AAEC/E195

Health Physics Research
(21)

AAEC/E195

81

AUSTRALIAN ATOMIC ENERGY COMMISSION
RESEARCH ESTABLISHMENT
LUCAS HEIGHTS

SOME ITERATIVE METHODS FOR SOLVING THE MATRIX EQUATION Ax = b

by

G. DOHERTY

October 1968

AUSTRALIAN ATOMIC ENERGY COMMISSION

RESEARCH ESTABLISHMENT

LUCAS HEIGHTS

SOME ITERATIVE METHODS FOR SOLVING THE MATRIX EQUATION $Ax = b$

by

G. DOHERTY

ABSTRACT

This paper reviews the salient features of a number of iterative methods for solving the matrix equation $Ax = b$, and includes a brief description of the calling sequences to Fortran subroutines written for the IBM 360/50 with an assessment of the computational efficiency of the different methods for low order full matrices. Some attention is given to the question of acceleration and the necessity for double precision arithmetic.

CONTENTS

1. INTRODUCTION

Some iterative methods for solving the matrix equation $Ax = b$ have been programmed for the IBM 360/50 H computer in Fortran IV. This report gives a brief outline of each of the iterative methods for which a routine is available (more detailed descriptions and discussion of the applicability of the different methods are given by Varga (1962) for example). The extension from point methods to block methods while retaining the same iteration scheme is described in some detail.

Finally some general remarks are made about acceleration procedures and the applicability of each scheme to a particular class of matrices. It is intended that a routine testing of the different methods against a matrix typical of a given class will be used to decide which method is most suitable for that class. A description of the calling sequences to the routines is supplied, in case the reader should wish to experiment with these routines himself.

2. POINT ITERATIVE METHODS

Iterative methods for solving the equation

$$Ax = b$$

may be written in the form

$$x = Mx + k ,$$

where an iteration matrix M is used to define the iterative procedure giving the $m+1$ iterate x^{m+1} from the previous iterate x^m

$$x^{m+1} = Mx^m + k .$$

Point iterative methods are characterised by the fact that each element of x^{m+1} is calculated in turn (as opposed to block methods where a number may be obtained simultaneously). A number of point iterative methods have been coded for the IBM 360/50H in Fortran IV. These routines were written to permit experimentation with convergence criteria, the effect of double precision on the speed of convergence for each method, and to provide a comparison of the various methods for some particular matrices. They are therefore not optimally coded, with respect to core requirements, because the iterates are kept separately in the routine; this enables double precision to be introduced without coding alterations. The point methods which have been coded are:

Gauss method

Gauss-Seidel method

Jacobi method

Aitken method

Successive over-relaxation method.

With the exception of the Gauss method all contain an acceleration procedure which we shall now describe.

Suppose that a cyclic scheme is started with an initial guess x^1 related to the real solution x by

$$x^1 = x + \alpha .$$

Applying the iteration scheme, the next iterate x^2 is given by

$$x^2 = Mx^1 + k ,$$

since $\quad x = Mx + k ,$

$$x^2 - x = M(x^1 - x)$$
$$= M\alpha .$$

In general

$$x^{m+1} - x = M^m \alpha .$$

Suppose further that the eigenvalues of M are real and positive and ordered so that

$$\lambda_1 > \lambda_2 > \lambda_3 \quad ----- \quad .$$

Then the vector α may be decomposed in terms of the eigenvectors y_i associated with the eigenvalues λ_i of M

$$\alpha = \sum_i \alpha_i \, y_i \quad ,$$

$$\begin{aligned} x^{m+1} &= x + M^m (\alpha) \\ &= x + M^m (\sum_i \alpha_i \, y_i) \\ &= x + \sum_i \alpha_i \, \lambda_i^m \, y_i \quad . \end{aligned}$$

We shall suppose also that the coefficient α_1 is non-zero. (Rounding errors will probably introduce a y_1 component even if the initial error vector did not contain one).

Then as m becomes large

$$x^{m+1} \longrightarrow x + \lambda_1^m \, \alpha_1 y_1 .$$

Considering successive iterations,

$$\begin{aligned} x^{m+1} &\approx x + \lambda_1^m \, \alpha_1 \, y_1 \quad , \\ x^{m+2} &\approx x + \lambda_1^{m+1} \, \alpha_1 \, y_1 \quad . \end{aligned}$$

Solving for x gives the result

$$x = x^{m+2} + \frac{\lambda_1}{1 - \lambda_1} (x^{m+2} - x^{m+1}) \quad ,$$

which requires the vectors x^{m+1} and x^{m+2}, whether or not the iterative scheme in question needs to have both stored simultaneously. For example the Jacobi method requires x^{m+1} to be retained until all the elements of x^{m+2} are calculated, while the Gauss-Seidel method successively replaces the elements of x^{m+1} by the corresponding elements of x^{m+2} and in its unaccelerated form does not need storage for both iterates. With acceleration the Jacobi and Gauss-Seidel methods both require the same storage. However the gain in computational speed when acceleration is used would usually outweigh storage considerations.

