Comp. & Maths. with Appls. Vol. 7, pp. 211–223, 1981
Printed in Great Britain.

0097–4943/81/030211–13$02.00/0
Pergamon Press Ltd.

THREE DIMENSIONAL x-PROJECTION METHOD (WITH ACCELERATION TECHNIQUES) FOR SOLVING SYSTEMS OF LINEAR EQUATIONS

ROGER L. WAINWRIGHT
Division of Mathematical Sciences, The University of Tulsa, Tulsa, OK 74104, U.S.A.

Communicated by E. Y. Rodin

(*Received January* 1980 *and, in revised form, April* 1980)

Abstract—The solution of linear systems of equations using a 3-dimensional x-projection method is presented. At each step of the iterative process the approximate solution vector is projected to a point in the intersection of three of the hyperplanes of the linear system. Nonsingularity of the coefficient matrix is the only requirement for convergence. An algorithm is presented to select triples of hyperplanes to project the approximate solution vector at each step. The algorithm is quasi-optimal since the hyperplanes, which are determined by the row vectors of the coefficient matrix, are selected *a priori*. This is shown to significantly reduce the number of cycles required for convergence. We observe that in some cases the ratio of the change vectors of the approximate solution vectors after some number of cycles becomes a constant. Thus, when this occurs a simple geometric acceleration can be applied to calculate the solution directly. Geometric acceleration can significantly reduce computation time and improve the accuracy of the solution by orders of magnitude. The 3-dimensional x-projection method was tested against the 2-dimensional x-projection method using random and Hilbert coefficient matrices and proved superior (less C.P.U. time required) in nearly every case.

1. INTRODUCTION

The purpose of this paper is to develop a 3-dimensional x-projection method for solving the equation

$$Ax = b \tag{1.1}$$

where A is a nonsingular matrix of order n and x and b are n-vectors. In this paper we also develop an acceleration technique for the 3-dimensional x-projection method. The paper is divided into six sections. In Section 2, the basic concepts of x-projection methods for solving systems of linear equations is reviewed. In Section 3, a 3-dimensional x-projection method is developed and in Section 4 a hyperplane selection criteria is developed to accelerate the rate of convergence of the 3-dimensional x-projection method. Test cases and comparisons are presented in Section 5. Concluding remarks are presented in Section 6 followed by a program listing of the 3-dimensional x-projection method using a quasi-optimal hyperplane selection criterion and geometric acceleration.

2. BASIC CONCEPTS OF x-PROJECTION METHODS

Wainwright[1] gives a detailed development of the 1-dimensional and 2-dimensional x-projection methods for solving systems of linear equations. To review briefly, the methods are as follows:

(*a*) 1-*Dimensional x-projection method*

Each equation of (1.1) (i.e. $(a^i, x) = b_i$ where a^i is the ith row of A) geometrically defines an $(n-1)$-dimensional hyperplane called the ith hyperplane which is orthogonal to a^i. (a^i, x) denotes the inner product of the two vectors, a^i and x. The solution vector is the point in which the n hyperplanes of (1.1) intersect. The iterative process begins with an arbitrary point, x^0, then moves it onto hyperplane 1 in a direction orthogonal to it. It then moves to hyperplane 2 in a direction orthogonal to it, and so on to hyperplane n in a direction orthogonal to it. Then the process starts over by moving to hyperplane 1 in a direction orthogonal to it, to hyperplane 2 in a direction orthogonal to it and so on. Pizer[2] proves that this process always converges for arbitrary x^0 assuming the non-singularity of A. To save time in the iteration, each equation of (1.1) is scaled according to the Euclidean norm so that $(a^i, a^i) = 1$. Thus the $(k+1)$th step of this method becomes

$$x^{k+1} = x^k + r_i^k a^i \tag{2.1}$$

where $r^k = b - Ax^k$ is the residual vector and $r_i^k = (b_i - (a^i, x^k))a^i$ where i is the residue of $k+1$ modulo n. This is a total step method in that every component of the approximate solution vector is altered at each step. Wainwright[1] calls this a 1-dimensional x-projection method because at each iterative step the approximate solution vector, x^k, is projected onto one $(n-1)$-dimensional hyperplane. Other methods of projection which project the residual vector, r^k, at each step are called r-projection methods. The 1-dimensional r-projection method was first developed by Garza[3] and later rediscovered by Keller[4]. [5, 6] also give discussions of the 1-dimensional r-projection method. Discussions concerning 2 and 3-dimensional r-projection methods can be found in [7–10]. The best overall discussion of r-projection methods can be found in [11].

(b) 2-Dimensional x-projection method

For the 2-dimensional x-projection method the iterative process begins with an arbitrary point, x^0, then moves it onto the intersection of hyperplanes 1 and 2 in a direction orthogonal to both of them. It then moves to the intersection of hyperplanes 3 and 4 in a direction orthogonal to both of them, and so on to the intersection of hyperplanes $n-1$ and n in a direction orthogonal to both of them. Then the process starts over by moving to the intersection of hyperplanes 1 and 2 in a direction orthogonal to both of them, then to hyperplanes 3 and 4 in a direction orthogonal to both of them and so on. More precisely, if n is even the hyperplanes are paired $(1, 2)$, $(3, 4), \ldots, (n-1, n)$ and if n is odd the hyperplanes are paired $(1, 2)$, $(3, 4), \ldots, (n-2, n-1), (n-1, n)$. Thus at each iterative step the approximate solution vector is projected onto two $(n-2)$-dimensional hyperplanes. The 2-dimensional x-projection method at the $k+1$ interaction step projecting the approximate solution vector onto hyperplanes i and j $(1 \le i, j \le n; i \ne j)$ is defined by

$$x^{k+1} = x^k + \alpha a^i + \beta a^j \tag{2.2}$$

where

$$\alpha = \frac{[(a^j, x^k) - b_j](a^i, a^j) - [(a^i, x^k) - b_i]}{1 - (a^i, a^j)^2}$$

$$\beta = \frac{[(a^i, x^k) - b_i](a^i, a^j) - [(a^j, x^k) - b_j]}{1 - (a^i, a^j)^2}. \tag{2.3}$$

Wainwright[1] proves that the 2-dimensional x-projection method converges for arbitrary x^0 assuming the nonsingularity of A. He also develops an acceleration technique for increasing the rate of convergence by appropriate hyperplane pair selections.

