this is due to rounding error propagation and how much is due to the tolerance limit used. At any rate rounding error propagation did not seem to occur to any great extent if any at all. This undoubtedly was due to the fact that double precision arithmetic was performed throughout. Although single precision examples were not run, it is suggested that rounding error propagation

correction should accompany it. The amount of additional calculations required per iteration step is offset by the reduction in C.P.U. time from double to single precision arithmetic.

VIII. SUMMARY AND FUTURE RESEARCH

A. Summary

This dissertation develops a new projection method for any dimension which is equivalent to the old or conventional projection method. The new projection method requires less than half the number of arithmetic operations to be performed per iteration step. In addition the number of operations per component per iterative step for the new method is independent of the dimension of the method used. This is not true for the old projection method. This means that the number of arithmetic operations required per cycle is the same regardless of the dimension used. One concludes from this that the dimension to use to solve a system of n equations should be n since this reduces the number of cycles to a minimum, namely one. This, however, is nothing more than the original problem in that a symmetic system of n equations must be solved. The iteration matrix for the new projection method is easily obtained and is presented for the one and two-dimensional cases. For a system of n equations the iteration matrix clearly shows for an m-dimensional projection method that the m components of the approximate solution vector to be determined during an iterative step is obtained from a predetermined linear combination of the other n-m components of the approximate

solution vector.

The new projection method has a few interesting side effects. First, it may be subject to rounding error propagation. An algorithm is presented for the prevention of rounding error propagation. Even if the algorithm is used every cycle the number of arithmetic operations is still less than that required by the old projection method. Secondly, the initial residual vector, r^o, is calculated as if the initial approximate solution vector, x^o, is zero, i.e., r^o becomes the constant vector of the system. x^o, however, may be any arbitrary vector and need not be zero. This apparent inconsistency becomes a blessing for x^o can be chosen at random and r^o need not be calculated from it. This observation lead to the development of two initial vector algorithms. The test problems presented indicate that neither initial vector algorithm proved to be of any significant advantage.

This dissertation also develops hybrid projection methods by using two or more new projection methods as a basis. Simple hybrid projection methods, those based on two new projection methods, are shown to be equivalent for any dimension to the old projection methods, hence to the new projection methods. The simple hybrid methods require less than half the number of arithmetic operations to be performed per iteration step. In addition the number of

operations per component per iterative step is independent of the dimension of the method used. Although the new and simple hybrid projection methods require about the same number of arithmetic operations per iteration step, the number of fixed overhead calculations required for the simple hybrid method is considerably more. However, simple hybrid projection methods are judged to be competitive with the new projection methods. Multiple hybrid projection methods, those based on more than two new projection methods, are shown to be inferior to the simple hybrid projection methods.

Even with additional overhead calculations required, the new projection method is shown to obtain the solution in less number of arithmetic operations than the old projection method. This is true independent of the dimension of the method used, the initial vector or the linear system itself. The theoretical developments presented are substantiated by the test problems presented in Chapter VII.

For best results it is suggested that the material presented in this dissertation be used with some a priori algorithm for selecting groups of columns vectors of the coefficient matrix based on results in (13, 14, 18, 6).

B. Future Research

The following areas are worthy of future research:

1. Further study with initial vector algorithms may yield a mechanism for significantly reducing the number of cycles normally required. It may be advantageous to use two or more initial vectors and process a few cycles simultaneously with each. Perhaps with two or more sequences of approximate solution vectors converging to the same solution a mechanism for accelerating the process or obtaining the solution directly can be determined.

2. In the author's opinion the relative worth of hybrid projection methods has not been fully determined. Perhaps more research in this area may prove profitable.

3. The results presented in theorem 6.2 needs to be expanded. The author feels that this can be used to show a 2p-dimensional projection method is an acceleration of a p-dimensional method, provided the number of equations to be solved is a multiple of 2p.

IX. BIBLIOGRAPHY

1. Bodewig, E. Matrix Calculus. Amsterdam, Netherlands: North-Holland Publishing Company, 1959.

2. Forsythe, G. E. Solving Linear Algebraic Equations can be Interesting. Bul. Amer. Math. Soc. 59 (1953): 299-329.

3. Fox, L. An Introduction to Numerical Linear Algebra. New York, N.Y.: Oxford University Press, 1964.

4. Garza, A. de la. An Iterative Method for Solving Systems of Linear Equations. Union Carbide and Carbon Corp. K-25 Plant (Oak Ridge, Tennessee) Report K-731, 1951.

5. Gastinel, N. Linear Numerical Analysis. New York, N.Y.: Academic Press, Inc., Publishers, 1970.

6. Georg, D. D. Criteria for Ordering Columns in Two-Dimensional Projection Methods. Unpublished Masters Thesis, Iowa State University, Ames, Iowa, 1973.

7. Hartree, F. R. S. Numerical Analysis. London: Oxford University Press, 1952.

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

9. Householder, A. S. The Theory of Matrices in Numerical Analysis. New York, N.Y.: Blaisdell Publishing Company, 1964.

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

11. Iverson, K. E. A Programming Language. New York, N.Y.: John Wiley and Son, Inc., 1962.

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

13. Pyron, H. D. A Non-Statistical Two-Dimensional Acceleration for the One-Dimensional Projection Method. Unpublished Ph.D. Dissertation, Iowa State University, Ames, Iowa, 1970.

14. Shen, I-Ming. Acceleration of the Projection Method for Solving Systems of Linear Equations. Unpublished Ph.D. Dissertation, Iowa State University, Ames, Iowa, 1970.

15. Southwell, R. V. Relaxation Method in Engineering Science; A Treatise on Approximate Computation. New York, N.Y.: Oxford University Press, Inc., 1940.

16. Stanton, R. G. Numerical Methods for Science and Engineering. Englewood Cliffs, N.J.: Prentice-Hall, 1961.

17. Tewarson, R. P. Projection Methods for Solving Sparse Linear Systems. Computer J. 12 (1969): 77-80.

18. Tokko, Mok. Optimal Three-Dimensional Projection Method for Solving Linear Algebraic Equations. Unpublished Ph.D. Dissertation, Iowa State University, Ames, Iowa, 1972.

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

X. ACKNOWLEDGMENTS

The author wishes to express his sincere appreciation for the encouragement and support of Dr. Roy F. Keller during the preparation of this dissertation.

I am especially grateful for the support and encouragement of my wife, Jenny, during my entire graduate program. Special thanks are due to Mrs. Marlys Phipps for the expert typing of this thesis.

XI. APPENDIX: COMPUTER PROGRAM IMPLEMENTATION

A. Documentation for the New m-Dimensional Projection Method (NEWM)

The new m-dimensional projection method described in Chapter IV has been implemented with a PL/1 program which was run on an I.B.M. 360/65 Hasp system.

1. Variable dictionary for NEWM

Those variables followed by (*) are ones which must be specified by the user. All variables containing fractional information are declared double precision, i.e., FLOAT DECIMAL (16). Integer variables are declared FIXED BINARY (15).

