Conjugate Gradient Methods for Indefinite Systems

R Fletcher

1. Introduction

Conjugate gradient methods have often been used to solve a wide variety of numerical problems, including linear and nonlinear algebraic equations, eigenvalue problems and minimization problems. These applications have been similar in that they involve large numbers of variables or dimensions. In these circumstances any method of solution which involves storing a full matrix of this large order, becomes inapplicable. Thus recourse to the conjugate gradient method may be the only alternative. For problems in linear equations, the conjugate gradient method requires that the coefficient matrix A, is not only symmetric but also positive definite. This restriction is also implicit in applications to unconstrained minimization. Yet there are many problems in which non-trivial equations have to be added to what would otherwise be an elliptic system. One example is the restriction to divergence-free vectors in fluid dynamics (linear or nonlinear), and there are various other examples from partial differential equations. Also minimization problems in general, when involving linear or nonlinear equality constraints, provide further examples. In all these cases the coefficient matrix of the linear model has the form

$$A = \begin{bmatrix} G & -C \\ -C^T & 0 \end{bmatrix} . \qquad (1.1)$$

It is usually possible to augment the quadratic objective function (see (2.5) below) with a penalty term, so that the submatrix G is positive definite. However the matrix A is clearly indefinite (that is neither positive nor negative semi-definite) except in trivial cases.

It is usually possible to apply the conjugate gradient algorithm to indefinite systems, the difficulty being the possibility of division by zero, and hence breakdown of the algorithm. There is also the possibility of substantial error growth when this situation is almost achieved. This paper will review some suggestions which have been made for modifying the conjugate gradient method, or proposing similar types of method, to deal with symmetric indefinite problems. Attention is restricted to deriving methods which do not break down, and which can be implemented using only a few vectors of storage. One possible approach is the hyperbolic pairs method of Luenberger [5], which is shown to be unsatisfactory as it stands. A new formulation is proposed which reduces to the hyperbolic pairs method in the limiting case; nonetheless the best way of applying the method is not entirely clear, and there are reasons to think that the stability problem is not completely solved.

Another type of method is that in which the sum of squares of residuals is minimized. Two ways which have been proposed for doing this are described, and it is shown that one is to be preferred. It provides a viable algorithm for application to indefinite problems. However the algorithm is not entirely convenient for application to certain types of problem. The algorithm is also shown to be directly related to others which have been proposed, in particular the method of biconjugate gradients. This latter method is described in some detail.

It is also shown that another different algorithm can be derived from the biconjugate gradient algorithm, and which is suitable for indefinite systems. Relationships are stated which exist between these methods derived from biconjugate gradients, and the standard conjugate gradient method. In addition some recent work by Paige and Saunders [6] is reviewed, in which a new method is generated, based on the explicit use of the orthogonal reduction of a tridiagonal matrix to upper triangular form. The formulae which are entailed in using this method are quite complicated. However it is shown that Paige and Saunders' method is equivalent to the second method based on the biconjugate gradient algorithm, although the latter provides the more convenient recurrence relations. Finally a few other possibilities which have been suggested, and which are potentially suitable for the indefinite problem are discussed. None of them are currently felt to be as promising as the two methods derived from biconjugate gradients, for this type of application.

2. The Conjugate Gradient Method

The conjugate gradient method can be applied to solving the system $Ax = b$, where $x,b \in R^n$ and where A is symmetric. The further assumption that A is positive definite guarantees that the algorithm is well defined, apart from (important) considerations of round-off error. The method involves recurrence relationships, so it is convenient to use $x_1, x_2, \ldots$ to denote successive iterates. It is also convenient to define a residual $r(x) \equiv b - Ax$, and r_i is often used to refer to $r(x_i)$, although this usage is corrupted at certain times. The algorithm is described in detail by Hestenes and Stiefel [2], and can be stated in various equivalent forms. All require x_1 and hence $r_1 \equiv r(x_1)$ to be given, and set $p_1 = r_1$. Then for $k = 1,2,\ldots$, recurrence relations are used, one form of which is