3. 3-DIMENSIONAL x-PROJECTION METHOD

Consider the following nonstationary iterative scheme for solving (1.1) that makes successive changes to the components of a starting vector, x^0

$$x^{k+1} = x^k + \alpha_k \omega^k + \beta_k \gamma^k + \lambda_k \sigma^k$$

where α_k, β_k and λ_k are scalars and ω^k, γ^k and σ^k are n-vectors. For the 3-dimensional x-projection method the iterative process begins with an arbitrary point, x^0, then moves it onto the intersection of hyperplanes 1, 2 and 3 in a direction orthogonal to all of them. It then moves to the intersection of hyperplanes 4, 5 and 6 in a direction orthogonal to all of them, and so on to the intersection of hyperplanes $n-2$, $n-1$ and n in a direction orthogonal to all of them. Then the process starts over by moving to the intersection of hyperplanes 1, 2 and 3 in a direction orthogonal to all of them and so on. If n modulo $3 = 1$, then the hyperplanes are grouped $(1, 2, 3), (4, 5, 6), \ldots, (n-3, n-2, n-1), (n-2, n-1, n)$ and if n modulo $3 = 2$ then the hyperplanes are grouped $(1, 2, 3), (4, 5, 6), \ldots, (n-4, n-3, n-2), (n-2, n-1, n)$. Let x^k be a vector to a point in the intersection of hyperplanes i, $i+1$ and $i+2$. Let x^{k+1} be the vector to the point in the intersection of hyperplanes $i+3$, $i+4$ and $i+5$ which is reached by moving from x^k in a direction orthogonal to all three hyperplanes, $i+3$, $i+4$ and $i+5$. Let y be any vector to the point in the intersection of hyperplanes $i+3$, $i+4$ and $i+5$ ($y \neq x^{k+1}$). Then the change to the approximate solution vector, $x^{k+1} - x^k$, will be orthogonal to $x^{k+1} - y$, which lies in the intersection of hyperplanes $i+3$, $i+4$ and $i+5$:

$$((x^{k+1} - x^k), (x^{k+1} - y)) = 0. \tag{3.1}$$

Since both x^{k+1} and y define points in the intersection of hyperplanes $i+3, i+4$ and $i+5$, both satisfy the $(i+3)$th, $(i+4)$th and $(i+5)$th equations of (1.1):

$$(a^{i+3}, x^{k+1}) = b_{i+3}; (a^{i+3}, y) = b_{i+3} \tag{3.2}$$

$$(a^{i+4}, x^{k+1}) = b_{i+4}; (a^{i+4}, y) = b_{i+4} \tag{3.3}$$

$$(a^{i+5}, x^{k+1}) = b_{i+5}; (a^{i+5}, y) = b_{i+5} \tag{3.4}$$

thus

$$\begin{aligned} (a^{i+3}, (x^{k+1} - y)) &= 0 \\ (a^{i+4}, (x^{k+1} - y)) &= 0 \\ (a^{i+5}, (x^{k+1} - y)) &= 0. \end{aligned} \tag{3.5}$$

Therefore, a linear combination of a^{i+3}, a^{i+4} and a^{i+5} will be orthogonal to $x^{k+1} - y$ for any y to the point in the interssction of hyperplanes $i+3$, $i+4$ and $i+5$ (y can range over $n-3$ dimensions). Thus it follows from (3.1) and (3.5) that a linear combination of a^{i+3}, a^{i+4} and a^{i+5} will be in the same direction as $x^{k+1} - x^k$, so for scalars α, β and λ

$$x^{k+1} = x^k + \alpha a^{i+3} + \beta a^{i+4} + \lambda a^{i+5}. \tag{3.6}$$

By taking the inner products of a^{i+3} with (3.6), a^{i+4} with (3.6) and a^{i+5} with (3.6) and using equations (3.2), (3.3) and (3.4) we obtain the following symmetric linear system of three equations and three unknowns:

$$\begin{aligned} \alpha(a^{i+3}, a^{i+3}) + \beta(a^{i+3}, a^{i+4}) + \lambda(a^{i+3}, a^{i+5}) &= b_{i+3} - (a^{i+3}, x^k) \\ \alpha(a^{i+4}, a^{i+3}) + \beta(a^{i+4}, a^{i+4}) + \lambda(a^{i+4}, a^{i+5}) &= b_{i+4} - (a^{i+4}, x^k) \\ \alpha(a^{i+5}, a^{i+3}) + \beta(a^{i+5}, a^{i+4}) + \lambda(a^{i+5}, a^{i+5}) &= b_{i+5} - (a^{i+5}, x^k). \end{aligned} \tag{3.7}$$

If we scale each equation of (1.1) according to the Euclidean norm so that $(a^j, a^j) = 1$, $1 \leq j \leq n$, then solving for α, β and λ we get

$$\begin{aligned} \alpha &= \frac{1}{D}\{r_{i+3}[1 - (a^{i+4}, a^{i+5})^2] + r_{i+4}[(a^{i+3}, a^{i+5})(a^{i+4}, a^{i+5}) - (a^{i+3}, a^{i+4})] \\ &\quad + r_{i+5}[(a^{i+3}, a^{i+4})(a^{i+4}, a^{i+5}) - (a^{i+3}, a^{i+5})]\} \\ \beta &= \frac{1}{D}\{r_{i+3}[(a^{i+3}, a^{i+5})(a^{i+4}, a^{i+5}) - (a^{i+3}, a^{i+4})] + r_{i+4}[1 - (a^{i+3}, a^{i+5})^2] \\ &\quad + r_{i+5}[(a^{i+3}, a^{i+4})(a^{i+3}, a^{i+5}) - (a^{i+4}, a^{i+5})]\} \\ \lambda &= \frac{1}{D}\{r_{i+3}[(a^{i+3}, a^{i+4})(a^{i+4}, a^{i+5}) - (a^{i+3}, a^{i+5})] + r_{i+4}[(a^{i+3}, a^{i+4})(a^{i+3}, a^{i+5}) - (a^{i+4}, a^{i+5})] \\ &\quad + r_{i+5}[1 - (a^{i+3}, a^{i+4})^2]\} \end{aligned} \tag{3.8}$$

where D is the determinant of the coefficient matrix in (3.7).