A - coefficient matrix of the system (*)

AA - matrix of the inner products of columns of A

B - constant vector of the system (*)

BA - vector of inner products of B and A

CNT - constant vector of the system to be solved at each step

COLS - matrix indicating what columns of A are to be grouped together at each step. This program assumes that the columns are grouped consecutively. If a different criteria is to be used for determining the groupings of the columns for A then code for doing so should replace the

code in NEWM between the following two comments:

/* BEGINNING OF THE CODE FOR DETERMINING COLS */

/* END OF THE CODE FOR DETERMINING COLS */

INITX - indicates what initial vector algorithm to use (*)

M - dimension of the projection method (*)

MM - coefficient matrix of the system to be solved at each step

N - dimension of A

NORMR - norm squared of the residual vector

PRT - indicates if summary information is to be printed after every step or not (*)

R - residual vector

RA - inner project of the initial residual vector with the columns of A

RD - indicates how often a correction for rounding error propagation should occur (*)

ROWSUMA - vector of the row sum of A

SA - a save area matrix used by subroutines UPDIA and CONVERT

SAVEX - approximate solution vector of the previous step

STEP - step counter

STEPLT - step limit (*)

TOLER - tolerance limit for determining convergence (*)

X - approximate solution vector, depending on INITX an initial X may need to be specified (*)

Y - the accumulated solution vector when rounding error propagation prevention is employed

2. Subroutines used by NEWM

UPDIA - This routine upper triangularizes a symmetric coefficient matrix by means of Gauss elimination with pivoting in the main diagonal. The upper triangular part of the matrix is passed to the routine columnwise. The code for UPDIA was taken in part from a subset of the code from GELS, a Gauss elimination routine for symmetric matrices from the I.B.M. Fortran scientific subroutine package.

CONVERT - This routine converts a constant vector as Gauss elimination would do when upper triangularizing a symmetric coefficient matrix. The code for CONVERT was taken in part from a subset of the code from the above mentioned GELS.

3. Input data

The following data is input in the order given:

TOLER - The iterative process terminates when every element of the approximate solution vector from two successive cycles is within this value. Suggested values for TOLER range from 10^{-4} to 10^{-7}.

PRT - This is an indicator for the amount of output desired. A value of zero indicates minimum output. Only the number of steps and the solution are given. A value not equal to zero will cause the following variables to be printed: TOLER, STEPLT, M, X, B, RA, A, AA, MM, COLS, SA, and BA when RD > 0. In addition after each iterative step, STEP, X, NORMR and an indication of what columns of A were used during the step are printed. Upon convergence the solution, the number of steps required, and resulting norm is given.

STEPLT - the maximum number of iterative steps to be allowed

RD - A value less than or equal to zero indicates no round-off error propagation prevention is to be employed. A positive value of w indicates round-off error propagation prevention is to be employed every w cycles.

M - the dimension of the projection method to be used ($2 \leq M \leq N$)

INITX - A value of 0 indicates the initial guess for x is to be read in. A value of 1 indicates initial vector I is to be used otherwise initial vector II is used.

N - the dimension of A

A - the coefficient matrix (row major order)

B - the constant vector

X - the initial guess for the solution provided INITX = 0

The input data are read by GET LIST, i.e., all data items must be separated by a comma and/or blank(s).

B. Documentation for NEW2, NEW3, and HYB4

1. Dictionary of variables

A - coefficient matrix of the system

AA - matrix of the inner products of the columns of A

B - constant vector of the system

N - dimension of A

NORMR - norm squared of the residual vector

PRT - indicates if summary information is to be printed after every step or not

QUIT - is an indicator if the tolerance limit has been reached as one processes through a cycle

R - residual vector

RA - inner product of the initial residual vector with the columns of A

STEP - step counter

STEPLT - step limit

TOLER - tolerance limit for determining convergence

X - the approximate solution vector

2. Additional variables for NEW2

C - array containing values from Equation 4.19b

G - array containing values from Equation 4.13

P - is an index in C and G containing information about the pair of columns of A used during the iteration step

T - array containing values from Equation 4.12

W - an integer indicating the first of the pair of columns of A used during the iterative step

3. Additional variables for NEW3

COS - array containing values from Equation 3.11b

D - array containing values from Equation 3.11a

P - an integer indicating the first of the triple of columns of A used during the iterative step

S - array containing values from Equation 3.9

T - array containing values from Equation 3.10

W - is an index in D, S, and T containing information about the triple of columns of A used during the iterative step

4. Additional variables for HYB4

AL - array containing information from Equation 4.13

C - array containing information from Equation 4.19b

QR - is an index in AL and C containing information about the quadruple of columns of A used during the iterative step

T - array containing information from Equation 4.12

W - an integer indicating the first of the quadruple of columns of A used during the iterative step

P, Q, SI, SJ, SK, SL are all quantities containing information from Equation 6.5a.

All three algorithms input TOLER, PRT, STEPLT, N, A, and B in the order given. The data are read by GET LIST and arrays are assumed to be given in row major order. The amount of output obtained depends on the value of PRT. Zero indicates minimum output. Only the number of steps required and the solution are given. A nonzero value for PRT will result in the output of all arrays used as well as the output of STEP, X, and NORMR at each iterative step.