$$x_{k+1} = x_k + \alpha_k p_k \tag{2.1a}$$

$$r_{k+1} = r_k - \alpha_k A p_k \tag{2.1b}$$

where

$$\alpha_k = r_k^T r_k / p_k^T A p_k \tag{2.1c}$$

and

$$p_{k+1} = r_{k+1} + \beta_k p_k \qquad (2.1d)$$

where

$$\beta_k = r_{k+1}{}^T r_{k+1} / r_k{}^T r_k \,. \qquad (2.1e)$$

The algorithm terminates in exact arithmetic in at most n iterations (m say) with $r_{m+1} = 0$ and $\beta_m = 0$. Four storage vectors are required to implement the method on a computer. The properties of the method will be stated here without explicit proof; (proof is given in [2] and also in section 5 as a special case of the proof of the biconjugate gradient algorithm.) If the Krylov sequence is defined as

$$\{r_1, Ar_1, A^2 r_1, \ldots, A^{k-1} r_1\} \,, \qquad (2.2)$$

then for $k \leq m$ this is a basis for a linear space S_k. Then r_k and p_k are in S_k but not S_{k-1}, the vector r_k satisfying orthogonality conditions

$$r_k{}^T r_j = r_k{}^T p_j = 0 \qquad \text{for all} \quad j < k \,, \qquad (2.3)$$

and the vector p_k satisfying conjugacy conditions

$$p_k{}^T A p_j = 0 \qquad \text{for all} \quad j < k \,. \qquad (2.4)$$

The vectors $r_1, r_2, \ldots, r_k$ are therefore an orthogonal basis for S_k.

An alternative interpretation is that $-r(x)$ is the gradient of the quadratic function

$$Q(x) = \tfrac{1}{2} x^T A x - b^T x \qquad (2.5)$$

and p_{k+1} is the component of r_{k+1} which is conjugate to p_k - hence the term conjugate gradients. Also the choice of α_k is that which makes $Q(x_k + \alpha p_k)$ stationary with respect to distance α along a direction of search p_k. Thus it is possible to rearrange equations (2.1) so that Ap_k is not evaluated explicitly, but r_{k+1} is calculated as $r(x_{k+1})$ and α_k is obtained by solving the equation $r(x_{k+1})^T p_k = 0$. In this interpretation the extension to general minimization problems becomes apparent. Of course for nonlinear problems the termination property no longer holds. However this result is largely illusory since the method suffers from build up of round-off error when k becomes large. Also in application to very large systems, even carrying out n iterations can be prohibitive. However, regarding the method as an iterative method (for linear or nonlinear systems), it has been observed that for certain systems, satisfactory convergence occurs in much fewer than n iterations. There are good reasons for this when the system satisfies certain symmetry properties (see Reid [7], for example), and it is under these circumstances that conjugate gradients is viable. However it is largely a matter of trial and error to find out whether any given system can be solved rapidly.

Further useful properties relating to conjugate gradients are obtained by defining the matrices ($\|.\| \equiv \|.\|_2$)

$$\mathcal{R}_k = \left[\frac{r_1}{\|r_1\|}, \frac{r_2}{\|r_2\|}, \ldots, \frac{r_k}{\|r_k\|}\right] \tag{2.6}$$

$$\mathcal{P}_k = \left[\frac{p_1}{\|r_1\|}, \frac{p_2}{\|r_2\|}, \ldots, \frac{p_k}{\|r_k\|}\right] \tag{2.7}$$

$$\mathcal{L}_k = \begin{bmatrix} 1 & & & & \\ -\beta_1^{\frac{1}{2}} & 1 & & & \\ & -\beta_2^{\frac{1}{2}} & 1 & & \\ & & \ddots & \ddots & \\ & & & -\beta_{k-1}^{\frac{1}{2}} & 1 \end{bmatrix} \tag{2.8}$$

and

$$\mathcal{D}_k = \begin{bmatrix} \alpha_1^{-1} & & & \\ & \alpha_2^{-1} & & \\ & & \ddots & \\ & & & \alpha_k^{-1} \end{bmatrix} \tag{2.9}$$