$$D = 1 + 2(a^{i+4}, a^{i+5})(a^{i+3}, a^{i+5})(a^{i+3}, a^{i+4}) - (a^{i+3}, a^{i+5})^2 - (a^{i+4}, a^{i+5})^2 - (a^{i+3}, a^{i+4})^2. \quad (3.9)$$

Thus the 3-dimensional x-projection method is defined by (3.6) where α, β and λ are defined in (3.8).

Theorem 3.1

A single step of a 3-dimensional x-projection method projecting x^k, which defines a point in the intersection of hyperplanes $i-3$, $i-2$ and $i-1$, to x^{k+1} which defines a point in the intersection of hyperplanes i, $i+1$ and $i+2$ in a direction orthogonal to hyperplanes i, $i+1$ and $i+2$, is equivalent to repeated 1-dimensional x-projection steps projecting the approximate solution vector back and forth to points in the ith, $(i+1)$th and $(i+2)$th hyperplanes until the change between two successive approximate solution vectors is zero.

The proof of this theorem follows the same pattern and form as the proof in the 2-dimensional case given in [1] only extended to three dimensions. The proof while not difficult is long and tedious and will not be presented in this paper.

Theorem 3.2

The 3-dimensional x-projection method for solving linear systems is convergent provided the coefficient matrix is nonsingular.

Proof. The proof follows immediately from Theorem 3.1. The 3-dimensional x-projection method is based on repeated applications of the 1-dimensional x-projection method which is always convergent. The Pythagorean theorem guarantees that we move closer to the intersection of three hyperplanes as we project back and forth between them in directions orthogonal to them. In fact it can be shown that we move closer to the intersection of all of the hyperplanes[2].

4. ACCELERATION TECHNIQUES FOR THE 3-DIMENSIONAL x-PROJECTION METHOD

(a) *Selection of projection hyperplanes*

The x-projection class of methods is typical of most projection methods in that the rate of convergence is very slow. One would like to be able to select *a priori*, a sequence of triplets of projection hyperplanes which guarantees most rapid convergence for a system of linear equations. One criterion for selecting an optimal sequence of projections is to observe how the residual at any step is related to the hyperplanes determined by three of the rows of A. In particular one would like to reduce the length of the residual vector as much as possible. Let x^k be a vector to a point in the intersection of hyperplanes f, g and h and we move x^k to x^{k+1} along a direction orthogonal to hyperplanes i, j and p using a 3-dimensional x-projection method. Thus

$$x^{k+1} = x^k + \alpha a^i + \beta a^j + \lambda a^p$$

and

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

or equivalently

$$(r^k, r^k) - (r^{k+1}, r^{k+1}) = 2\alpha(r^k, Aa^i) + 2\beta(r^k, Aa^j) + 2\lambda(r^k, Aa^p) - 2\alpha\beta(Aa^i, Aa^j) - 2\alpha\lambda(Aa^i, Aa^p) - 2\beta\lambda(Aa^j, Aa^p) - \alpha^2(Aa^i, Aa^i) - \beta^2(Aa^j, Aa^j) - \lambda^2(Aa^p, Aa^p).$$

Substituting (3.8) for α, β and λ and $\cos\theta_{ij}$ for (a^i, a^j) (since $(a^i, a^i) = (a^j, a^j) = 1$), we get

$$(r^k, r^k) - (r^{k+1}, r^{k+1}) = \frac{1}{D^{ijp}}\left[R^{ijp} - \frac{S^{ijp}}{D^{ijp}}\right]$$

where

$$R^{ijp} = 2(r^k, Aa^i)W_i + 2(r^k, Aa^j)W_j + 2(r^k, Aa^p)W_p$$

$$S^{ijp} = 2(Aa^i, Aa^j)W_iW_j + 2(Aa^i, Aa^p)W_iW_p + 2(Aa^i, Aa^p)W_jW_p + (Aa^i, Aa^i)W_i^2 + (Aa^j, Aa^j)W_j^2 + (Aa^p, Aa^p)W_p^2$$

and

$$W_i = r_i^k[1 - \cos^2\theta_{jp}] + r_j^k[\cos\theta_{ip}\cos\theta_{jp} - \cos\theta_{ij}] + r_p{}^k[\cos\theta_{ij}\cos\theta_{jp} - \cos\theta_{ip}],$$

$$W_j = r_i^k[\cos\theta_{ip}\cos\theta_{jp} - \cos\theta_{ij}] + r_j^k[1 - \cos^2\theta_{ip}] + r_p{}^k[\cos\theta_{ij}\cos\theta_{ip} - \cos\theta_{jp}],$$

$$W_p = r_i^k[\cos\theta_{ij}\cos\theta_{jp} - \cos\theta_{ip}] + r_j^k[\cos\theta_{ij}\cos\theta_{ip} - \cos\theta_{jp}] + r_p{}^k[1 - \cos^2\theta_{ij}],$$

$$D^{ijp} = 1 + 2\cos\theta_{ij}\cos\theta_{ip}\cos\theta_{jp} - \cos^2\theta_{ij} - \cos^2\theta_{ip} - \cos^2\theta_{jp}.$$