C. Program Listings

```
NEWM: PROCEDURE OPTIONS(MAIN);
 DECLARE(K,L,W,RD,P,PRT,STEPLT,QUIT,RDCT,I,J,N,M,Q,PP,
 IER,LST,NL,INITX)
 FIXED BIN,((X(*),Y(*),BA(*),A(*,*),R(*),B(*),COLS(*,*),
 SA(*,*,*),AUX(*),
 AA(*,*),RA(*),MM(*,*)  ,SAVEX(*),CNT(*),ROWSUMA(*))
 CONTROLLED,
 TOLER,STEP,NORMR,ESP)FLOAT DEC(16);
 ESP=0.000001;    QUIT=1;    RDCT=0;    STEP=0;
 /* GET LIST THE FOLLOWING PARAMETERS IN THE ORDER GIVEN:*/
 /*          TOLER,PRT,STEPLT,RD,M,INITX,N,A,B,X          */
 GET LIST(TOLER,PRT,STEPLT,RD,M,INITX,N);
 IF M>N | M<2 THEN
 DO; PUT SKIP EDIT('ABEND--M IS OUT OF RANGE')(A); RETURN;
 END;
 W=(N+M-1)/M;  NL=M*(M+1)/2;
 ALLOCATE X(N),B(N),A(N,N),R(N),AA(N,N),RA(N),
 SA(M+1,M,W),AUX(M-1),MM(NL,W),
 SAVEX(M),CNT(M),COLS(M,W);
 IF PRT¬=0 THEN SA=0;
 GET LIST(A,B);
 IF RD>0 THEN
 DO; ALLOCATE Y(N),BA(N); Y=0;
     DO I=1 TO N; BA(I)= SUM(B*A(*,I)); END;
 END;
 /* CALCULATE R,AA,RA,MM                                  */
 R=B;
 DO I= 1 TO N; RA(I)=SUM(R*A(*,I));
     DO J=I TO N;
         AA(I,J)= SUM(A(*,I)*A(*,J));
         AA(J,I)=AA(I,J);
     END;
 END;
 /*             DETERMINE IF AN INITIAL                   */
 /*             VECTOR ALGORITHM IS TO BE USED            */
 IF INITX=0 THEN GET LIST(X); ELSE
     DO; ALLOCATE ROWSUMA(N);
         DO I=1 TO N; ROWSUMA(I)=SUM(A(I,*)); END;
         IF INITX=1 THEN /*        ALGORITHM I           */
         X=SUM(B*ROWSUMA)/SUM(ROWSUMA*ROWSUMA);
         ELSE
         DO;  /*                   ALGORITHM II          */
             IF RD<=0 THEN
                 DO; ALLOCATE BA(N);
                     DO I=1 TO N; BA(I)=SUM(B*A(*,I)); END;
                 END;
             DO I=1 TO N;
                 X(I)= BA(I)/SUM(A(*,I)*ROWSUMA);
             END;
         END;
```

```
      END;
  /* DETERMINE BY SOME CRITERIA WHICH COLUMNS ARE                    */
  /*  GROUPED TOGETHER ON EACH OF THE W ITERATIONS                   */
  /*  PER CYCLE.  PLACE THIS INFORMATION ONTO  COLS                  */
  /*  IN INCREASING ORDER.                                           */
  /*  THIS ROUTINE ASSUMES THAT THE COLUMNS OF A ARE                 */
  /*  GROUPED CONSECUTIVELY.                                         */
  /*  BEGINNING OF THE CODE FOR DETERMINING COLS                     */
  P=1-M;
  DO I=1 TO W;
      IF I=W THEN P=N-M+1; ELSE P=P+M;
      K=0;
      DO J=P TO P+M-1;  K=K+1;  COLS(K,I)=J;  END;
  END;
  /*  END OF THE CODE FOR DETERMINING COLS                           */
  DO I=1 TO W;
      /* DETERMINE FROM COLS(*,I) THE COEFFICIENT MATRIX             */
      /* AND STORE THE UPPER TRIANGULAR PORTION BY                   */
      /* COLUMNS INTO MM(*,I)                                        */
      L=0;
      DO J=1 TO M;
      DO K=1 TO J;
          L=L+1;  MM(L,I) = AA(COLS(K,I),COLS(J,I));
      END;
      END;
      CALL UPDIA(MM(*,I),I);
  IF IER¬=0 THEN
      DO;  IF IER=-1 THEN
          DO; PUT SKIP EDIT
          ('NO RESULT. STEP IN THE CYCLE IS ',I)
          (A,F(4));  GO TO PRINT;
          END;
          PUT SKIP EDIT
          ('WARNING--POSSIBLE LOSS OF SIGNIFICANCE',
          ' HAS OCCURED','THE COLUMNS OF A  INVOLVED ARE:',
          (COLS(J,I) DO J=1 TO M))(A,A,SKIP,A,SKIP,(M)F(4));
      END;
  END;
  Q=0;
  /*  END OF THE FIXED OVER HEAD OPERATIONS                          */
  IF PRT=0 THEN GO TO LOOP;

 PRINT:
  PUT PAGE EDIT
  ('THE SUCCESSIVE APPROXIMATION VECTOR TOLERANCE= ',
  TOLER,'THE STEP LIMIT= ',STEPLT,
  'THE DIMENSION OF THE METHOD USED IS ',M)
  (A,F(15,10),SKIP,A,F(5),SKIP,A,F(3));
  PUT SKIP(3) EDIT
  ('X','B','RA',(X(I),B(I),RA(I) DO I=1 TO N))
```

```
(COL(10),A,COL(30),A,COL(50),A,
(N)(SKIP,F(15,6),COL(20),F(15,6),COL(40),F(15,6)));
PUT SKIP(2) EDIT('THE COEFFICIENT MATRIX')(COL(50),A);
DO I=1 TO N;
    PUT SKIP EDIT((A(I,J) DO J=1 TO N)) (R(LAB));
END;
PUT SKIP(2) EDIT('THE AA MATRIX')(COL(50),A);
DO I=1 TO N;
    PUT SKIP EDIT((A(I,J) DO J=1 TO N)) (R(LAB));
END;
PUT SKIP(2) EDIT('THE MM MATRIX')(COL(20),A);
DO I=1 TO NL;
    PUT SKIP EDIT((MM(I,J) DO J=1 TO W)) (R(LAB));
END;
PUT SKIP(2) EDIT('THE COLS MATRIX')(COL(20),A);
DO I=1 TO M;
    PUT SKIP EDIT((COLS(I,J) DO J=1 TO W)) (R(LAB));
END;
PUT SKIP(2) EDIT('SA  ARRAY FOLLOWS IN LEVELS')(COL(30),A);
DO K=1 TO W;