Note that $\mathcal{R}_n$ is an orthogonal matrix if $m = n$. Using the fact that $r_{m+1} = 0$, the recurrence relation (2.1b) can be written

$$A\mathcal{P}_m\mathcal{D}_m^{-1} = \mathcal{R}_m\mathcal{L}_m \ , \tag{2.10}$$

and using $p_1 = r_1$, (2.1d) can be written

$$\mathcal{P}_m\mathcal{L}_m^T = \mathcal{R}_m \ . \tag{2.11}$$

Eliminating $\mathcal{P}_m$ from (2.10), and defining $T_k = \mathcal{L}_k\mathcal{D}_k\mathcal{L}_k^T$, gives the equation

$$A\mathcal{R}_m = \mathcal{R}_m\mathcal{L}_m\mathcal{D}_m\mathcal{L}_m^T = \mathcal{R}_m T_m \tag{2.12}$$

showing that when $m = n$, $\mathcal{R}_n$ provides an orthogonal transformation of A to symmetric tridiagonal form T_n. The equations implicit in (2.12) were used by Lanczos [3] as the basis of a method for finding eigenvalues of A. Any normalized vector r_1 is chosen (not $r(x_1)$ as when solving equations) and the three term recurrence implicit in (2.12) (rearranged to give r_{k+1} on the left) is carried out for all k (assuming $m = n$). Then the elements of T_n and hence the eigenvalues of A can be determined. Note that any subsequent determination of eigenvectors by inverse iteration might require the solution of a symmetric indefinite system of linear equations. In what follows, it is also convenient to have an expression for $A\mathcal{R}_k$ for an intermediate stage $k < m$. Let the extension $\tilde{\mathcal{L}}_k$ of $\mathcal{L}_k$ be defined as the first k columns of $\mathcal{L}_{k+1}$ (so that $(\tilde{\mathcal{L}}_k; e_{k+1}) \equiv \mathcal{L}_{k+1}$ where e_i denotes the i-th coordinate vector). Then (2.1b) becomes

$$A P_k D_k^{-1} = R_{k+1} L_k \tag{2.13}$$

and using (2.11), then

$$A R_k = R_{k+1} L_k D_k L_k^T = R_{k+1} \tilde{T}_k \tag{2.14}$$

say, where $\tilde{T}_k$ is the extension of T_k. The result that

$$x_{k+1} - x_1 = \sum_{i=1}^{k} \alpha_i p_i$$

$$= R_k T_k^{-1} e_1 \|r_1\| \tag{2.15}$$

can be derived readily, using (2.11) and the definition of T_k.

In applying conjugate gradients to elliptic systems, A is positive definite, so all the matrices T_k are positive definite, and there is no difficulty with the nonexistence of the factors L_k and D_k, or of the iterates x_k. However when A is indefinite, then although T_k exists for all k, T_k^{-1} and hence x_{k+1} may not exist or may be large, and also the factors may not exist for certain k, even when T_k is non-singular. The search for new methods may therefore be regarded as looking for different simple factorizations of either A or T_n.

3. Hyperbolic pairs

The precise way in which the conjugate gradient algorithm can fail for an indefinite system is that in calculating α_k by (2.1c) the denominator satisfies

$$p_k^T A p_k = 0 . \tag{3.1}$$

Luenberger [5] has shown that the algorithm can be rescued when this occurs, by exploiting the concept of a hyperbolic pair. He defines a vector p_{k+1} as a linear combination of Ap_k and p_k such that

$$p_{k+1}^T A p_{k+1} = 0 \tag{3.2}$$

and (3.1) and (3.2) constitute the definition of p_k and p_{k+1} as a hyperbolic pair. Then scalars α_k and α_{k+1} can be calculated by formulae different from (2.1c) giving $x_{k+2} = x_k + \alpha_k p_k + \alpha_{k+1} p_{k+1}$ (see (3.4) below), with an analogous formula for r_{k+2}. A formula is also given for p_{k+2} which therefore enables the algorithm to be continued using the recurrence relations (2.1).

Luenberger's algorithm is unsuitable in general because it does not adequately cope with the situation when $p_k^T A p_k$ is small (ε say) but not zero. Then x_{k+1} is still calculated by (2.1a) and can be arbitrarily large. This will introduce relatively large rounding errors into the subsequent calculation.