To select a single step optimal method (i.e. one which would give a minimum value for (r^{k+1}, r^{k+1})), i, j, p are chosen such that $(1/D^{ijp})[R^{ijp} - (S^{ijp}/D^{ijp})]$ is a maximum. We observe that $R^{ijp} = \underline{0}(r_i^k + r_j^k + r_p{}^k)$ and $S^{ijp} = \underline{0}(r_i^k + r_j^k + r_p{}^k)^2$. Thus S^{ijp} will approach zero at a faster rate than R^{ijp} as $r^k \to \emptyset$. ($\emptyset$ is the zero vector). In fact since $(1/D^{ijp})$ is a constant for the choice of i, j and p it dominates the product $(1/D^{ijp})[R^{ijp} - (S^{ijp}/D^{ijp})]$ as $r^k \to \emptyset$. Therefore a quasi-optimal selection for the 3-dimensional x-projection method is to select triples of rows of A in descending order of the values of $(1/D^{ijp})$ $(i, j, p = 1, 2, \ldots, n)$. All $(1/D^{ijp})$ $(i, j, p = 1, 2, \ldots, n)$ are computed initially and a stationary algorithm is obtained by selecting *a priori* triples of rows of A based on the values of $(1/D^{ijp})$ such that each row is used at least once and at most two rows are used twice in each cycle. Calculating all $(1/D^{ijp})$ values *a priori* results in $\underline{0}\ 8n^3$ operations. The most coplanar triples of rows of A results in the largest $(1/D^{ijp})$ value. Therefore another quasi-optimal selection is to select triples of A which are most coplanar. Specifically, the quasi-optimal selection process that was used in all of the test problems and comparisons and which appears in the program listing is described as follows:

First, the minimum angle is determined among the pairs of rows of A. From among the remaining rows we determine which row makes the best coplanar fit. This process is repeated again with the unused rows of A until all rows are used. Calculating the angles *a priori* results in $\underline{0}\ 2n^3$ operations which is slightly more efficient than calculating all of the $(1/D^{ijp})$ values.

(b) *Acceleration using geometric series*

Probably the most significant characteristic of the x-projection class of methods is that the ratio of the change vectors of the approximate solution vector as $r^k \to \emptyset$ converges to a constant[1]. That is, at some cycle, m, the solution can be written

$$x = x^m + \Delta x^{m+1} + \Delta x^{m+2} + \cdots, (\Delta x^{i+1} = x^{i+1} - x^i)$$

where $(\Delta x^{i+1}/\Delta x^i) = \rho$ for all $i > m$, $(\rho < 1)$ and m, $m+1$, $m+2, \ldots$ represents successive cycles of an x-projection method. Then the solution can be calculated directly:

$$x = x^m + \frac{\Delta x^{m+1}}{1-\rho}.$$

This result will not be proven. For some linear systems the iterative process converges before the mth cycle has been reached (i.e. before the value of ρ converges). Thus the geometric acceleration may not always be able to be applied.

5. TEST PROBLEMS AND COMPARISONS

In this section, four programs are compared. Each program is identified in the comparison tables by the following notation:

(1) 2-X is the 2-dimensional x-projection method with quasi-optimal hyperplane selection as described in [1].

(2) 2-XA is the 2-dimensional x-projection method with quasi-optimal hyperplane selection and if possible geometric acceleration as described in [1].

(3) 3-X is the 3-dimensional x-projection method with quasi-optimal hyperplane selection as described in Sections 3 and 4.

(4) 3-XA is the 3-dimensional x-projection method with quasi-optimal hyperplane selection and if possible, geometric acceleration as described in Sections 3 and 4.

Each method was programmed in FORTRAN and executed from a library using a Honeywell Sigma 6. All calculations were performed in double precision and each test case used an initial starting vector of zero. Any I/O that was performed was not included in the C.P.U. time.

(a) Random coefficient matrix

In test case 1 we tested the methods in the following manner. For a given dimension n, we generated a random matrix (a matrix consisting of random numbers between 0 and 1). We assumed a solution vector with components all 1's and generated an appropriate righthand side. The resulting systems were solved and the C.P.U. times and row vector groupings (BEST and WORST) were recorded. This procedure was repeated for each method for the same matrix. BEST indicates the quasi-optimal hyperplane selection was used to select the best groupings of the row vectors of A. For comparison, WORST indicates the quasi-optimal selection was used in reverse to select the worst groupings of the row vectors of A. This shows the importance of hyperplane selections. Results are displayed in Table 1. The interative process terminated when every corresponding component of two successive approximate solution vectors was within 0.000005. AVE $(1/D)$ USED indicates the average value of $(1/D^{iip})$ for the selected triples. This is indicated in both the BEST and WORST cases for comparison. TIME indicates the C.P.U. time required for convergence in seconds.

Table 1. Results for test case 1 (random matrices)

Dimen-sion	Methods							
	2-X (BEST) TIME	2-XA (BEST) TIME	3-X (BEST) TIME	3-XA (BEST) TIME	AVE $\frac{1}{D}$ USED	3-X (WORST) TIME	3-XA (WORST) TIME	AVE $\frac{1}{D}$ USED
8	9.75	2.56	9.89	1.73	43.30	23.50	2.64	16.52
12	14.29	4.40	3.03	2.71	47.36	14.14	8.00	12.03
16	52.38	13.40	29.50	13.96	27.06	38.49	34.56	6.74
20	20.89	15.26	17.52	13.19	13.48	66.35	66.55	5.30
30	322.82	106.36	155.81	97.69	13.26	354.19	131.74	6.75
40	325.78	270.89	290.62	198.02	10.90	478.49	320.186	5.13
50	458.71	459.70	236.44	237.28	10.03	528.59	530.21	5.08

(b) Ill-conditioned matrices

A classical example of ill-conditioned matrices is the set of *Hilbert matrices* H_n of order n with elements $H_n(i,j) = 1/(i+j-1)$ (see [12]). In test case 2 we compare methods 2-X, 2-XA, 3-X and 3-XA for various orders of Hilbert coefficient matrices. In each case, a solution vector with components all 1's was assumed and the appropriate righthand side was generated. Table 2 gives the results of test case 2 where C.P.U. times in seconds for various dimensions of Hilbert matrices are given. In addition, the maximum absolute difference of any component of the final approximate solution vector from 1 is given. This is used as a measure of accuracy rather than the inner product of the final residual vector. In each case, the best quasi-optimal hyperplane selection was performed (i.e. the rows were grouped consecutively $(1,2,3)$, $(4,5,6),\ldots$). The only exception was 3-X and 3-XA for $n = 30$; the rows were grouped $(1,11,21)$, $(2,12,22)\ldots(10,20,30)$. Worst case quasi-optimal hyperplane selections are not included; as one might suspect in each case the rate of convergence was extremely slow. In all cases, a tolerance of 0.000005 was used to terminate the iterative process.