PUT SKIP(2) EDIT('LEVEL ',K)(COL(35),A,F(3));
    DO I=1 TO M+1;
        PUT SKIP EDIT((SA(I,J,K) DO J=1 TO M)) (R(LAB));
    END;
END;
IF RD>0 THEN
DO; PUT SKIP(2) EDIT('THE BA VECTOR',
    (BA(I) DO I=1 TO N))(COL(50),A,R(LAB));
END;
IF IER < 0 THEN RETURN;
PUT PAGE EDIT('STEP','X','NORM R','GROUP')
(A,COL(40),A,COL(119),A,COL(127),A);
LAB: FORMAT((N)(SKIP,( 8)F(15,6)));

LOOP:
IF STEP > STEPLT THEN
DO; DO I= 1 TO N; R(I)=B(I)-SUM(A(I,*)*X); END;
    NORMR= SUM(R*R);
    PUT SKIP EDIT('STEP LIMIT WAS REACHED.',
    ' WITH NORM SQUARED R= ',NORMR)(A,A,F(15,6));
    GO TO EXIT;
END;
Q=Q+1;     /* Q IS WHICH ITERATION FOR THE CYCLE          */
           /*  IE., RANGE FROM  1 TO W                    */
IF Q>W THEN
DO; IF QUIT=1 THEN
    DO; PUT SKIP EDIT('TOLERANCE LIMIT WAS REACHED')
        (A); GO TO EXIT;
    END;
    /* THE END OF A CYCLE HAS OCCURED BUT                 */
```

```
      /* MORE PROCESSING REMAINS                                    */
      IF RD>0 THEN
      DO; RDCT=RDCT+1; IF RDCT=RD THEN
          DO; RDCT=0;
              /* PREVENT ROUND OFF                                  */
              /* ERROR PROPAGATION                                  */
              Y=Y+X;
              DO I= 1 TO N;
                 RA(I)=BA(I)-SUM(Y*AA(*,I));
              END;
              X=0;
          END;
      END;
  QUIT=1; Q=0; GO TO LOOP;
  END;
  STEP=STEP+1;
  DO I=1 TO M; SAVEX(I) = X(COLS(I,Q)); END;
  DO I=1 TO M; CNT(I) = RA(COLS(I,Q)); PP=1;
      DO J=1 TO N;
          IF X(J) = 0  THEN GO TO BOT;
          IF PP<=M THEN IF J = COLS(PP,Q) THEN
              DO; PP=PP+1; GO TO BOT; END;
          CNT(I) = CNT(I)-X(J)*AA(J,COLS(I,Q));
      BOT: END;
  END;
  CALL CONVERT(Q);     /*  CNT IS ALTERED                          */
  CALL BACKSUB(MM(*,Q));
  /* CNT CONTAINS THE NEW VALUES FOR X                             */
  DO I=1 TO M;  X(COLS(I,Q)) = CNT(I);  END;
  IF QUIT=1 THEN
  DO;
      DO I=1 TO M; IF ABS(SAVEX(I)-X(COLS(I,Q))) >TOLER THEN
          DO; QUIT=0; GO TO OUT; END;
      END;
  END;
 OUT: IF PRT=0 THEN GO TO LOOP;
  PUT SKIP EDIT(STEP,(X(I) DO I= 1 TO N))
  (F(5),(N)(COL(6),( 8)F(13,6)));
  IF RD>0 THEN
      DO; PUT EDIT(Q)(COL(130),F(3)); GO TO LOOP; END;
  DO I= 1 TO N; R(I)=B(I)-SUM(A(I,*)*X); END;
  NORMR= SUM(R*R);
  PUT EDIT(NORMR,Q)(COL(114),F(15,6),COL(130),F(3));
  GO TO LOOP;
 EXIT:
  IF RD>0 THEN X=X+Y;
  PUT SKIP(2) EDIT('NUMBER OF STEPS = ',STEP,
  'THE SOLUTION FOLLOWS',(X(I) DO I= 1 TO N))
  (A,F(6),SKIP,COL(50),A,(N)(SKIP,( 8)F(15,6)));
```

```
/***************** SUBROUTINE UPDIA ************************/
UPDIA: PROCEDURE(A,Q);
 DECLARE(L,K,J,LEND,LR,LT,LL,II,LLST,LLD,SACT,I,Q)FIXED BIN;
 DECLARE(PIVI,TB,TOL,PIV,A(*))FLOAT DEC(16);
 /* SEARCH FOR THE GREATEST MAIN DIAGONAL ELEMEMENT      */
 IER=0;  PIV=0;  L=0;
 DO K=1 TO M;    L=L+K;  TB=A(L);
 IF TB>PIV THEN DO; PIV=TB; I=L; J=K; END;
 END;
 TOL=ESP*PIV;
 /* J IS WHICH COLUMN AND I IS WHICH ELEMENT IN A        */
 /* PIV IS THE VALUE OF A(I) WHICH IS A(J,J)             */
 /* START THE ELIMINATION LOOP;                          */
 LST=0;  LEND=M-1;
 DO K=1 TO M;
 IF PIV<=0 THEN DO; IER=-1; GO TO EXIT; END;
 IF IER=0  THEN IF  PIV<=TOL THEN IER=K-1;
 LT=J-K;   LST=LST+K;   PIVI=1./A(I);
 /* SAVE FOR CONVERT THE VALUES OF LT AND PIVI IN SA     */
 SA(1,K,Q)=LT;  SA(2,K,Q)=PIVI;
 /* IS ELIMINATION TERMINATED                            */
 IF K>=M THEN GO TO EXIT;
 /* ROW AND COLUMN INTERCHANGE AND PIVOT ROW REDUCTION   */
 /* IN MATRIX A ELEMENTS OF PIVOT COLUMN ARE SAVED IN    */
 /* AUXILLARY VECTOR AUX                                 */
 LR=LST+(LT*(K+J-1))/2; LL=LR; L=LST;
     DO II=K TO LEND;
     L=L+II; LL=LL+1; IF L=LR THEN
         DO; A(LL)=A(LST); TB=A(L); GO TO L13; END;
         ELSE IF L>LR THEN LL=L+LT;
     TB=A(LL); A(LL)=A(L);
     L13:  AUX(II)=TB; A(L)=PIVI*TB;
     END;
 /* SAVE COLUMN INTERCHANGE INFORMATION                  */
 A(LST)=LT;
 /* ELEMENT REDUCTION AND SEARCH FOR NEXT PIVOT          */
 PIV=0; LLST=LST; LT=0; SACT=2;
     DO II=K TO LEND;
     /* ROW REDUCTION                                    */
     PIVI=-AUX(II); LL=LLST; LT=LT+1;
         DO LLD=II TO LEND;
         LL=LL+LLD; L=LL+LT;
         A(L)=A(L)+PIVI*A(LL);
         END;
