SIAM J. NUMER. ANAL.
Vol. 9, No. 2, June 1972

THE USE OF CONJUGATE GRADIENTS FOR SYSTEMS OF LINEAR EQUATIONS POSSESSING "PROPERTY A"*

J. K. REID†

Abstract. It is shown that for systems possessing Young's "Property A" [7] the work in applying the conjugate gradients algorithm [3] may be approximately halved and a vector of storage may be saved by using a technique analogous to that of Golub and Varga [2] for Chebyshev semi-iteration. We illustrate by numerical results on two test problems, and comment on the advantages of the algorithm over successive overrelaxation and Chebyshev semi-iteration.

1. Introduction. We consider the solution of systems of n linear equations

$$Ax = b, \tag{1}$$

where A is symmetric, positive definite, possesses "Property A" [7] and has been scaled so that its diagonal elements are unity; thus if the variables are suitably ordered, the system has the form

$$\begin{pmatrix} I & -F \\ -F^T & I \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}. \tag{2}$$

At the end of § 2 we also consider the case where the diagonal elements are not all unity. We have in mind problems in which F is large and sparse and the system (2) is not pathologically ill-conditioned so that an iterative method of solution is indicated. Golub and Varga [2] showed that Chebyshev acceleration of Jacobi's iteration could be implemented with only one vector of storage by using the recursions

$$\begin{aligned} x_1^{(2m+1)} &= \omega_{2m+1}\{Fx_2^{(2m)} + b_1 - x_1^{(2m-1)}\} + x_1^{(2m-1)}, && m \geqq 1, \\ x_2^{(2m+2)} &= \omega_{2m+2}\{F^Tx_1^{(2m+1)} + b_2 - x_2^{(2m)}\} + x_2^{(2m)}, && m \geqq 0, \end{aligned} \tag{3}$$

with $x_1^{(1)} = Fx_2^{(0)} + b_1$, where $x^{(0)}$ is the initial approximation. The use of this form of the algorithm reduces the work required per iteration to approximately half and saves storage too. It is our purpose here to show that similar gains can be made for the conjugate gradients algorithm [3].

In § 2 we describe the proposed new version, in § 3 we explain the advantages of the method in comparison with Chebyshev semi-iteration and successive overrelaxation, and finally in § 4 we report the results of some numerical tests.

2. The proposed version of the conjugate gradients algorithm. We consider the conjugate gradients algorithm in the form described by Rutishauser [5]. Starting with an initial iterate $x^{(0)}$ we form the corresponding residual $r^{(0)} = b - Ax^{(0)}$, set $x^{(-1)} = x^{(0)}$, $r^{(-1)} = r^{(0)}$, $e_{-1} = 0$ and for $k = 0, 1, 2, \cdots$ calculate

$$q_k = \frac{(r^{(k)}, Ar^{(k)})}{(r^{(k)}, r^{(k)})} - e_{k-1}, \tag{4}$$

* Received by the editors May 6, 1971, and in revised form August 18, 1971.

† Theoretical Physics Division, U.K.A.E.A. Research Group, Atomic Energy Research Establishment, Harwell, Didcot, Berkshire, England.

$$r^{(k+1)} = r^{(k)} + \frac{1}{q_k}[-Ar^{(k)} + e_{k-1}(r^{(k)} - r^{(k-1)})], \tag{5}$$

$$x^{(k+1)} = x^{(k)} + \frac{1}{q_k}[r^{(k)} + e_{k-1}(x^{(k)} - x^{(k-1)})] \tag{6}$$

and

$$e_k = q_k \frac{(r^{(k+1)}, r^{(k+1)})}{(r^{(k)}, r^{(k)})}, \tag{7}$$

where the inner product is defined by $(x, y) = x^T y$. The vectors $r^{(k)}$ are the residuals $b - Ax^{(k)}$, and if this substitution is made in (6) and the equation is expanded into its components, it is readily seen that it reduces to the form of (3) with $\omega_{k+1} = 1/q_k$ if and only if