Table 2. Results for test case 2 (Hilber matrices)

Dimension	Methods							
	2-X		2-XA		3-X		3-XA	
	Max Dif	Time	Max Dif	Time	Max Dif	Time	Max Dif	Time
8	.0096	7.200	.0092	0.58	.0078	0.78	.0078	0.38
12	.0191	6.07	.0191	0.84	.0255	1.98	.0256	0.50
16	.0161	56.65	.0068	15.07	.0311	51.93	.0691	5.06
20	.0128	93.48	.0097	14.95	.0107	19.49	.0100	2.57
30	.0169	108.74	.0186	17.58	.0147	39.57	.0162	6.87
40	.0267	108.15	.0271	108.94	.0361	183.26	.0223	43.84
50	.0149	827.64	.0145	298.37	.0187	170.30	.0093	56.03

6. SUMMARY AND CONCLUSIONS

We feel the test cases provide fairly clear-cut conclusions. The 3-dimensional x-projection method with hyperplane selection and geometric acceleration, 3-XA, proved to be superior to the best known x-projection method to date, 2-XA. 3-XA required significantly less C.P.U. time for convergence in both test cases with the same accuracy. Wainwright[1] has previously shown the 2-XA method to be superior in C.P.U. time and accuracy over a wide range of test cases to Gauss–Seidel, the 1-dimensional x-projection method with acceleration, 1-XA, the best known r-projection method, 2-R as well as Gaussian elimination (in the case of ill-conditioned matrices).

The quasi-optimal hyperplane selection criteria presented in Section 4(a) proved in practice to be extremely successful for increasing the rate of convergence (see Table 1, 3-X BEST vs 3-X WORST comparing C.P.U. TIME and AVE $(1/D)$ USED). The geometric acceleration technique presented in Section 4(b) also proved extremely successful in practice. For the cases where it could be applied, the C.P.U. time was reduced significantly and in the Random matrices accuracy improved by orders of magnitude (see 3-X BEST vs 3-XA BEST in Table 1, 3-X WORST vs 3-XA WORST in Table 1 and 3-X vs 3-XA in Table 2). For the single example where it could not be applied, Table 1 $n = 50$, the added cost in C.P.U. time to test for the geometric acceleration condition was insignificant. In addition, Table 2 clearly shows one of the most important characteristics of the class of x-projection methods: the ability to solve ill-conditioned systems. The nature of most ill-conditioned matrices is that they produce very large $(1/D)$ values thus making the hyperplane selections extremely effective in increasing the rate of convergence (see 2-X and 3-X in Table 2). For example, for 3-X for $n = 12$ the AVE $(1/D)$ USED is 1.38×10^7; for $n' = 20$ the AVE $(1/D)$ USED $= 1.09 \times 10^7$; for $n = 40$ the AVE $(1/D)$ USED $= 1.58 \times 10^8$ and for $n = 50$ the AVE $(1/D)$ USED $= 3.51 \times 10^7$. Compare this to the AVE $(1/D)$ USED values in Table 1. Furthermore, in addition to large $(1/D)$ values we were able in each instance to apply geometric acceleration (see 3-XA in Table 2).

In conclusion, 3-XA like all projection methods is guaranteed to converge for arbitrary starting vector and nonsingular coefficient matrix. Also, 3-XA has been shown to be a superior method over the best projection method known to date, 2-XA. 2-XA has previously been shown to be competitive and in most cases superior to other well known iterative algorithms both in rate of convergence and accuracy[1]. Furthermore, with ill-conditioned matrices 3-XA proved very effective and was shown to be superior to the 2-XA method. The author is currently working on the 4-dimensional x-projection method where even better results are expected. A program listing of method 3-XA in the form of a FORTRAN subroutine follows.

REFERENCES

1. R. L. Wainwright, Acceleration techniques for a class of x-projection methods for solving systems of linear equations. *Comp. Maths. Applics* **5**, 59–73 (1979).
2. S. M. Pizer, *Numerical Computing and Mathematical Analysis.* Science Research Associates, Chicago (1975).
3. A. de la Garza, An iterative method for solving systems of linear equations. Report K-731. Union Carbide and Carbon Corp. K-25 Plant, Oak Ridge, Tennessee (1951)
4. R. F. Keller, A single-step gradient method for solving systems of linear equations. Technical Report 8, Mathematical Sciences, University of Missouri, Columbia, Missouri (1964).
5. L. Fox, *An Introduction to Numerical Linear Algebra.* Oxford University Press, New York (1964).