An estimate of the quantity λ_1 is usually available without much additional effort. When the iterates are obtained the true solution x is of course unknown, but some measure of the convergence of the iterative procedure must be calculated to determine when the iterations can be terminated. One common measure of the convergence is the Euclidean norm of the vector

$$\| x^{m+2} - x^{m+1} \| = \left[\sum_i (x_i^{m+2} - x_i^{m+1})^2 \right]^{\frac{1}{2}}$$

which can be used to stop iteration with a test of the type:

$$\| x^{m+2} - x^{m+1} \| < 10^{-5} .$$

In any event the quantities $\| x^{m+2} - x^{m+1} \|$ can be used to give an estimate of λ_1. Thus

$$\begin{aligned} x^{m+2} - x^{m+1} &\approx \lambda_1^m \, (\lambda_1 - 1) \, \alpha_1 \, y_1 \quad , \\ x^{m+1} - x^m &\approx \lambda_1^{m-1} \, (\lambda_1 - 1) \, \alpha_1 \, y_1 \quad , \end{aligned}$$

$$\frac{\| x^{m+2} - x^{m+1} \|}{\| x^{m+1} - x^{m} \|} \approx \lambda_1 \quad .$$

Some difficulties may be encountered in the application of this acceleration procedure. In deriving the formulae we have used

$$x^{m+1} = x + \lambda_1^m \alpha_1 y_1 \quad ,$$

which assumes that sufficient iterations have elapsed to make the contribution of the other eigenvalues to x^{m+1} negligible. This assumption could be untrue if the coefficient α_1 were small for the particular x^1 chosen, or, more usually, if the subdominant eigenvalues are very close to λ_1 in magnitude. There does not seem to be a generally applicable answer to the problem of finding an optimum acceleration interval. If a condition is placed on the variation of the estimate of λ_1 from iteration to iteration then we may miss the advantage of applying the acceleration earlier (even when it is not quite accurate in ignoring lower eigenvalues). We must also store the previous vector iterate if the unaccelerated procedure would be overwriting it and this also lowers the computational speed of the resulting programme. On the other hand, if the number of iterations is too small,introduction of the acceleration procedure often will slow the convergence considerably and may inhibit convergence altogether. A little experimentation on a typical matrix problem would be the ideal way of resolving this question for a given application. Other difficulties may arise when the matrix M does not have the spectral properties which have been assumed. For example, if the largest numerical eigenvalue is negative, then for m even,

$$x^{m+1} = x + \lambda_1^m \alpha_1 y_1 \quad ,$$

$$x^{m+2} = x - \lambda_1^{m+1} \alpha_1 y_1 \quad ,$$

$$x = x^{m+2} + \frac{\lambda_1}{1-\lambda_1} (x^{m+2} - x^{m+1}) \quad ,$$

where now λ_1 is negative. If we use the Euclidean norm estimate of λ_1 we will be unable to obtain the negative value and the acceleration will then produce a serious error. This problem can of course be surmounted by applying the acceleration to iterates separated by two so that the quantity of interest is λ_1^2, which will be positive and real if λ_1 is real.

When the dominant eigenvalues of the iteration matrix are complex, such a simple extension is usually not possible since $|\lambda_1|^n \neq \lambda_1^n$ for general complex λ_1 and any integer n. To overcome this problem the Aitken double-sweep method can be employed; this can be shown (Fox 1964) to have real eigenvalues when the other methods (Gauss-Seidel in particular) are convergent but have a complex dominant eigenvalue.

To conclude the discussion of acceleration, it must be said that the use of acceleration raises the problem of the necessity for double precision. Provided the convergence criterion on the Euclidean norm is larger than the roundoff error which may arise in calculating it, double precision arithmetic has no influence on unaccelerated procedures. However, the use of the acceleration factor $\lambda_1/(1-\lambda_1)$ when $\lambda \to 1$ requires careful calculation of the differences $(x^{m+2} - x^{m+1})$. For safety, double precision arithmetic is necessary at this point, though single precision sometimes suffices.

Each method will be illustrated for a 3 x 3 matrix and also written so that the matrix M of the iteration is clearly exhibited. A few comments on the methods are also included in the description. We will be looking at solutions of the equation

$$Ax = b$$

and will make the decomposition of A

$$A = D - E - F \quad ,$$

where D is diagonal $(a_{ij} = 0, \quad j \neq i)$,

E is strictly lower triangular $(a_{ij} = 0, \quad j \geqslant i)$,

F is strictly upper triangular $(a_{ij} = 0, \quad j \leqslant i)$.