$$q_k = 1 - e_{k-1}, \qquad k \geqq 0; \tag{8}$$

that is (see (4)), if and only if

$$(r^{(k)}, Ar^{(k)}) = (r^{(k)}, r^{(k)}), \qquad k \geqq 0. \tag{9}$$

An obvious way to satisfy this condition initially is to choose

$$x_2^{(0)} = b_2 + F^T x_1^{(0)} \tag{10}$$

so that $r_2^{(0)} = 0$ and the algorithm needs an initial approximation $x_1^{(0)}$ to x_1 only. It is readily verified by induction that as a consequence $r_1^{(2m-1)} = r_2^{(2m)} = 0$, $m = 1, 2, \cdots$, and relations (8) and (9) always hold. The recurrence (5) may therefore be written in the form

$$\begin{aligned} r_2^{(2m+1)} &= \frac{1}{q_{2m}}[F^T r_1^{(2m)} - e_{2m-1} r_2^{(2m-1)}], && m \geqq 0, \\ r_1^{(2m+2)} &= \frac{1}{q_{2m+1}}[F r_2^{(2m+1)} - e_{2m} r_1^{(2m)}], && m \geqq 0, \end{aligned} \tag{11}$$

which requires storage for only n variables instead of $2n$ and involves approximately half as much work. Of course the work involved in calculating e_k (equation (7)) is also halved. The recursion (6) takes the form (3) with $\omega_{k+1} = 1/q_k, k = 0, 1, 2, \cdots$, and the storage required for x may thus be halved, but with this form in use it would be necessary to calculate the residuals to find e_k (equation (7)) so we would need to store b_1 and b_2 or generate them at alternate iterations, and a multiplication of a vector by F and another vector by F^T would be involved for each iteration; since each step of the ordinary conjugate gradients algorithm involves a multiplication by A, no work would be saved.

An alternative is to update only one of $x_1^{(k)}$, $x_2^{(k)}$ by using stored residuals, updated by formulas (11); for $x_1^{(k)}$ we find from (6) the relations

$$x_1^{(2m+1)} - x_1^{(2m)} = \frac{1}{q_{2m}}[r_1^{(2m)} + e_{2m-1}(x_1^{(2m)} - x_1^{(2m-1)})], \qquad m \geqq 0, \tag{12}$$

and

$$x_1^{(2m+2)} - x_1^{(2m+1)} = \frac{e_{2m}}{q_{2m+1}}(x_1^{(2m+1)} - x_1^{(2m)}), \qquad m \geqq 0. \tag{13}$$

From (13) and (8) we find the relations

$$\begin{aligned} x_1^{(2m+2)} - x_1^{(2m)} &= \frac{1}{q_{2m+1}}(x_1^{(2m+1)} - x_1^{(2m)}) \\ &= \frac{1}{e_{2m}}(x_1^{(2m+2)} - x_1^{(2m+1)}), \qquad m \geqq 0. \end{aligned} \tag{14}$$

Using these and (12) we find the equation

$$x_1^{(2m+2)} - x_1^{(2m)} = \frac{1}{q_{2m}q_{2m+1}}[r_1^{(2m)} + e_{2m-1}e_{2m-2}(x_1^{(2m)} - x_1^{(2m-2)})], \quad m \geqq 0, \tag{15}$$

if we set $e_{-2} = 0$, $x_1^{(-2)} = x_1^{(0)}$. Thus we normally update $x_1^{(k)}$ every other iteration, but if it is decided to terminate the iteration after an odd number of steps, there is no difficulty about finding $x_1^{(2m+1)}$. We find the final value of $x_2^{(k)}$ from the equation

$$r_2^{(k)} = b_2 + F^T x_1^{(k)} - x_2^{(k)}. \tag{16}$$

We may summarize the proposed method as follows.

ALGORITHM. (a) Given $x_1^{(0)}$ find $x_2^{(0)}$ from (10) and calculate $r_1^{(0)} = b_1 - x_1^{(0)} + Fx_2^{(0)}$.

(b) For each step of the iteration use (7), (8) and (11); for each even step use (15).