6. A. S. Householder, *The Theory of Matrices in Numerical Analysis.* Blaisdell, New York (1964).
7. D. D. Georg and R. F. Keller, Criteria for ordering columns in two-dimensional projection methods. Ames Laboratory Report IS-3147, National Technical Information Service, Springfield, Virginia (1973).
8. H. D. Pyron, A non-stationary optimizing of projection methods. Ph.D. Dissertation, Iowa State University, Ames, Iowa, Report IS-T-452, National Technical Information Service, Springfield, Virginia (1971).
9. M. Tokko and R. F. Keller, Optimal three-dimensional projection method for solving linear algebraic equations. Ames, Iowa Laboratory Report IS-3039, National Technical Information Service, Springfield, Virginia (1972).
10. R. L. Wainwright, Algorithms for projection methods for solving linear systems of equations. Ph.D. Dissertation, Iowa State University, Ames, Iowa, Report IS-3397, National Technical Information Service, Springfield, Virginia (1974).
11. R. L. Wainwright and R. F. Keller, Algorithms for projection methods for solving linear systems of equations. *Comp. Maths. Applics* **3**, 235–245 (1977).
12. R. T. Gregory and D. L. Karney, *A Collection of Matrices for Testing Computational Algorithms.* Wiley, New York (1969).

```
      SUBROUTINE XPROJ3(N,A,B,X,TOL,IPRT,RDXDIF,ISUMY,MAXCYL)
      DOUBLEPRECISION A(N,N),B(N),X(N)
      DOUBLEPRECISION TOL,MAXDIF,DIFF,RDXDIF,DPA,DPR,DPSUM
      DOUBLEPRECISION DPSUMK,DPSUML,DPSUMM,COEF1,COEF2,COEF3
      DOUBLEPRECISION SQRKL,SQRKM,SQRLM
      DOUBLEPRECISION DX(50),RDX(50),OLDX(50),OLDDX(50)
      DOUBLEPRECISION AA(52),PASTX(50)
      DOUBLEPRECISION AS(52),DET(52)
      INTEGER ANGL(50,50),SP(52)
      INTEGER SW(50)
C
C****************************************************************
C**
C**  THIS SUBROUTINE SOLVES A NON-SINGULAR LINEAR SYSTEM
C**  OF EQUATIONS USING THE 3-DIMENSIONAL X-PROJECTION
C**  METHOD WITH QUASI-OPTIMAL HYPERPLANE SELECTION
C**  AND GEOMETRIC ACCELERATION.
C**
C**         ----- DESCRIPTION OF THE ARGUMENTS -----
C**
C**  N      - THE DIMENSION OF THE LINEAR SYSTEM.  THE
C**           SUBROUTINE IS CURRENTLY SET UP FOR A MAX OF N=50
C**  A      - THE  COEFFICIENT MATRIX OF THE LINEAR SYSTEM
C**  B      - THE CONSTANT VECTOR OF THE LINEAR SYSTEM
C**  X      - THE APPROXIMATE SOLUTION VECTOR.  INITIALLY,
C**           X CONTAINS THE INITIAL GUESS VECTOR AND
C**           WILL CONTAIN THE SOLUTION UPON COMPLETION OF
C**           THE ALGORITHM. UNLESS SOME PRIOR
C**           KNOWLEDGE OF THE SOLUTION IS KNOWN THE ZERO
C**           VECTOR IS SUGGESTED FOR THE INITIAL GUESS
C**  TOL    - TOLERANCE LIMIT FOR DETERMINING CONVERGENCE. THE
C**           ITERATIVE PROCESS TERMINATES WHEN EVERY
C**           CORRESPONDING ELEMENT OF THE APPROXIMATE SOLUTION
C**           VECTOR FROM TWO SUCCESSIVE CYCLES IS WITHIN
C**           THIS VALUE. SUGGESTED VALUES RANGE FROM 0.00001
C**           TO 0.0000001.   .000005 HAS BEEN USED WITH SUCCESS
C**  IPRT   - EVERY IPRT NUMBER OF CYCLES STATISTICS WILL BE
C**           GATHERED AND ANALYZED TO DETERMINE IF GEOMETRIC
C**           ACCELERATION CAN BE PERFORMED. IN MOST CASES
C**           VALUES FROM 25 TO 200 WORK QUITE WELL,
C**  RDXDIF- WHEN THE RATIO OF THE CHANGES IN THE APPROXIMATE
C**           SOLUTION VECTOR (AS GATHERED EVERY IPRT CYCLES)
C**           IS WITHIN THIS VALUE THEN GEOMETRIC ACCELERATION
C**           IS PERFORMED. **ONE MUST BE VERY CAREFUL WITH
C**           THIS ARGUMENT**  IF RDXDIF IS TOO LARGE PRE-
C**           MATURE ACCELERATION MAY OCCUR AND THE SYSTEM MAY
C**           DIVERGE.  OF COURSE, IF RDXDIF IS TOO SMALL
C**           ACCELERATION MAY NEVER OCCUR. RECOMMENDED VALUE
C**           IS .005.  ( ON THE HILBERT MATRIX OF SIZE>=40
C**           I USED .1 AND IT WORKED QUITE WELL. FOR A WELL-
C**           CONDITIONED SYSTEM .00005 MAY BE A BETTER VALUE.)
C**  ISUMY - EVERY ISUMY CYCLES PROGRESS OF THE SOLUTION IS
C**           REPORTED.  THE CYCLE NUMBER, APPROX. SOLUTION
C**           VECTOR AND INNER PRODUCT OF THE ERROR VECTOR
C**           ARE GIVEN.  IF ISUMY IS <= 0 THEN NO SUMMARY
C**           IS GIVEN.
```

```
C**  MAXCYL- THE MAXIMUM NUMBER OF CYCLES YOU WISH TO ITERATE.
C**         IF THE SOLUTION IS NOT DETERMINED BEFORE THIS,