If

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

then

$$D = \begin{bmatrix} a_{11} & 0 & 0 \\ 0 & a_{22} & 0 \\ 0 & 0 & a_{33} \end{bmatrix}, \; E = \begin{bmatrix} 0 & 0 & 0 \\ -a_{21} & 0 & 0 \\ -a_{31} & -a_{32} & 0 \end{bmatrix} \text{ and } F = \begin{bmatrix} 0 & -a_{12} & -a_{13} \\ 0 & 0 & -a_{23} \\ 0 & 0 & 0 \end{bmatrix} .$$

The formal descriptions of the iteration matrices in matrix notation are due to Varga (1962).

2.1 Gauss Method

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1$$
$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2$$
$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3$$

$$r_1 = b_1 - (a_{11}x_1 + a_{12}x_2 + a_{13}x_3)$$
$$r_2 = b_2 - (a_{21}x_1 + a_{22}x_2 + a_{23}x_3)$$
$$r_3 = b_3 - (a_{31}x_1 + a_{32}x_2 + a_{33}x_3)$$

The method selects, at any point in the calculation, the largest (in modulus) r_i and replaces the old x_i by the value which reduces r_i to zero

$$x_i = x_i + \frac{r_i}{a_{ii}} .$$

The remaining r_i are then recalculated with the new value of x_i and the largest of these is again selected, solved, and so on. The procedure is not necessarily cyclic (though it is expected that a cyclic process will emerge as the number of iterations increases). Because it is non-cyclic it cannot be written in the form

$$x^{m+1} = M x^m + k ,$$

nor can it be accelerated. This is a serious drawback because the acceleration offers such a saving in execution time for the other methods. Its success in the WDSN programme (Askew and Brissenden 1963) must be viewed as fortunate, and the more recent addition of an acceleration procedure to this code (which suppresses the group selection technique and turns the scheme into a cyclic one) seems assured of better execution times (Green 1967). There seems to be no possible advantage to be gained by its use, except for some particular starting vectors, and our discussion will be on the basis of general starting vectors.

2.2 Jacobi Method

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1$$
$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2$$
$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3$$

$$x_1 = \frac{1}{a_{11}} [b_1 - (a_{12}x_2 + a_{13}x_3)]$$

$$x_2 = \frac{1}{a_{22}} [b_2 - (a_{21}x_1 + a_{23}x_3)]$$

$$x_3 = \frac{1}{a_{33}} [b_3 - (a_{31}x_1 + a_{32}x_2)] \quad .$$

The iteration procedure for passing from x^m to x^{m+1} is

$$x_1^{m+1} = \frac{1}{a_{11}} [b_1 - (a_{12}x_2^m + a_{13}x_3^m)]$$

$$x_2^{m+1} = \frac{1}{a_{22}} [b_2 - (a_{21}x_1^m + a_{23}x_3^m)]$$

$$x_3^{m+1} = \frac{1}{a_{33}} [b_3 - (a_{31}x_1^m + a_{32}x_2^m)] \quad .$$

In matrix notation

$$\begin{aligned} Dx &= (E + F)x + b \quad , \\ x &= D^{-1}(E + F)x + D^{-1}b \quad , \\ x^{m+1} &= D^{-1}(D + F)x^m + D^{-1}b \\ &= Mx^m + k \quad . \end{aligned}$$

We require that D^{-1} exists and that $\rho[D^{-1}(E + F)]$, the spectral radius of the iteration matrix, be less than 1 for the procedure to converge.

The method requires storage for the iterates x^m and x^{m+1} but as the accelerated version of all these routines require this double storage, this cannot be viewed as a serious disadvantage. Double precision arithmetic is recommended for the accelerated procedure though convergence can sometimes be obtained as rapidly with single precision.

2.3 Gauss-Seidel Method

As in the Jacobi method,

$$x_1 = \frac{1}{a_{11}} [b_1 - (a_{12}x_2 + a_{13}x_3)]$$

$$x_2 = \frac{1}{a_{22}} [b_2 - (a_{21}x_1 + a_{23}x_3)]$$

$$x_3 = \frac{1}{a_{33}} [b_3 - (a_{31}x_1 + a_{32}x_2)] \quad .$$

Here the iteration procedure replaces a component of x as soon as it is calculated.

$$x_1^{m+1} = \frac{1}{a_{11}} [b_1 - (a_{12}x_2^m + a_{13}x_3^m)]$$

$$x_2^{m+1} = \frac{1}{a_{22}} [b_2 - (a_{21}x_1^{m+1} + a_{23}x_3^m)]$$

$$x_3^{m+1} = \frac{1}{a_{33}} [b_3 - (a_{31}x_1^{m+1} + a_{32}x_2^{m+1})] \quad .$$

In matrix notation

$$(D - E)\,x = F\,x + b \;,$$

$$x = (D - E)^{-1} F\,x + (D - E)^{-1} b \;,$$

$$x^{m+1} = (D - E)^{-1} F\,x^{m} + (D - E)^{-1} b \;.$$

$(D - E)^{-1}$ must exist; since E is strictly lower triangular D^{-1} must exist. For the process to converge, $\rho\,[(D - E)^{-1} F] < 1$.