     LLST=LLST+II; LR=LLST+LT;
     TB=ABS(A(LR)); IF TB> PIV THEN
         DO; PIV=TB; I=LR; J=II+1; END;
     /* SAVE FOR CONVERT THE VALUES OF LT AND PIVI IN SA */
     SACT=SACT+1; SA(SACT,K,Q)=PIVI;
     END;
```

```
 END;
EXIT: END UPDIA;

/***************** SUBROUTINE CONVERT **********************/
CONVERT: PROCEDURE(Q);
 DECLARE(K,LT,L,LL,II,LR,SACT,Q)FIXED BINARY;
 DECLARE(TB,PIVI)FLOAT DEC(16);
 DO K=1 TO M;
 LT=SA(1,K,Q); PIVI=SA(2,K,Q); L=K; LL=L+LT;
 TB=PIVI*CNT(LL); CNT(LL)=CNT(L); CNT(L)=TB;
 IF K=M THEN RETURN;
 LT=0; SACT=2;
      DO II=K TO M-1; SACT=SACT+1;
      PIVI=SA(SACT,K,Q); LT=LT+1;
      LR=K; LL=LR+LT;
      CNT(LL)=CNT(LL)+PIVI*CNT(LR);
      END;
 END;
END CONVERT;

/***************** SUBROUTINE BACKSUB **********************/
BACKSUB: PROCEDURE(A);
 DECLARE(II,I,L,J,LL,K,LT)FIXED BINARY;
 DECLARE(TB,A(*))FLOAT DEC(16);
 II=M;    LST=NL;
 DO I=2 TO M;
 LST=LST-II; II=II-1; L=A(LST)+.5;
 J=II; TB=CNT(J); LL=J; K=LST;
      DO LT=II TO M-1;
      LL=LL+1; K=K+LT;
      TB=TB-A(K)*CNT(LL);
      END;
 K=J+L; CNT(J)=CNT(K); CNT(K)=TB;
 END;
END BACKSUB;

END NEWM;
```

```
NEW2: PROCEDURE          OPTIONS(MAIN);
 DECLARE (K,L,W,RD,P,PRT,STEPLT,QUIT,RDCT,I,J,N)
 FIXED BINARY, ((X(*),C(*,*),Y(*),BA(*),B(*),A(*,*),
 R(*),AA(*,*),G(*),T(*),RA(*)) CONTROLLED,TOLER,
 XX,D1,D2,STEP,NORMR) FLOAT DECIMAL(16);
 GET LIST(TOLER,PRT,STEPLT);
 RD=0;
 QUIT=1;
 RDCT=0;
 P=-1;
 STEP=0;
 GET LIST(N);
 ALLOCATE X(N),C(N+1,N  ),B(N),A(N,N),R(N),AA(N,N),
 RA(N),G(N+1),T(N+1);
 GET LIST(A,B);  X=0;
 IF RD¬=0 THEN
 DO; ALLOCATE Y(N),BA(N); Y=0;
     DO I=1 TO N; BA(I)=SUM(B*A(*,I)); END;
 END;
 /* CALCULATE R,RA,AA,T,G AND C        */
 R=B;
 DO I=1 TO N;  RA(I)= SUM(R*A(*,I));
     DO J=I TO N;
         AA(I,J)=SUM(A(*,I) * A(*,J));
         AA(J,I)=AA(I,J);
     END;
 END;
  IF PRT¬=0  THEN DO;
 T(N+1)=0; G(N+1)=0; C(N+1,*)=0;
  END;
  /*           CALULATIONS FOR T                                  */
  DO I=1 TO N BY 2;
  IF I=N THEN W=N-1;  ELSE W=I;
  T(I)  = AA(W,W+1)*AA(W,W+1)-AA(W,W)*AA(W+1,W+1);
  T(I+1)  = T(I);
  END;
  /*           CALULATIONS FOR G                                  */
  DO I=1 TO N BY 2;
  IF I=N  THEN W= N-1;  ELSE W=I;
  G(I)  =(RA(W+1)*AA(W,W+1)-RA(W)*AA(W+1,W+1))/T(I);
  G(I+1)=(RA(W)*AA(W+1,W)-RA(W+1)*AA(W,W))/T(I);
  END;
  /*           CALULATIONS FOR C                                  */
  DO I=1 TO N BY 2;   IF I=N THEN W= N-1;  ELSE W=I;
  DO K=1 TO N;
  C(I,K)  =(AA(W,K)*AA(W+1,W+1)
          -AA(W+1,K)*AA(W,W+1))/T(I);
  C(I+1,K)=(AA(W+1,K)*AA(W,W)
         -AA(W,K)*AA(W+1,W))/T(I);
  END;
```

```
  END;
 IF PRT= 0 THEN GO TO LOOP;
 PUT PAGE EDIT
 ('THE SUCCESSIVE APPROXIMATION VECTOR TOLERANCE= ',
 TOLER,'  THE STEP LIMIT = ',STEPLT)(A,F(15,10),A,F(5));
 PUT SKIP(3) EDIT
 ('X','B','R',(X(I),B(I),R(I) DO I= 1 TO N))
 (COL(10),A,COL(30),A,COL(50),A,
 (N)(SKIP,F(15,6),COL(20),F(15,6),COL(40),F(15,6)));
 PUT SKIP(2) EDIT('C MATRIX')(COL(50),A);
 DO I= 1 TO N+1;
     PUT SKIP EDIT((C(I,J) DO J=1 TO N  )) (R(LAB1));
 END;
 PUT SKIP(2) EDIT('A MATRIX')(COL(50),A);
 DO I= 1 TO N;
     PUT SKIP EDIT((A(I,J) DO J=1 TO N)) (R(LAB1));
 END;
 PUT SKIP(2) EDIT('AA MATRIX')(COL(50),A);
 DO I= 1 TO N;
     PUT SKIP EDIT((AA(I,J) DO J=1 TO N)) (R(LAB1));
 END;
 PUT SKIP(2) EDIT('G VECTOR',G) (R(LAB2));
 PUT SKIP(2) EDIT('RA VECTOR',RA)(R(LAB2));
 PUT SKIP(2) EDIT('T VECTOR',T) (R(LAB2));
 NORMR= SUM(R*R);
 PUT PAGE EDIT('STEP','X','NORM R','PAIR')
 (A,COL(40),A,COL(119),A,COL(128),A);
 PUT SKIP(2) EDIT(0,(X(I) DO I=1 TO N))
 (F(5),(N)(COL(6),( 8)F(13,6)));
 PUT EDIT(NORMR)( COL(113),F(15,6));
LAB1: FORMAT((N)(SKIP,( 8)(F(15,6))));
 LAB2: FORMAT(COL(50),A,SKIP,(N)(SKIP,( 8)F(15,6)));

LOOP:
 IF STEP > STEPLT THEN
     DO; PUT SKIP EDIT('STEP LIMIT WAS REACHED.')(A);
         GO TO EXIT;
     END;
 P=P+2;
 IF P>N THEN
     DO; IF QUIT=1 THEN
         DO; PUT SKIP EDIT('TOLLERANCE LIMIT WAS REACHED')
             (A); GO TO EXIT;
         END;
         /* THE END OF A CYCLE HAS OCCURED BUT          */
         /* MORE PROCESSING REMAINES                    */
         IF RD>0 THEN
         DO; RDCT=RDCT+1; IF RDCT=RD THEN
             DO; RDCT=0;
                 /*  PREVENT ROUND OFF                  */
```

```
                    /*  ERROR PROPAGATION                          */