C**         THE PROCESS IS HALTED WITH SUMMARY INFO. GIVEN.
C**
C*****************************************************************

 C ************** OVERHEAD CALCULATIONS *********************
 C ***                                                     ***
 C ***                                                     ***
 C
 C --- NORMALIZE THE COEF MATRIX, A ---------------
 C
       DO 30 I=1,N
         DPSUM=0.
         DO 10 J=1,N
           DPA=A(I,J)
           DPSUM=DPSUM + DPA*DPA
 10      CONTINUE
         DPSUM= DSQRT(DPSUM)
         DO 20 J=1,N
           A(I,J)= A(I,J)/DPSUM
 20      CONTINUE
       B(I)=B(I)/DPSUM
 30    CONTINUE
 C
 C ---   END OF NORMALIZING THE A MATRIX ----------
 C
 C ----- CALCULATE THE ANGLES BETWEEN ROWS OF A ---
 C
       DPA=180.0
       DPA=DPA/3.14159
       DO 60 I=2,N
         L=I-1
         DO 50 J=1,L
           DPSUM=0.
           DO 40 K=1,N
             DPSUM=DPSUM+A(I,K)*A(J,K)
 40        CONTINUE
            TEMP=DPSUM
            ANGL(I,J)=ACOS(TEMP)*DPA
 C               MAKE ALL ANGLES <= 90 TO MORE EASILY
 C               DETERMINE THE ROW VECTOR PAIRINGS
            IF(ANGL(I,J).GT.90) ANGL(I,J)=180-ANGL(I,J)
            ANGL(J,I)=ANGL(I,J)
 50      CONTINUE
 60    CONTINUE
 C
 C --- END OF CALCULATING THE ANGLES ---------------
 C
 C --- SELECT QUASI-OPTIMAL ROW VECTOR TRIPLES -----
 C --- THE SELECTED TRIPLES ARE PLACED INTO SP -----
 C
       LASTSP = N
       IF( LASTSP/3*3.NE.LASTSP) LASTSP=LASTSP+1
       IF( LASTSP/3*3.NE.LASTSP) LASTSP=LASTSP+1
 C          LASTSP IN THE SIZE (LENGTH) OF SP
 C
 65    SPCT=1
 ..    DO 70 I=1,N
         SW(I)=0
 70    CONTINUE

 75    MIN=1000
 C     FIND THE SMALLEST ANGL
       DO 85 I=2,N
          IF(SW(I).EQ.1) GO TO 85
          L=I-1
          DO 80 J=1,L
             IF(SW(J).EQ.1) GO TO 80
             IF(ANGL(I,J).GE.MIN) GO TO 80
 C           FOUND THE NEW MIN
             MIN=ANGL(I,J)
             LOCI=I
             LOCJ=J
```

```
80    CONTINUE
85    CONTINUE
      SW(LOCI)=1
      SW(LOCJ)=1
      SP(SPCT)=LOCI
      SP(SPCT+1)=LOCJ
C     FIND THE SMALLEST ANGL FOR LOCI AND LOCJ
      MIN=1000
      DO 90 I=1,N
        IF(SW(I).EQ.1) GO TO 90
        ISUM=ANGL(LOCI,I)+ANGL(LOCJ,I)
        IF(ISUM.GE.MIN) GO TO 90
        MIN=ISUM
        LOCK=I
90    CONTINUE
      SW(LOCK)=1
      SP(SPCT+2)=LOCK
      SPCT=SPCT+3
C
      IF(LASTSP.EQ.N.AND.SPCT.EQ.N+1) GO TO 130
C     ABOVE IS CASE OF N A MULT OF 3 AND FINISHED
      IF(LASTSP.EQ.N+2.AND.SPCT.EQ.N) GO TO 95
C     ABOVE IS CASE OF N ONE OVER MULT OF 3 AND FINISHED
      IF(LASTSP.EQ.N+1.AND.SPCT.EQ.N-1) GO TO 110
C     ABOVE IS CASE OF N TWO OVER MULT OF 3 AND FINISHED
C     NOT YET FINISHED
      GO TO 75
C
C     SPECIAL LAST CASES
C
C     CASE OF N ONE OVER MULT OF 3
95    DO 100 I=1,N
        IF(SW(I).EQ.0) GO TO 105
100   CONTINUE
105   SP(SPCT)=I
      SP(SPCT+1)=SP(1)
      SP(SPCT+2)=SP(2)
      GO TO 130
C

C     CASE OF N TWO OVER MULT OF 3
110   L=0
      DO 115 I=1,N
        IF(SW(I).EQ.1) GO TO 115
        IF(L.EQ.1) GO TO 120
        L=L+1
        J=I
115    CONTINUE
120    K=I
C      J,K ARE THE TWO  UNUSED ROWS
C
       MIN=1000
       DO 125 I=1,N
          IF(I.EQ.J) GO TO 125
          IF(I.EQ.K) GO TO 125
          ISUM=ANGL(J,I)+ANGL(K,I)
          IF(ISUM.GE.MIN) GO TO 125
          MIN=ISUM
          LOCK=I
125    CONTINUE
       SP(SPCT)=J
       SP(SPCT+1)=K
       SP(SPCT+2)=LOCK
C
C --- END OF SELECTING THE ROW VECTOR TRIPLES -----------
C
130   CONTINUE
C
C --- CALCULATE THE INNER PRODUCTS ----------------------
C --- I,I1,I2 ARE USED IN AA,AS,SP AND DET
C --- K,L,M ARE USED IN  A
C --- THE INNER PRODUCTS ARE CALCULATED FOR ONLY
C --- THE SELECTED TRIPLES
```

```
C
      DO 150 I=1,LASTSP,3
        I1 = I+1
        I2 = I+2
        K  = SP(I)
        L  = SP(I1)
        M  = SP(I2)
        SUMKL = 0.
        SUMKM = 0.
        SUMLM = 0.
        DO 140 J=1,N
          SUMKL = SUMKL+A(K,J)*A(L,J)
          SUMKM = SUMKM+A(K,J)*A(M,J)
          SUMLM = SUMLM+A(L,J)*A(M,J)
140     CONTINUE
        SQRKL = SUMKL*SUMKL
        AA(I) = SUMKM*SUMLM-SUMKL
        AS(I) = 1.-SQRKL
        SQRKM = SUMKM*SUMKM
        AA(I1) = SUMKL*SUMLM-SUMKM
        AS(I1) = 1.-SQRKM
        SQRLM = SUMLM*SUMLM