(c) On termination find $x_1^{(k)}$ from (14) and (15) if k is odd and find $x_2^{(k)}$ from (16).

In this way the work per iteration is approximately halved and the storage required is reduced to n variables for $x_1^{(2m)}$ and $(x_1^{(2m)} - x_1^{(2m-2)})$ and n variables for $r_1^{(2m)}$ and $r_2^{(2m-1)}$, making $2n$ in all instead of the usual $3n$ (see [4], for example).

In the more general case in which the equations have the form

$$\begin{pmatrix} M & -F \\ -F^T & N \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}, \tag{17}$$

where M and N are diagonal matrices, we may apply the algorithm just described to the system

$$\begin{pmatrix} I & -M^{-1/2}FN^{-1/2} \\ -N^{-1/2}F^TM^{-1/2} & I \end{pmatrix}\begin{pmatrix} M^{1/2}x_1 \\ N^{1/2}x_2 \end{pmatrix} = \begin{pmatrix} M^{-1/2}b_1 \\ N^{-1/2}b_2 \end{pmatrix}. \tag{18}$$

This may be done implicitly by working with $x^{(k)}$ and $r^{(k)} = b - Ax^{(k)}$ by modifying the inner product in (7) to be given by

$$(x, y) = x_1^T M^{-1} y_1 + x_2^T N^{-1} y_2 \tag{19}$$

and replacing equations (10), (11), (15) and (16) by the equations

$$x_2^{(0)} = N^{-1}(b_2 + F^T x_1^{(0)}), \tag{20}$$

(21)
$$r_2^{(2m+1)} = \frac{1}{q_{2m}}[F^T M^{-1} r_1^{(2m)} - e_{2m-1} r_2^{(2m-1)}], \qquad m \geqq 0,$$
$$r_1^{(2m+2)} = \frac{1}{q_{2m+1}}[F N^{-1} r_2^{(2m+1)} - e_{2m} r_1^{(2m)}], \qquad m \geqq 0,$$

(22)
$$x_1^{(2m+2)} - x_1^{(2m)} = \frac{1}{q_{2m} q_{2m+1}}[M^{-1} r_1^{(2m)} + e_{2m-1} e_{2m-2} \cdot (x_1^{(2m)} - x_1^{(2m-2)})], \qquad m \geqq 0,$$

and

(23)
$$r_2^{(k)} = b_2 + F^T x_1^{(k)} - N x_2^{(k)}.$$

3. Comparison with Chebyshev semi-iteration and successive overrelaxation. For all three methods most of the computer time is likely to be spent in multiplying vectors by F or F^T, and in our numerical comparisons of the next section we have therefore compared the errors after the same number of such multiplications. Apart from these multiplications, Chebyshev iteration and successive overrelaxation require the implementation of (3) and (presumbly) the calculation of the norm of the residual in order to test for convergence whereas conjugate gradients requires the implementation of (7), (11) and (for each even step) equation (15). For each double step, involving multiplications by both F and F^T, conjugate gradients requires about $4n$ floating-point multiplications and $3n$ additions whereas the comparable figures for the other methods are $2n$ and $4n$, respectively, excluding those operations involved in multiplying by F and F^T. Thus conjugate gradients requires marginally more arithmetic, but in our numerical experiments (see next section) we found that this difference was rarely detectable by the IBM 360 timer.

The version of conjugate gradients described in § 2 requires storage for $r_1^{(2m)}$, $r_2^{(2m+1)}$, $x_1^{(2m)}$ and $(x_1^{(2m)} - x_1^{(2m-2)})$, making a total of $2n$ variables ignoring the storage required for F. The same amount of storage is required for the other two methods unless the vector b can be generated by the program in which case storage for only $x_1^{(2m+1)}$ and $x_2^{(2m)}$, that is n variables, is required.