                    Y=Y+X;
                    DO I= 1 TO N;
                        RA(I)=BA(I)-SUM(Y*AA(*,I));
                    END;
                    X=0;
                    /*  RECALCULATE G                              */
                    G(N+1)=0; K=0;
                    DO I= 1 TO N; IF I=N & K=0 THEN
                        DO; G(I)=(RA(I)*AA(I-1,I)
                            -RA(I-1)*AA(I,I))/T(I);
                            G(I+1)=(RA(I-1)*AA(I,I-1)
                            -RA(I)*AA(I-1,I-1))/T(I);
                        END; ELSE
                        DO; IF K=0 THEN
                            DO; K=1;
                                G(I)=(RA(I+1)*AA(I,I+1)
                                -RA(I)*AA(I+1,I+1))/T(I);
                            END; ELSE
                            DO; K=0;
                                G(I)=(RA(I-1)*AA(I,I-1)
                                -RA(I)*AA(I-1,I-1))/T(I);
                            END;
                        END;
                    END;
                END;
            END;
        QUIT=1; P=-1; GO TO LOOP;
        END;
    STEP=STEP+1;
    IF P=N THEN W=P-1; ELSE W=P;
    D1=X(W); D2=X(W+1);
    X(W)=G(P); X(W+1)=G(P+1);
    DO K=1 TO N;    IF K=W | K=W+1 THEN GO TO BOT;
        XX=X(K);    IF XX=0        THEN GO TO BOT;
        X(W)=X(W)+XX*C(P,K);
        X(W+1)=X(W+1)+XX*C(P+1,K);
    BOT: END;
    IF QUIT=1 THEN IF ABS(D1-X(W)) > TOLER
                   THEN QUIT=0;
                   ELSE IF ABS(D2-X(W+1)) > TOLER THEN QUIT=0;
    IF PRT=0 THEN GO TO LOOP;

    DO I= 1 TO N; R(I)=B(I)-SUM(A(I,*)*X); END;
    NORMR= SUM(R*R);
    PUT SKIP EDIT(STEP,(X(I) DO I= 1 TO N))
    (F(5),(N) (COL(6),( 8)F(13,6)));
    PUT EDIT(NORMR,W) (COL(114),F(15,6),COL(130),F(3));
    GO TO LOOP;
EXIT:
```

```
 IF RD¬=0 THEN X=X+Y;
 PUT SKIP(2) EDIT('NUMBER OF STEPS = ',STEP,
 'THE SOLUTION FOLLOWS',(X(I) DO I= 1 TO N))
 (A,F(6),SKIP,COL(50),A,(N)(SKIP,(10)F(12,6)));
END NEW2;
```

```
NEW3:  PROC          OPTIONS(MAIN);


 GET LIST(TOLER,PRT,STEPLT);
 DCL P FIXED BIN;   P=-1;
 /*  P IS THE BEGINNING OF THE TRIPLE   */
 DCL PRT FIXED BIN,
   STEPLT FIXED BIN;
 DCL W FIXED BIN;
 DCL QUIT FIXED BIN;
 QUIT=1;
 DCL (D1,D2,D3,G1,G2,G3,H1,H2,H3)  FLOAT DEC(16);
 DCL (X(*),S(*),T(*),D(*),B(*),A(*,*),
     AA(*,*),R(*),RA(*),COS(*))
     CTL FLOAT DEC(16);
 DCL (NORMR,TOLER) FLOAT DEC(16);
 DCL(I,J,N,STEP) FIXED BIN;
 GET LIST(N);
 ALLOCATE X(N),          B(N),A(N,N),AA(N,N),R(N),RA(N),
 S(N+2),T(N+2),
      D(N+2), COS(N+2)   ;

 /*  READ IN A, B  */   GET LIST(A,B);
 /*  CALCULATE R,RA,AA,COS,D,S,T  IN THAT ORDER  */
 STEP=0;  X=0;  R=B;
 IF PRT=1 THEN DO; D=0; COS=0; S=0; T=0; END;

 DO I=1 TO N;  RA(I) = SUM(R*A(*,I));
   DO J= I TO N;
     AA(I,J) = SUM(A(*,I) * A(*,J));      AA(J,I) = AA(I,J);
 END;   END;

 DO I=1 TO N  BY 3;
 IF I=N-1  THEN P=I-1;  /* N IS TWO OVER A MULT. OF 3  */
 ELSE IF I=N THEN P=I-2;  /*  N IS ONE OVER A MULT. OF 3  */
 ELSE P=I;
 /*  THE TRIPLE IS ELEMENTS P,P+1,P+2  WHOSE COS VALUES */
 /*GO INTO COS(I),COS(I+1),COS(I+2)  AND WHOSE D VALUE  */
 /*GOES INTO D(I)                                      */
 /*COS(I)  GETS COS 1,2                                */
 /*COS(I+1) GETS COS 1,3                               */
 /*COS(I+2) GETS COS 2,3                               */
   COS(I) = AA(P,P+1) / SQRT( AA(P,P) * AA(P+1,P+1));
 COS(I+1) = AA(P,P+2) / SQRT(AA(P,P)*AA(P+2,P+2));
 COS(I+2) = AA(P+1,P+2) / SQRT(AA(P+1,P+1)* AA(P+2,P+2));

 D(I) = 1+2*COS(I)*COS(I+1)*COS(I+2) - COS(I)*COS(I)
     - COS(I+1)*COS(I+1) - COS(I+2)*COS(I+2);

 S(I) = (1- COS(I+2)*COS(I+2) )/  AA(P,P);
```

```
   S(I+1) = (1 - COS(I+1)*COS(I+1))/ AA(P+1,P+1);
   S(I+2) =(1- COS(I)*COS(I))/ AA(P+2,P+2);

   T(I) =(COS(I+1)*COS(I+2)-COS(I))/SQRT(AA(P,P)*AA(P+1,P+1));
   T(I+1) = (COS(I)*COS(I+2)-COS(I+1))/ SQRT(AA(P,P)
                                          * AA(P+2,P+2));
   T(I+2) = (COS(I)*COS(I+1)-COS(I+2))/  SQRT(AA(P+1,P+1)
                                           * AA(P+2,P+2));
 END;

  W=-2;
 /* PRINT EVERYTHING CALCULATED AS FIXED OVERHEAD IF PRT=1;  */
  IF PRT=0  THEN GO TO LOOP;

  PUT SKIP(3) EDIT
                ('X','B','R',(X(I),B(I),R(I) DO I=1 TO N))