When the Jacobi matrix $D^{-1}(E + F)$ is non-negative (when all the elements are $\geqslant 0$), then it may be shown that the unaccelerated Gauss-Seidel is superior to the unaccelerated Jacobi method. However, for the accelerated methods the same conclusion may not be valid because comparison of accelerated methods will require theorems on the subdominant eigenvalues and these theorems usually cannot be established.

2.4 Successive Over-Relaxation Method (SOR)

Successive over-relaxation is an extension of the Gauss-Seidel scheme, which is computationally similar to the acceleration procedure we have previously discussed. The Gauss-Seidel scheme is

$$x_1^{m+1} = \frac{1}{a_{11}}(b_1 - a_{12}\,x_2^{m} - a_{13}\,x_3^{m}) \;,$$

$$x_2^{m+1} = \frac{1}{a_{22}}(b_2 - a_{21}\,x_1^{m+1} - a_{23}\,x_3^{m}) \;,$$

$$x_3^{m+1} = \frac{1}{a_{33}}(b_3 - a_{31}\,x_1^{m+1} - a_{32}\,x_2^{m+1}) \;.$$

In the Point SOR method outlined below, the auxiliary vector y^{m+1} is used for convenience in the definition and is not retained in the actual computation.

$$y_1^{m+1} = \frac{1}{a_{11}}(b_1 - a_{12}\,x_2^{m} - a_{13}\,x_3^{m})$$

$$x_1^{m+1} = w\,y_1^{m+1} + (1 - w)\,x_1^{m}$$

$$y_2^{m+1} = \frac{1}{a_{22}}(b_2 - a_{21}\,x_1^{m+1} - a_{23}\,x_3^{m})$$

$$x_2^{m+1} = w\,y_2^{m+1} + (1 - w)\,x_2^{m}$$

$$y_3^{m+1} = \frac{1}{a_{33}}(b_3 - a_{31}\,x_1^{m+1} - a_{32}\,x_2^{m+1})$$

$$x_3^{m+1} = w\,y_3^{m+1} + (1 - w)\,x_3^{m} \;,$$

where w is the over-relaxation parameter.

In matrix notation

$$(D - w\,E)\,x^{m+1} = \{(1 - w)\,D + w\,F\}\,x^{m} + w\,b \;,$$

$$x^{m+1} = (D - w\,E)^{-1}\{(1 - w)\,D + w\,F\}\,x^{m} + (D - w\,E)^{-1}\,w\,b \;,$$

The parameter w must be estimated, and this is the major problem in the use of point SOR. In block relaxation problems the best value of w is often determined by some cyclic property of the block iteration matrix. No such theory exists for the point methods and there seems little advantage in using anything other than w = 1 (which is, of course, Gauss-Seidel).

As has previously been remarked, the use of the acceleration procedure in these methods, while computationally desirable or even necessary, removes the possibility of performing a rigorous comparative analysis of the point methods. Firstly the acceleration procedure is applied after some specific number of iterations (or, with more sophistication, when the estimate of the largest eigenvalue has settled to a specified tolerance). In any event the success of the acceleration procedure depends on the proximity of the subdominant eigenvalues to the dominant one and the size of the coefficients of the appropriate eigenvectors in the expansion of the initial error. These properties are very much problem-dependent and the working principle which has been adopted (in the pious hope that it is reasonable) is that if a particular procedure is better than another when both are unaccelerated it will be no worse when both are accelerated.

The acceleration procedure also assumes that the largest eigenvalue of the iteration matrix is real and positive. The last point method which is discussed is designed to ensure that this result will be true. The analysis of this method and the proof that the largest eigenvalue is real is given by Fox (1964).

2.5 Aitken Method

This is a variant of Gauss-Seidel in which an up-and-down sweep of the equations is performed on each iteration. y^{m+1} is, as before, an auxiliary vector.

$$x_1 = \frac{1}{a_{11}} (b_1 - a_{12} x_2 - a_{13} x_3)$$

$$x_2 = \frac{1}{a_{22}} (b_2 - a_{21} x_1 - a_{23} x_3)$$

$$x_3 = \frac{1}{a_{33}} (b_3 - a_{31} x_1 - a_{32} x_2)$$

$$y_1^{m+1} = \frac{1}{a_{11}} (b_1 - a_{12} x_2^m - a_{13} x_3^m)$$

$$y_2^{m+1} = \frac{1}{a_{22}} (b_2 - a_{21} y_1^{m+1} - a_{23} x_3^m)$$

$$x_3^{m+1} = \frac{1}{a_{33}} (b_3 - a_{31} y_1^{m+1} - a_{32} y_2^{m+1})$$

$$x_2^{m+1} = \frac{1}{a_{22}} (b_2 - a_{21} y_1^{m+1} - a_{23} x_3^{m+1})$$

$$x_1^{m+1} = \frac{1}{a_{11}} (b_1 - a_{12} x_2^{m+1} - a_{13} x_3^{m+1}) \; .$$