        AA(I2) = SUMKL*SUMKM-SUMLM
        AS(I2) = 1.-SQRLM
        DET(I) = 1. + 2.*SUMKL*SUMKM*SUMLM-SQRKL-SQRKM-SQRLM
150   CONTINUE
C
C --- END OF INNER PRODUCT CALCULATIONS -----------------
C
      DO 160 I=1,N
      OLDX(I)=X(I)
      OLDDX(I)=.00005
C         OLDDX IS INITIALIZED NON-ZERO TO PREVENT
C         ZERO DIVISION WHEN CALCULATING THE FIRST RDX
      PASTX(I)=X(I)
160   CONTINUE
C ***                                                      ***
C ***                                                      ***
C ****** END OF THE OVERHEAD (INITIALIZATION) CALCULATIONS ****
C ************************************************************
C
      IF(ISUMY.EQ.0) ISUMY=9999999
C ************** START OF THE CYCLE LOOP *********************
C ***                                                      ***
C ***                                                      ***
      DO 280 ICYCLE=1,MAXCYL
C
C --- CHECK IF SUMMARY INFORMATION IS TO BE PRINTED -------
C
      IF(ICYCLE/ISUMY*ISUMY.NE.ICYCLE) GO TO 190
C     PRINT SUMMARY INFORMATION
      DPR=0.
      DO 180 J=1,N
        DPSUM=0.
        DO 170 K=1,N
        DPSUM=DPSUM+ A(J,K)*X(K)
170     CONTINUE
        DPSUM=B(J)-DPSUM
        DPR=DPR+ DPSUM*DPSUM
180    CONTINUE
      WRITE(108,510) ICYCLE,DPR,(X(K),K=1,N)
C
C ---- END OF PRINT SUMMARY INFORMATION -------------------
C
190   CONTINUE
C
C --- ITERATION LOOP.  COMPUTE THE NEW
C --- APPROX. SOLUTIION VECTOR BY PROJECTING
C --- ONTO X(K),X(L) AND X(M)
C
      DO 220 I =1,LASTSP,3
        I1 =I+1
        I2 = I+2
```

```
          K  = SP(I)
          L  = SP(I1)
          M  = SP(I2)
C
C --- I,I1,I2 ARE USED IN AA,AS SP AND DET
C --- K,L,M ARE USED IN A
C
          DPSUMK =0.
          DPSUML =0.
          DPSUMM =0.
          DO 200 J=1,N
            DPSUMK = DPSUMK+X(J)*A(K,J)
            DPSUML = DPSUML+X(J)*A(L,J)
            DPSUMM = DPSUMM+X(J)*A(M,J)
200       CONTINUE
          DPSUMK  = B(K)-DPSUMK
          DPSUML = B(L)-DPSUML
          DPSUMM = B(M) -DPSUMM
          COEF1=(DPSUMK*AS(I2)+DPSUML*AA(I)+DPSUMM*AA(I1))/DET(I)
          COEF2=(DPSUMK*AA(I)+DPSUML*AS(I1)+DPSUMM*AA(I2))/DET(I)
          COEF3=(DPSUMK*AA(I1)+DPSUML*AA(I2)+DPSUMM*AS(I))/DET(I)
          DO 210 J=1,N
            X(J)=X(J)+COEF1*A(K,J)+COEF2*A(L,J)+COEF3*A(M,J)
210       CONTINUE
220      CONTINUE
C
C
        MAXDIF=0.
        DO 230 J=1,N
          DIFF=DABS(X(J)-PASTX(J))
          IF(DIFF.GT.MAXDIF) MAXDIF=DIFF
          PASTX(J)=X(J)
230     CONTINUE
        IF(MAXDIF.LE.TOL) GO TO 290
C
C --- END OF THE CYCLE CALCULATIONS -------------------
        IF(ICYCLE/IPRT*IPRT.NE.ICYCLE)GO TO 280
C
C --- ATTEMPT TO ACCELERATE
        DO 240 J=1,N
          DX(J)=X(J)-OLDX(J)
          OLDX(J)=X(J)
          RDX(J)=DX(J)/OLDDX(J)
          OLDDX(J)=DX(J)
240     CONTINUE
C
C --- IF RDX VALUES ARE ALL WITHIN RDXDIF OF EACH
C --- OTHER THEN ACCELERATE TO THE SOLUTION
C
        L=N-1
        DO 260 I=1,L
          K=I+1
          DO 250 J=K,N
            IF(DABS(RDX(I)-RDX(J)).GT.RDXDIF)  GO TO 280
250       CONTINUE
260      CONTINUE
C
        DO 270 I=1,N
          X(I)= (X(I)-DX(I)) + DX(I)/(1.0-RDX(I))
          PASTX(I)=X(I)
          OLDDX(I)=0.000001
C            OLDDX IS MADE NON-ZERO TO PREVENT ZERO
C            DIVISION ON THE NEXT CYCLE
          OLDX(I)=X(I)
270     CONTINUE

        C
        C
        280   CONTINUE
        C ***                                                  ***
        C ***                                                  ***
        C ************ END OF THE CYCLE (280 LOOP) ******************
```

```
C
C ************ SOLUTION FOUND ********************************
C
290    CONTINUE
C      CALCULATE (R,R)
       DPR=0.
       DO 310 J=1,N
         DPSUM=0.
         DO 300 K=1,N
         DPSUM=DPSUM + A(J,K)*X(K)
300      CONTINUE
         DPSUM=B(J)-DPSUM
         DPR=DPR + DPSUM*DPSUM
310    CONTINUE
C ---  PRINT THE SOLUTION
       WRITE(108,500) ICYCLE,DPR,(X(K),K=1,N)
C
500    FORMAT('0'//,'SOLUTION'/'NO OF CYCLES = ',
      X  I9,'   (R,R) = ',E18.9/'OSOLUTION VECTOR FOLLOWS',
      X /(' ',8(F13.7,1X)))
510    FORMAT(' AT CYCLE = ',I9,'   (R,R) = ',E18.9/
      X ' APPROX. SOLN. VECTOR FOLLOWS'/
      X (' ',8(F13.7,1X)))
       RETURN
       END
```