  (COL(10),A,COL(30),A,COL(50),A,
  (N) (SKIP,F(15,6),COL(20),F(15,6),COL(40),F(15,6)));
  PUT SKIP (2) EDIT('A') (COL(50),A);
  PUT EDIT(A) (SKIP,(N) F(13,6));
  PUT SKIP(2) EDIT('AA') (COL(50),A);
  PUT EDIT(AA) (SKIP,(N) F(13,6));
  PUT SKIP(2) EDIT('RA') (COL(50),A);
  PUT EDIT(RA) (SKIP,(N)F(13,6));
  PUT SKIP(2) EDIT('COS') (COL(50),A);
  PUT EDIT(COS) (SKIP,(N+2)F(13,6));
  PUT SKIP(2) EDIT('D') (COL(50),A);
  PUT EDIT(D) (SKIP,(N+2)F(13,6));
  PUT SKIP(2) EDIT('S') (COL(50),A);
  PUT EDIT(S)  (SKIP,(N)F(13,6));
  PUT SKIP(2) EDIT('T') (COL(50),A);
  PUT EDIT(T) (SKIP,(N)F(13,6));

  NORMR= SUM(R*R);


  PUT PAGE EDIT('STEP','X','NORM R','GROUP','R','RA')
  (A,COL(40),A,COL(119),A,COL(128),A,SKIP,
  COL(40),A,COL(40),A);
  PUT SKIP(2) EDIT(0,(X(I) DO I=1 TO N), NORMR,
  (R(I) DO I=1 TO N),(RA(I) DO I=1 TO N))
  (F(4),      (N) (F(11,6)), COL(115), F(11,6),SKIP,
  COL(5),(N) F(11,6),COL(5),(N)F(11,6));


LOOP:
  IF STEP > STEPLT  THEN  DO;
    PUT SKIP(2) EDIT('STEP LIMIT WAS REACHED')(A);
  GO TO EXIT;  END;
```

```
    W=W+3;
    IF W>N  THEN DO;
      IF QUIT=1   THEN DO;
    PUT SKIP(2) EDIT('TOLLERANCE LIMIT HAS BEEN REACHED')(A);
      GO TO EXIT;   END;
    QUIT=1;  W= -2;  GO TO LOOP;  END;

      /*  THIS TRIPLE IS ELEMENTS  P,P+1,P+2  IN X  AND ITS INFO
    IS IN COS AND  D  AS ELEMENTS  W,W+1,W+2   */

    IF W=N-1  THEN P=W-1;  ELSE IF W=N THEN P=W-2; ELSE P=W;

    STEP= STEP+1;
    D1= X(P);  D2=X(P+1);  D3=X(P+2);
    G1,G2,G3=0;
    DO I= 1 TO N;
      IF I=P | I=P+1 | I=P+2  THEN GO TO STP;
    G1=G1+ AA(I,P)*X(I);
    G2=G2+ AA(I,P+1)*X(I);
    G3=G3+ AA(I,P+2)*X(I);
   STP: END;

    H1= RA(P) -G1;
    H2= RA(P+1) - G2;
    H3= RA(P+2) -G3;

    X(P) =(H1*S(W) + H2*T(W) + H3*T(W+1)) / D(W);
    X(P+1) =(H1*T(W) + H2*S(W+1) +H3* T(W+2))/ D(W);
    X(P+2) = (H1*T(W+1) +H2 * T(W+2) + H3 * S(W+2)) / D(W);

    IF QUIT=1  THEN IF ABS(D1-X(P)) > TOLER   THEN QUIT=0;
      ELSE IF ABS(D2-X(P+1)) > TOLER   THEN QUIT=0;
      ELSE IF ABS(D3-X(P+2)) > TOLER   THEN QUIT=0;
    IF  PRT=0   THEN GO TO LOOP;
    DO I=1 TO N;   R(I) = B(I) -SUM(A(I,*)*X);  END;
    NORMR= SUM(R*R);
    PUT SKIP EDIT(STEP,(X(I) DO I= 1 TO N),NORMR,   P)
    (F(4), (N) F(11,6),COL(115),F(13,8),F(3));
    IF PRT=0  THEN RETURN;
    GO TO LOOP;

  EXIT:   PUT SKIP EDIT('NUMBER OF STEPS= ',STEP,
   'THE SOLUTION FOLLOWS',(X(I) DO I= 1 TO N))
   (A,F(6),SKIP,COL(50),A,(N)(SKIP,(10)F(12,6)));

   END NEW3;
```

```
HYB4: PROCEDURE          OPTIONS(MAIN);
 GET LIST(TOLER,PRT,STEPLT);
 DCL QUIT FIXED BIN;   QUIT=1;
 DCL QR FIXED BIN;    QR= -3;
   /*    QR    IS THE BEGINNING OF THE QUADRUPLE   */
 DCL PRT FIXED BIN,
   STEPLT FIXED BIN, STEP FIXED BIN;
 DCL (X(*),C(*,*),B(*),A(*,*),R(*),AA(*,*),AL(*),T(*))
   CTL FLOAT DEC (16);
 DCL(P(*),Q(*)    )    CTL FLOAT DEC(16);
 DCL(QJ,PI,QL,PK)  FLOAT DEC(16);
 DCL(DI,DJ,DK,DL)   FLOAT DEC(16);
 DCL(ZI,ZJ,ZK,ZL, SI,SJ,SK,SL)   FLOAT DEC(16);
 DCL RA(*) FLOAT DEC(16) CTL;
 DCL (NORMR,TOLER) FLOAT DEC(16);
 DCL(I,J,K,L,M,W,N) FIXED BIN;
 GET LIST(N);
 ALLOCATE X(N),C(N+3,N  ),
                                      B(N),A(N,N),R(N),AA(N,N);
 ALLOCATE RA(N),AL(N+3),T(N+3);
 ALLOCATE  P(N+3), Q(N+3);
 /*  READ IN  A,B    */
 GET LIST(A,B);
 /*   CALCULATE   C,AL,AA,R,RA,T  THEN PRINT ALL VALUES   */
 STEP=0;    X=0;
 R=B;
 DO I=1 TO N;  RA(I) = SUM(R*A(*,I));
   DO J= I TO N;
     AA(I,J) = SUM(A(*,I) * A(*,J));      AA(J,I) = AA(I,J);
 END;    END;


 IF PRT¬=0  THEN DO;    C(N+1,*)=0;    C(N+2,*)=0;
                 C(N+3,*)=0;   T(N+1)=0;   T(N+2)=0;
                 T(N+3)=0;   AL(N+1)=0;   AL(N+2)=0;
                 AL(N+3)=0;   END;
 /*    CALCULATE T IN GROUPS OF 4   */
      DO    I=1 TO N BY 4;  IF I+3>N THEN W=N-3; ELSE W=I;
 /*    I IS THE INDEX TO T;W IS THE INDEX TO AA    */
 T(I) = AA(W,W+1)*AA(W,W+1)-AA(W,W)*AA(W+1,W+1);
 T(I+1)=T(I);    W= W+2;
 T(I+2)=AA(W,W+1)*AA(W,W+1)-AA(W,W)*AA(W+1,W+1);
 T(I+3)=T(I+2);
 END;

 /*  CALCULATE AL  IN GROUPS OF 4     */
 DO I=1 TO N BY 4;    IF I+3>N THEN W=N-3; ELSE W=I;
   /*    I IS THE INDEX TO AL,T ;  W IS THE INDEX TO RA,AA*/
 AL(I)   =(RA(W+1)*AA(W,W+1)-RA(W)*AA(W+1,W+1))/T(I);
 W=W+1;
```

```
AL(I+1)=(RA(W-1)*AA(W,W-1)-RA(W)*AA(W-1,W-1))/T(I+1);
W=W+1;
AL(I+2) =(RA(W+1)*AA(W,W+1)-RA(W)*AA(W+1,W+1))/T(I+2);