In matrix notation

$$\begin{aligned}
(D-E)y &= F x + b \\
y^{m+1} &= (D-E)^{-1} F x^m + (D-E)^{-1} b \\
(D-F)x^{m+1} &= E y^{m+1} + b \\
x^{m+1} &= (D-F)^{-1} E y^{m+1} + (D-F)^{-1} b \\
&= (D-F)^{-1} E (D-E)^{-1} F x^m + \\
&\quad (D-F)^{-1} [E(D-E)^{-1} + I] b \; .
\end{aligned}$$

This method requires approximately twice as many computations per iteration as the others, but its property of ensuring that the acceleration procedure is well-founded will be useful for some types of matrices which do not have a largest real eigenvalue with the other methods. If the Jacobi iteration matrix is non-negative and irreducible there will certainly be a largest real eigenvalue, but if the matrix is cyclic of index k there will be k-1 other roots equally spaced around the circle $|\lambda| = \rho(M)$.

2.6 Calling Sequences

CALL	GAUSS	(A, M, N, NIT, CON)	Gauss
CALL	GSPACC	(A, M, N, NIT, CON, NACC)	Gauss-Seidel
CALL	JPACC	(A, M, N, NIT, CON, NACC)	Jacobi
CALL	ATPACC	(A, M, N, NIT, CON, NACC)	Aitken
CALL	SOPACC	(A, M, N, NIT, CON, NACC, W)	SOR

where A is assumed dimensioned (M, M+2) ,

the order of the equations is $N \leqslant M$,

$a_{11}, a_{12}, a_{13}, \ldots a_{1N}$ are in A(1,1) , A(1,2) . . . A(1,N) ,

$a_{N1}, a_{N2}, \ldots\ldots a_{NN}$ are in A(N,1) , A(N,2) . . . A(N,N) ,

b is in A(1,N+1) . . . . A(N,N+1) ,

x guess is in A(1,N+2) . . . . A(N,N+2) ,

NIT is the maximum number of iterations permitted,

CON is the convergence criterion on the Euclidean norm,

NACC is the number of iterations before each acceleration attempt,

and W is the over-relaxation parameter of that method.

In each routine the matrix A and the vector b remain undisturbed. On exit the x guess is replaced by the solution obtained by the routine.

3. BLOCK ITERATIVE METHODS

Block iterative methods are extensions of the point iterative methods in that a different splitting of the matrix is employed to generate the iteration matrix. Recalling the method for deriving the point iterative methods, let

$$A = D - E - F,$$

where D is diagonal, E is lower triangular, and F is upper triangular. Then we may write the equation,

$$A x = b ,$$

as

$$(D - E - F) x = b .$$

Now the various methods are simply different combinations of the two sides of this equation. For example,

$$\left.\begin{array}{l} D\,x = (E + F)\,x + b \\ \quad x = D^{-1}(E + F)\,x + D^{-1}\,b \\ \text{which gives the iteration scheme} \\ \quad x^{m+1} = D^{-1}(E + F)\,x^{m} + D^{-1}\,b \end{array}\right\}\ \text{Jacobi}\ ,$$

and

$$\left.\begin{array}{l} (D - E)\,x = F\,x + b \\ \quad x = (D - E)^{-1}\,F\,x + (D - E)^{-1}\,b \\ \text{which gives the iteration scheme} \\ \quad x^{m+1} = (D - E)^{-1}\,F\,x^{m} + (D - E)^{-1}\,b \end{array}\right\}\ \text{Gauss-Seidel}\ ,$$

and so on for the methods we have previously discussed.

Block iterative schemes can be derived in precisely the same manner by relaxing the condition that D is to be diagonal and requiring instead that D is to be block diagonal. Thus D can be written

$$\left[\begin{array}{c|c|c|c|c} D_{11} & 0 & 0 & 0 & 0 \\ \hline 0 & D_{22} & 0 & - & 0 \\ \hline 0 & 0 & D_{33} & - & 0 \\ \hline 0 & 0 & 0 & - & 0 \\ \hline 0 & 0 & 0 & - & D_{nn} \end{array}\right]$$

where each D_{ii} is square.