It is also bad in a nonlinear situation, since the information contained in r_{k+1} is worthless. However it will now be shown that further modifications can be introduced into the algorithm to cater for this situation to some extent. It is easy to verify that when $p_k^T A p_k$ tends to zero, then p_{k+1} tends to p_k in direction. In fact although the point x_{k+1} tends to infinity, the point x_{k+2} is well-defined in the limit. It is therefore possible to rearrange the computation so that x_{k+2} is calculated directly from x_k, and so that the need to compute x_{k+1}, r_{k+1} and p_{k+1} is removed. To do this it is convenient to write

$$\bar{r}_{k+1} = \alpha_k^{-1} r_{k+1} = \alpha_k^{-1} r_k - A p_k \,. \tag{3.3a}$$

Then it can be verified that

$$x_{k+2} - x_k = -\frac{r_k^T r_k}{\bar{r}_{k+1}^T \bar{r}_{k+1}} \left\{ \frac{\bar{r}_{k+1}^T A \bar{r}_{k+1}}{\bar{r}_{k+1}^T \bar{r}_{k+1}} p_k + \bar{r}_{k+1} \right\} / (1-\mu_k) \tag{3.3b}$$

where

$$\mu_k = \alpha_k^{-1} \left(\frac{r_k^T r_k}{\bar{r}_{k+1}^T \bar{r}_{k+1}} \right) \left(\frac{\bar{r}_{k+1}^T A \bar{r}_{k+1}}{\bar{r}_{k+1}^T \bar{r}_{k+1}} \right) \,. \tag{3.3c}$$

In addition it also can be shown that

$$p_{k+2} = r_{k+2} + \frac{r_{k+2}^T r_{k+2}}{r_k^T r_k} \left\{ \frac{p_k^T A p_k}{\bar{r}_{k+1}^T \bar{r}_{k+1}} \bar{r}_{k+1} + p_k \right\} . \tag{3.3d}$$

Thus if it is decided that $p_k^T A p_k$ is too small to permit the recurrences in (2.1) to be used in a stable fashion, the alternative is to calculate $\bar{r}_{k+1}$ by (3.3a). Then $A\bar{r}_{k+1}$ must be calculated, in which case μ_k, x_{k+2}, $r_{k+2} \equiv r(x_{k+2})$ and p_{k+2} follow. To use this algorithm in a nonlinear situation without A being explicitly available, requires an additional search along $\bar{r}_{k+1}$ to calculate $A\bar{r}_{k+1}$. When $p_k^T A p_k = 0$, then $\alpha_k^{-1} = 0$ and hence $\bar{r}_{k+1} = -Ap_k$ and $\mu_k = 0$. Then (3.3b) and (3.3d) reduce to

$$x_{k+2} - x_k = \frac{r_k^T r_k}{p_k^T A^2 p_k} \left\{ A p_k - \frac{p_k^T A^3 p_k}{p_k^T A^2 p_k} p_k \right\} \tag{3.4}$$

and

$$p_{k+2} = r_{k+2} + \frac{r_{k+2}^T r_{k+2}}{r_k^T r_k} p_k \,. \tag{3.5}$$

Although these formulae are more simple than those given by Luenberger, it is not difficult to verify that they are equivalent.

One difficult with using this algorithm is that of deciding when $p_k^T A p_k$ is sufficiently small to merit using the formulae (3.3), rather than (2.1). However the algorithm has apologies with the algorithm of Bunch and Parlett [1] for factorizing a symmetric matrix efficiently using either 1×1 or 2×2

symmetric block diagonal pivots (corresponding to the one step (2.1) or two step (3.3) formulae here). In this case the symmetric matrix in question is T_n and the algorithm has to be applied without interchanges. With this restriction in mind, it might be possible to adapt the test of Bunch and Parlett to work here. Unfortunately using 1×1 or 2×2 pivots to factorize a tridiagonal matrix without interchanges is still potentially unstable to some extent. Because of this the methods developed in sections 4 and 5 seem preferable.

4. Least squares and minimum residuals

Another possible approach towards determining a stable algorithm is to attempt to minimize the sum of squares of residuals, that is $r(x)^T r(x)$. In the linear case the necessary and sufficient conditions for this are that the normal equations of least squares, $A^2x = Ab$, are satisfied, when A is symmetric and non-singular. It is possible to apply the algorithm (2.1) to this system, since A^2 is positive definite. However there are two disadvantages to this approach. One is that two multiplications by A are required per iteration, which roughly doubles the amount of work, since there is no reason to expect fewer iterations to be required. The other reason is that severe ill-conditioning often becomes apparent. This is related to the fact that the Krylov sequence for the least squares system is $\{Ar_1, A^3r_1, A^5r_1, \ldots\}$, which is much harder to resolve numerically into orthogonal vectors than the sequence $\{r_1, Ar_1, A^2r_1, \ldots\}$ for conjugate gradients applied to $Ax = b$, by analogy with the power method for computing eigenvalues of A.