Both conjugate gradients and Chebyshev semi-iteration result in error vectors of the form

(24)
$$x^{(k)} - x = P_k(A)(x^{(0)} - x),$$

where P_k is a polynomial of degree k such that $P_k(0) = 1$. In the case of conjugate gradients the choice is such that the quadratic form

(25)
$$(x - x^{(k)}, A(x - x^{(k)}))$$

is minimized (see [3, p. 416] or [4, p. 234]), and in the case of Chebyshev iteration it is proportional to the kth Chebyshev polynomial on some range $[a, b]$ containing all the eigenvalues of A. In the sense of expression (25) conjugate gradients is optimal among all methods using iterates of the form (24), and it will choose polynomials whose values at the eigenvalues of A are small. Chebyshev iteration chooses its polynomial to be small over the whole range $[a, b]$ and ignores the detailed positions of the eigenvalues. In the early stages this makes little difference but the later convergence rate of conjugate gradients is usually much better and Figs. 1 and 2 show typical examples.

Golub and Varga [2] show that successive overrelaxation and Chebyshev iteration in the form (3) have the same asymptotic rate of convergence if each has its parameters chosen for best asymptotic convergence. Thus we may expect conjugate gradients to show gains over successive overrelaxation similar to those it shows over Chebyshev iteration, and this is again illustrated by Figs. 1 and 2.

Besides this better ultimate rate of convergence conjugate gradients has the substantial advantage that no estimation of a parameter is necessary.

The conjugate gradients method should give an exact answer after at most n steps, and the fact that this usually does not happen in the presence of roundoff has perhaps led it into disrepute. The failure of this property is of no consequence here, for we expect far less than n steps to be necessary anyway. We are, in fact, using it as an iterative method although it is often regarded as a direct one. We have discussed this point in more detail in [4].

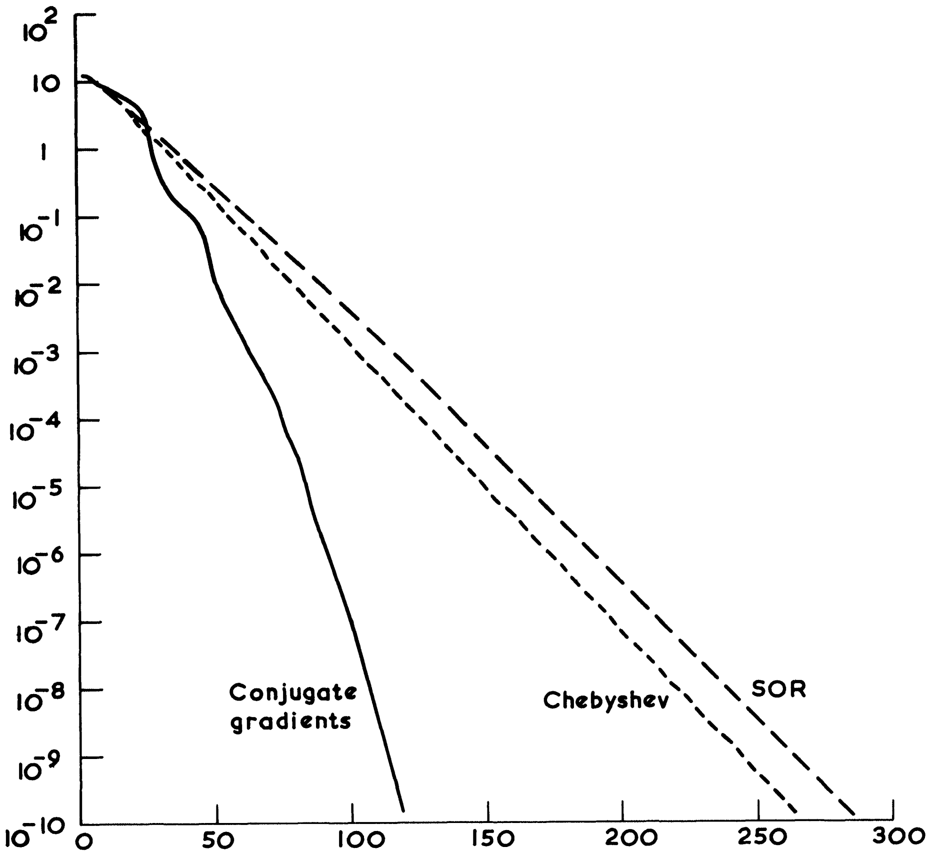

FIG. 1. *Graph of* $\|x_1 - x_1^{(k)}\|_2$ *for case* (a)