Then each element x_i of a point iterative scheme becomes an element X_i of the partitioned vector X. For example, if D_{11} is a 3 x 3 matrix then X_1 will consist of 3 elements which will have to be obtained by simultaneous solution of the first three equations. Iterative schemes cannot really be contemplated for the simultaneous solutions required in each step of a block iterative method, because the convergence properties of inner-outer iterative schemes cannot easily be determined theoretically. Two methods of solving within a block present themselves – either the set can be solved on each iteration using an elimination technique such as Gauss-Jordan, or the diagonal matrices D_{ii} can be completely inverted before iteration commences and the block element X_i is then obtained from a single matrix multiplication of a vector. In view of the likely number of iterations, and the likely block sizes for a particular matrix, the inversion of the diagonal elements will usually be worthwhile. If the D_{ii} are of small order a large number of iterations will often be required and the extra effort to compute D^{-1} once will be negligible. If the D_{ii} are of large order (for example half the order of A) then it would probably not be economic to compute and store the inverse.

Of course these block methods can be accelerated in the same way as can the point methods, and only the iteration matrix M is different. The purpose of using the block methods rather than the point methods is to reduce the spectral radius of the iteration matrix M. Against this reduction must be weighted the extra time involved in computing the inverses or solving the equations within a block at each iteration. The main advantage of the block methods is that special partitionings can lead to matrix equations with cyclic properties, which enable quantities such as the optimum over-acceleration parameter w to be simply determined.

With these methods, the extension from point to block method can be made quite easily. It is necessary to retain an additional storage area, as long as the vector in any of its partitions, to cope with the matrix multiplication in the block methods. The coding is only slightly more complex than for the point relaxation methods and the opportunity to use such spectral-radius-reducing schemes offsets the additional effort involved.

A further vector beyond those required for the point methods is needed as input to these routines. This vector defines the disposition of partitionings within the matrix A. The additional variables are NCUT and (NCUTT(I), I = 1, NCUT) where NCUT is the number of diagonal blocks. Block 1 consists of Equations 1 to NCUTT(1); Block 2 contains Equations NCUTT(1) + 1 to NCUTT(2) etc. Obviously NCUTT (NCUT) = N. By allowing the partitioning to be completely arbitrary we have introduced some coding difficulties into the methods where the inverses are not computed, but the equations are solved by elimination on each iteration.

Variations in the answers in the block methods are larger than in the corresponding point methods, due to the single precision computation of the inverse. Two different block sizes in a block Jacobi solution of the same 50 x 50 matrix problem will produce differences in the fifth significant digit, and the error can get larger if the diagonal submatrices are nearly singular. In most physical problems errors introduced by the inversion of the diagonal submatrices are not an embarrassment since the precision to which the inverse is calculated is usually better than that to which the original elements are known. The purpose of introducing double precision is not to ensure that the 'right' answer is obtained but rather to establish definitely when an answer is properly converged. Within this framework, double precision computation of the inverses of the diagonal submatrices might give a slightly different set of answers, but it would have no effect on the convergence properties and is therefore unnecessary.

The previous remarks about retaining an extra storage area for these matrices need to be qualified. If the matrix is explicitly defined (and thus occupies a storage area equal to the square of the length of the vector), when the inverses have been computed the multiplication by D^{-1} can be performed explicitly. If this is done the computation time per iteration is reduced by about 10 per cent and the extra storage area for an intermediate vector iterate is not yet required. If the matrix is only defined implicitly it is not possible to form the complete iteration matrix by multiplying by D^{-1}. The penalties incurred are a slight increase in time and the extra storage; against this is the saving of core storage achieved by implicit definition of the matrix A.

3.1 Calling Sequences

```
CALL  GSBAC1  (A, M, N, NIT, CON, NACC, NCUT, NCUTT)  Gauss-Seidel

CALL  JBAC1   (A, M, N, NIT, CON, NACC, NCUT, NCUTT)  Jacobi

CALL  JBAC2   (A, M, N, NIT, CON, NACC, NCUT, NCUTT)  Jacobi

CALL  AIBAC1  (A, M, N, NIT, CON, NACC, NCUT, NCUTT)  Aitken

CALL  SOBAC1  (A, M, N, NIT, CON, NACC, W, NCUT, NCUTT)  SOR
```

The arguments of the subroutines are those defined for the point methods with the extra variables NCUT and NCUTT, where

NCUT = the number of diagonal blocks in the partitioning of the matrix, and

NCUT(I) I = 1, NCUT specifies the way in which the partitioning is to be achieved.

The diagonal block I is assumed to contain elements A(J,K) where

J = NCUTT(I-1) + 1 to NCUTT(I) ,

K = NCUTT(I-1) + 1 to NCUTT(I) ,

NCUTT(0) is assumed 0 and need not be defined,

and NCUTT(NCUT) = N (the order of the equations) and should be defined.