W=W+1;
AL(I+3) =(RA(W-1)*AA(W,W-1)-RA(W)*AA(W-1,W-1))/T(I+3);
END;

/*  CALCULATE  C  IN GROUPS OF 4   */
DO I=1 TO N BY 4;   IF I+3>N THEN W=N-3;  ELSE W=I;
/*   I  IS THE INDEX TO C,T;  W IS THE INSEX TO AA   */
DO K=1 TO N;
C(I  ,K)=(AA(W  ,K)*AA(W+1,W+1)-AA(W+1,K)*AA(W  ,W+1))
/ T(I  );
C(I+1,K)=(AA(W+1,K)*AA(W  ,W  )-AA(W  ,K)*AA(W+1,W  ))
/ T(I+1);
C(I+2,K)=(AA(W+2,K)*AA(W+3,W+3)-AA(W+3,K)*AA(W+2,W+3))
/ T(I+2);
C(I+3,K)=(AA(W+3,K)*AA(W+2,W+2)-AA(W+2,K)*AA(W+3,W+2))
/ T(I+3);
 END;
END;

IF PRT=1 THEN DO; DO I=N TO N+3; P(I)=0; Q(I)=0; END; END;
 /*   CALCULATE P  AND Q   */
DO M=1 TO N BY 4;
IF M+3>N  THEN I= N-3;  ELSE I=M;
                J= I+1; K= I+2; L= I+3;
DCL(E,F,G) FIXED BIN;
E= M+1;  F= M+2;  G= M+3;
    P(M)= C(M,K) * C(F,I) + C(M,L) * C(G,I);
    Q(M) = C(M,K) * C(F,J) + C(M,L) * C(G,J);
 P(M+1)  = C(E,K) * C(F,I) + C(E,L) * C(G,I);
Q(M+1)   = C(E,K) * C(F,J) + C(E,L) * C(G,J);
P(M+2)   = C(F,I) * C(M,K) + C(F,J) * C(E,K);
Q(M+2)   = C(F,I) * C(M,L) + C(F,J) * C(E,L);
P(M+3)   = C(G,I) * C(M,K) + C(G,J) * C(E,K);
Q(M+3)   = C(G,I) * C(M,L) + C(G,J) * C(E,L);
END;

IF PRT=0  THEN GO TO LOOP;

PUT SKIP(3) EDIT
             ('X','B','R',(X(I),B(I),R(I) DO I=1 TO N))
(COL(10),A,COL(30),A,COL(50),A,
(N) (SKIP,F(15,6),COL(20),F(15,6),COL(40),F(15,6)));
PUT SKIP(2) EDIT('C MATRIX')(COL(50),A);
PUT EDIT(C  ) (SKIP,(N)F(13,6));
PUT SKIP (2) EDIT('A') (COL(50),A);
PUT EDIT(A) (SKIP,(N) F(13,6));
PUT SKIP(2) EDIT('AA') (COL(50),A);
```

```
  PUT EDIT(AA) (SKIP,(N) F(13,6));
  /*   PRINT AL RA T   */
  PUT SKIP(2)EDIT('AL') (COL(50),A);
 PUT EDIT(AL)(SKIP,(N+1)F(13,6));
  PUT SKIP(2) EDIT('RA') (COL(50),A);
  PUT EDIT(RA) (SKIP,(N)F(13,6));
  PUT SKIP(2) EDIT('T') (COL(50),A);
   PUT EDIT(T)  (SKIP,(N+1)F(13,6));
  /*   PRINT P,Q  */
  PUT SKIP(2) EDIT('P') (COL(50),A);
  PUT EDIT(P) (SKIP,(N+3)F(13,6));
  PUT SKIP(2) EDIT('Q')(COL(50),A);
  PUT EDIT(Q) (SKIP,(N+3)F(13,6));

  PUT PAGE EDIT('STEP','X','NORM R','PAIR','R','RA')
  (A,COL(40),A,COL(119),A,COL(128),A,SKIP,
  COL(40),A,COL(40),A);
  PUT SKIP(2) EDIT(0,(X(I) DO I=1 TO N), NORMR,
  (R(I) DO I=1 TO N),(RA(I) DO I=1 TO N))
  (F(4),      (N) (F(11,6)), COL(115), F(11,6),SKIP,
  COL(5),(N) F(11,6),COL(5),(N)F(11,6));


LOOP:
  IF STEP > STEPLT  THEN DO;
 DO I= 1 TO N; R(I)=B(I)-SUM(A(I,*)*X); END;
 NORMR= SUM(R*R);
  PUT SKIP EDIT('STEP LIMIT HAS BEEN REACHED.  NORM R= ',
            NORMR) (A,F(15,6));
  PRT=0;  GO TO LINE;   END;


  QR= QR+4;   /*   QR,QR+1,QR+2,QR+3  ARE THE QUADRUPLE   */

  IF QR>N  THEN DO;   IF QUIT=1  THEN DO;
  PUT SKIP EDIT('TOLLERANCE LIMIT HAS BEEN REACHED')(A);
  PRT=0;   GO TO LINE;    END;
  QUIT=1;  QR=1;  END;

  STEP= STEP+1;
  /*   QR IS THE BEGINNING OF THE QUADRUPLE IN THE P,Q   */
  /* ARRAYS AND W IS THE BEGINNING OF THE QUADRUPLE IN X */
  IF QR+3 >N  THEN W= N-3;   ELSE W= QR;
  DI=X(W);  DJ= X(W+1);  DK= X(W+2);  DL= X(W+3);
  /*   CALCULATE THE Z'S   */
  ZI,ZJ,ZK,ZL=0;
  I=QR;
  J= I+1;   K= I+2;   L= I+3;

  DO M=1 TO N;   IF M>=W & M<W+4  THEN GO TO STP1;
```

```
   ZI= ZI+ X(M) * C(I,M);
   ZJ= ZJ + X(M) * C(J,M);
   ZK= ZK+ X(M) * C(K,M);
   ZL= ZL+ X(M) * C(L,M);
STP1:   END;

   /*   CALCULATE THE S  VALUES   */
   SI= C(I,W+2) *(AL(K) + ZK) + C(I,W+3) *(AL(L)+ZL)
       + ZI + AL(I);
   SJ= C(J,W+2) * (AL(K)+ZK) + C(J,W+3)*(AL(L)+ZL)
       + ZJ + AL(J);
   SK= C(K,W) *(AL(I) + ZI) + C(K,W+1) *(AL(J) + ZJ)
       + ZK + AL(K);
   SL= C(L,W) *(AL(I) + ZI) + C(L,W+1) *(AL(J) + ZJ)
       + ZL + AL(L);
   /*   CLACULATE   THE NEW X'S   */
   QJ= Q(J)-1; PI= P(I)-1; QL= Q(L)-1;  PK= P(K) -1;
   X(W) =(QJ * SI - SJ *Q(I))/
         (P(J) * Q(I) -  PI * QJ);
   X(W+1)= (P(J) * SI - SJ *  PI)/
   (QJ * PI - Q(I) * P(J));
   X(W+2)= ( QL       * SK - SL * Q(K)) /
        (P(L) * Q(K) -  PK * QL);
   X(W+3) = (P(L) *SK - SL* PK)
        /  (  QL  *  PK          - Q(K) * P(L));

   IF QUIT=1  THEN IF ABS(DI-X(W))> TOLER  THEN QUIT=0;
   ELSE IF ABS(DJ-X(W+1))> TOLER  THEN QUIT=0;
   ELSE IF ABS(DK-X(W+2)) >  TOLER  THEN QUIT=0;
   ELSE IF ABS(DL-X(W+3)) >  TOLER  THEN QUIT=0;

   IF  PRT=0   THEN GO TO LOOP;
LINE:
   PUT SKIP(2)
            EDIT(STEP,(X(I) DO I= 1 TO N),W)
   (F(4), (N) F(11,6),COL(115),F(3));
   IF PRT=0  THEN RETURN;
   GO TO LOOP;

   END HYB4;
```