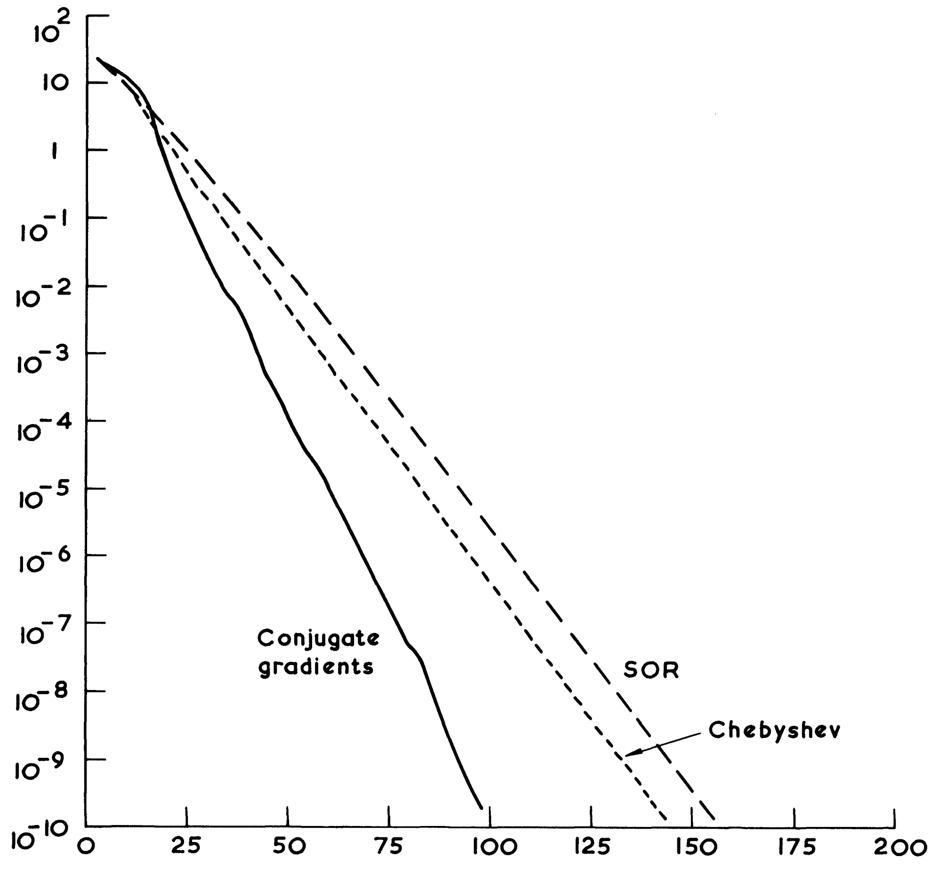

FIG. 2. *Graph of* $\|x_1 - x_1^{(k)}\|_2$ *for case* (b)

4. Numerical tests. For our numerical tests we have used (a) the matrix of order 961 arising from the solution of Laplace's equation over a grid of 31 × 31 points and (b) the matrix of order 4080 arising from the solution of Laplace's equation over a grid of 15 × 16 × 17 points. Problem (a) was used as a model problem by Stone [6], and problem (b) was considered by Reid [4]. In each case the vector b was chosen so that the solution x had all components unity, and an initial iterate was generated by using the recursion

$$\alpha_{i+1} = \text{fractional part of } 2899\,\alpha_i, \qquad i = 1, 2, \cdots, \tag{26}$$