4. SOME RESULTS

Some results will now be presented for a 50 x 50 matrix of the type that might be found in solving the thermal group fluxes of a space-independent reactor cell, where the solution of the slowing-down groups can be found without iteration and the source for the thermal groups is provided by scattering from the slowing-down groups. The matrix has been generated with all off-diagonal elements negative and drawn randomly from (0,1). The diagonal elements a_{ii} satisfy the condition

$$a_{ii} = - \underset{j \neq i}{\mathrm{Sum}} \; a_{ij} + 0.1 \; ,$$

ensuring that the Jacobi point iterative matrix will have a spectral radius less than unity and will also be irreducibly non-negative. In fact, with this definition the spectral radius of the matrix is very close to unity, so that the use of unaccelerated point methods may be excluded by their extremely slow rates of convergence. Such a matrix in a reactor cell problem would be produced by the presence of very small concentrations of absorber in a moderator of low thermal cross section..

The convergence criterion used throughout the comparisons was that the Euclidean norm $\| x^m - m^{m+1} \|$ where x^m and x^{m+1} represent successive iterates should be less than 10^{-5}. A maximum number of 50 iterations was imposed because it would be absurd to use an iterative method involving a number of iterations equal to the order of the matrix. Matrix inversions involved in the block iterative methods were performed using the SID routine (Pollard 1963, unpublished work) and the time involved in inversion (and remultiplication) is included.

Table 1 shows a number of different runs of SOR routines using various values of w and the acceleration interval, as well as the number of blocks in the partitioning of the matrix. This matrix is a full matrix so the block methods decompose into cyclic matrices only for the particular case of the 2-block partition where the Jacobi iteration matrix becomes of the form

$$\left(\begin{array}{c|c} O & F \\ \hline E & O \end{array} \right) \qquad \text{which is 2-cyclic.}$$

Without any cyclic properties, determination of a relaxation parameter is difficult and for the results quoted the best value of w would be w = 1 (the Gauss-Seidel method).

For those Gauss-Seidel cases where the acceleration interval is 10 and the process is stopped after 11 iterations, the time spent in the routine decreases as the number of blocks increases (with a corresponding reduction in the size of the diagonal submatrices which must be inverted). It appears that the convergence properties of the accelerated method depend more on the acceleration interval than on the number of blocks in the matrix partitioning and that in such circumstances the point method will be quicker than any non-trivial partitioning (especially since the coding of the point method is simpler, involving substantially fewer indexing operations).

For intercomparison of block methods, we shall ignore the SOR results since the best SOR results were obtained with w = 1 and Gauss-Seidel will be included specifically. Comparisons of the results presented in Table 2 indicate that block methods are not suitable for use with matrices of this type. This is hardly surprising since the matrix is full (its elements are non-zero) and block methods do involve an inversion before iteration can commence. Should the matrix exhibit some features such as block tridiagonal form then obviously the block methods would assert their advantages in skipping multiplications by zero which the point method will perform. The attainment of an optimum over-relaxation parameter is difficult without some cyclic property of the matrix to guide the choice, and it would appear that Gauss-Seidel is probably as good as any simple SOR scheme could achieve.

Between Jacobi, Gauss-Seidel and Aitken there seems little to choose. In their accelerated forms Jacobi and Gauss-Seidel require identical storage and take equivalent times and while theoretically the Gauss-Seidel matrix has a smaller spectral radius than the corresponding Jacobi matrix, the acceleration procedure evens out the advantage of the Gauss-Seidel, leaving little to choose between the two. The Aitken method is distinct from either of these and offers more certainty of success for this type of matrix. In each iteration of the Aitken method there is almost twice as much arithmetic as in Gauss-Seidel, with the coding not quite so simple. Against this the eigenvalues of the Aitken iteration matrix are real (for the class of matrices under discussion) and positive, so the acceleration procedure will always rest on correct assumptions.

It does not seem worthwhile to pursue the discussion of block methods further. Most applications of the block methods are somewhat specialised, in that the physical problem is presented in a form which is obviously suited to a block iteration scheme. When such a form does not exist, as in the matrices presently under discussion, it is not surprising that the advantages of block methods are not manifested.

The matrix used in the test runs has an irreducible non-negative Jacobi iteration matrix for which a number of theorems exist. For example, it can be proved that the rate of convergence of the Gauss-Seidel method is twice that of the Jacobi method for this type of matrix. This result is of no practical interest, since neither method will converge without acceleration and the acceleration procedure prevents the application of non-negative matrix theory to the iteration procedure. The effect of acceleration on different methods is strikingly illustrated for the 2-block Gauss-Seidel and Jacobi comparison where the 2-cyclic form of the block Jacobi iteration matrix precludes the application of the standard acceleration technique because both $+\rho(M)$ and $-\rho(M)$ are eigenvalues of M. The theory at the moment is confined mostly to 'straight' iteration techniques, which when implemented often contain empirical improvements of varying degrees of sophistication, for which no satisfactory theory has been evolved. Some numerical experiments should always be performed on a typical matrix problem to decide the merits of the different methods for a particular type of matrix.

5. REFERENCES

Askew, J.R. and Brissenden, R.J. (1963). – Some improvements in the discrete ordinate method of B. G. Carlson for solving the neutron transport equation, AEEW-R161.

Fox, L. (1964). – An Introduction to Numerical Linear Algebra, Clarendon Press, Oxford, p. 197.

Green, C. (1967). – The Winfrith DSN Programme, Mark 2, AEEW-R498.

Pollard, J. (1963). – SID – A Gauss-Jordan reduction routine for matrix inversion and solution (unpublished).

Varga, R.S. (1962). – Matrix Iterative Analysis, Prentice Hall, New Jersey.

TABLE 1

A COMPARISON OF OVER-RELAXATION PARAMETERS

Number of Cuts	Acceleration Interval	w-Value	// Residual Vector //	Number of Iterations	Time (minutes)
2	5	0.5		50	
"	"	0.75	8.6 E-6	16	0.18
"	"	1.0	9.0 E-9	16	.14
"	"	1.25	2.7 E-6	16	.18
"	"	1.50	7.4 E-6	35	.25
2	10	0.50	9.6 E-6	24	.21
"	"	0.75	5.2 E-6	11	.16
"	"	1.0	1.8 E-15	11	.16
"	"	1.25	3.2 E-6	12	.17
"	"	1.50	6.6 E-6	24	.25
10	5	0.5		50	
"	"	0.75	2.6 E-6	21	.17
"	"	1.0	1.2 E-6	11	.10
"	"	1.25	1.35E-6	22	.18
"	"	1.50	5.9 E-6	40	.29
10	10	0.5	6.8 E-6	31	.23
"	"	0.75	9.9 E-6	11	.11
"	"	1.0	7.7 E-9	11	.10
"	"	1.25	9.17E-6	18	.15
"	"	1.50	5.9 E-6	27	.21
25	5	0.5		50	
"	"	0.75	5.8 E-6	22	.17
"	"	1.0	1.6 E-6	11	.094
"	"	1.25	4.2 E-6	22	.17
"	"	1.50	9.6 E-6	43	.32
25	10	0.5	6.2 E-6	31	.23
"	"	0.75	5.3 E-6	12	.10
"	"	1.0	1.3 E-8	11	.093
"	"	1.25	9.9 E-6	13	.11
"	"	1.50	7.9 E-6	28	.21
50	5	0.5		50	
"	"	0.75	7.2 E-6	22	.16
"	"	1.0	1.1 E-6	6	.078
"	"	1.25	5.0 E-6	22	.16
"	"	1.50	4.8 E-6	45	.32
50	10	0.5	8.7 E-6	33	.22
"	"	0.75	6.2 E-6	12	.093
"	"	1.0	1.9 E-8	11	.078
"	"	1.25	7.1 E-6	13	.093
"	"	1.50	9.4 E-6	28	.20

TABLE 2

A COMPARISON OF THE VARIOUS ROUTINES (BLOCK AND POINT)

Number of Cuts	Acceleration Interval	Routine	// Residual Vector //	Number of Iterations	Time (minutes)
2	5	Aitken	2.0 E−10	6	0.16
"	"	Jacobi		50	*
"	"	Gauss-Seidel	9.0 E−9	6	.14
2	10	A	2.0 E−15	11	.18
"	"	J		50	*
"	"	GS	1.8 E−15	11	.16
5	5	A	5.3 E−6	6	.12
"	"	J	5.9 E−6	18	.16
"	"	GS	3.9 E−7	11	.12
5	10	A	1.9 E−12	11	.17
"	"	J	4.1 E−6	11	.12
"	"	GS	5.1 E−9	11	.12
10	5	A	3.0 E−6	6	.11
"	"	J	7.9 E−6	11	.10
"	"	GS	1.2 E−6	11	.10
10	10	A	8.2 E−13	11	.17
"	"	J	2.7 E−8	11	.10
"	"	GS	7.7 E−9	11	.10
16	5	A	2.5 E−6	6	.10
"	"	J	2.2 E−6	11	.098
"	"	GS	1.7 E−6	11	.098
16	10	A	6.6 E−13	11	.16
"	"	J	6.3 E−9	11	.098
"	"	GS	5.9 E−9	11	.098
25	5	A	2.8 E−6	6	.098
"	"	J	1.1 E−6	11	.094
"	"	GS	1.6 E−6	11	.094
25	10	A	7.7 E−13	11	.16
"	"	J	2.7 E−9	11	.093
"	"	GS	1.3 E−8	11	.093
50	5	A	2.5 E−6	6	.084
"	"	J	8.0 E−7	11	.078
"	"	GS	1.1 E−6	11	.078
50	10	A	7.1 E−13	11	.15
"	"	J	1.9 E−9	11	.078
"	"	GS	1.9 E−8	11	.078

* The 2-cyclic Jacobi method breaks down because the assumptions underlying the acceleration procedure are then violated.