with $\alpha_1 = (2899)^2 \times 2^{-23}$; this produces pseudo-random numbers in [0, 1] which have been checked statistically to be rectangularly distributed. All computations were performed on an IBM 360/75 computer using 8-byte arithmetic (approximately 16 significant decimal places). Figures 1 and 2 show $\|x_1 - x_1^{(k)}\|_2$ for the two problems using successive overrelaxation with ω chosen to give optimum asymptotic convergence, Chebyshev semi-iteration with the correct eigenvalue range and conjugate gradients. To make a fair comparison we took the SOR iterates to be $x^{(2m)}$, $m = 0, 1, 2, \cdots$, and we regarded conjugate gradients as starting at $k = 2$ because multiplications by F and F^T are necessary to initialize this iteration. The conjugate gradients algorithm shows slightly poorer results for small k, mainly because of these initialization costs, but for k large it shows clear advantages. In fact we have rather understated the case for using conjugate gradients because the optimum parameters for the other two iterations may not be known in advance, and might have to be estimated, hence giving slower convergence.

To check that the difference in arithmetic operations does not have any substantial effect we timed our program when compiled under the FORTRAN H (optimizing) compiler. We found that the difference in time per iteration was so small that, over the number of iterations performed, it could not be quantified in view of the variation in the times produced by the IBM clock. However there was a significant difference between the times taken by the methods to achieve a specified accuracy and we show some examples in Table 1.

TABLE 1
Numbers of iterations and times for the two problems

$\|x_1 - x_1^{(k)}\|$	Number of iterations			Time (secs)		
	SOR	Cheb.	c.g.	SOR	Cheb.	c.g.
10^{-1}	62 38	54 32	42 24	1.8 5.8	1.6 5.0	1.1 3.9
10^{-4}	138 78	124 70	74 50	4.0 11.1	3.6 10.5	2.0 8.0
10^{-7}	212 118	196 106	98 76	6.0 16.7	5.5 15.2	2.7 12.0
10^{-10}	286 156	264 144	118 98	7.9 22.1	7.9 20.5	3.4 15.2

We have also used the conjugate gradients algorithm with great success (see [1]) for the least-squares problem of minimizing the expression

$$\phi(\rho, c) = \sum_{g_{ij} \neq 0} (\log_\beta |g_{ij}| - \rho_i - c_j)^2 \tag{27}$$

which arises from the problem of choosing row and column scaling factors $\beta^{-\rho_i}$ and β^{-c_j} for a given matrix $\{g_{ij}\}$ prior to triangularizing it by Gaussian elimination on a computer with floating-point base β. The normal equations arising from the least-squares problem (27) have the form (17) with $f_{ij} = -1$ if $g_{ij} \neq 0$ and zero otherwise. We have found that only about 8 or 9 multiplications by F or F^T are needed even when G is large and sparse with order in the hundreds.

REFERENCES

[1] A. R. CURTIS AND J. K. REID, *On the automatic scaling of matrices for Gaussian elimination*, J. Inst. Math. Appl., to appear.

[2] G. H. GOLUB AND R. S. VARGA, *Chebyshev semi-iterative methods, successive overrelaxation iterative methods and second order Richardson iterative methods*, Numer. Math., 3 (1961), pp. 147–168.

[3] M. R. HESTENES AND E. STIEFEL, *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. Standards Sect. B, 49 (1952), pp. 409–436.

[4] J. K. REID, *On the method of conjugate gradients for the solution of large sparse systems of linear equations*, Proc. Conference on "Large sparse sets of linear equations", Academic Press, New York, 1971.

[5] H. RUTISHAUSER, *Theory of gradient methods*, Refined Iterative Methods for Computation of the Solution and the Eigenvalues of Self-Adjoint Boundary Value Problems, M. Engeli, Th. Ginsburg, H. Rutishauser and E. Stiefel, Birkhauser, Basel, 1959.

[6] H. L. STONE, *Iterative solution of implicit approximations of multi-dimensional partial differential equations*, this Journal, 5 (1968), pp. 530–558.

[7] D. YOUNG, *Iterative methods for solving partial difference equations of elliptic type*, Trans. Amer. Math. Soc., 76 (1954), pp. 92–111.