However Hestenes and Stiefel [2] point out that there exist vectors $\{x_k\}$ say, in the convex hull of the iterates, $\{x_k^{CG}\}$ generated by the conjugate gradient method (2.1), whose residuals r_k minimize r^Tr for r in the space $\{r_1^{CG}, Ar_1^{CG}, A^2r_1^{CG}, \ldots, A^{k-1}r_1^{CG}\}$, and which satisfy a simple recurrence. Choosing this sequence of vectors $\{x_k\}$ provides an algorithm which does not suffer from the disadvantages above, and which is termed the minimum residual algorithm. The same algorithm can be derived in different ways by using the generalization of conjugate gradients given by Rutishauser [8] in which the general inner product (p,q) is defined as $p^TA^\mu q$, for any integer μ. In this case $\mu = 1$ gives the minimum residual algorithm. This is easy to see since the conditions $(r_k^Tp_j) = 0$ for $j < k$, become $r_k^TAp_j = 0$ when the innerproduct is redefined with $\mu = 1$. But the gradient $\nabla(r^Tr) = -2Ar$ by definition of $r(x)$, so it follows that $\nabla(r^Tr)|_{x_k}$ is orthogonal to p_j for all $j < k$, and so r^Tr is minimized in the space spanned by these $\{p_j\}$. Since only the innerproduct is redefined, it is easy to verify that this space is the same as in standard conjugate gradients.

For indefinite systems the minimum residual algorithm is not stable, in the form given on redefining the innerproduct and using (2.1). This is because $(r_k,r_k) \equiv r_k^T A r_k$ can become zero in the definition of β_k. However this difficulty can be overcome by rearranging the algorithm suitably as will be demonstrated in section 5. In this rearranged form, the algorithm requires 6 vectors of storage for a computer program. One disappointing feature of the minimum residual algorithm is that it is not entirely convenient when A is not available (such as in applications to minimization subject to constraints). This is because in determining $p_{k+1} = r_{k+1} + \beta_k p_k$, the computation of $\beta_k = r_{k+1}^T A r_{k+1} / r_k^T A r_k$ (or its equivalent when the algorithm is rearranged) requires Ar_{k+1} which is not available, so has to be obtained by searching along r_{k+1}. Thus each iteration $(k > 1)$ requires a search first along r_{k+1} (or alternatively Ap_k), before the search along p_{k+1} can be made.

5. Biconjugate Gradients

In the search for a suitable algorithm for solving indefinite systems it is fruitful to examine a different generalization of conjugate gradients. This is the biconjugate gradient algorithm, used by Lanczos [3] as a means of computing the eigenvalues of an unsymmetric matrix A. In this form, two vectors r_1 and $\bar{r}_1$ are given, and letting $p_1 = r_1$ and $\bar{p}_1 = \bar{r}_1$, then for $k = 1,2,\ldots$, the recurrence relations

$$r_{k+1} = r_k - \alpha_k A p_k \tag{5.1a}$$

$$\bar{r}_{k+1} = \bar{r}_k - \alpha_k A^T \bar{p}_k \tag{5.1b}$$

where

$$\alpha_k = \bar{r}_k^T r_k / \bar{p}_k^T A p_k , \tag{5.1c}$$

and

$$p_{k+1} = r_{k+1} + \beta_k p_k \tag{5.1d}$$

$$\bar{p}_{k+1} = \bar{r}_{k+1} + \beta_k \bar{p}_k \tag{5.1e}$$

where

$$\beta_k = \bar{r}_{k+1}^T r_{k+1} / \bar{r}_k^T r_k , \tag{5.1f}$$

are used. The scalar α_k is chosen so as to force the biorthogonality conditions $r_{k+1}^T \bar{r}_k = \bar{r}_{k+1}^T r_k = 0$ and β_k is chosen to force the biconjugacy condition $\bar{p}_{k+1}^T A p_k = p_{k+1}^T A^T \bar{p}_k = 0$. It is a feature of the algorithm (analogous to the algorithm (2.1)) that these conditions also hold for any pair of vectors without having to be explicitly enforced. This can be stated formally as follows.

Theorem Assuming that the biconjugate gradient algorithm (5.1) does not break down, then for all $j < i$,

$$\bar{r}_i^T r_j = r_i^T \bar{r}_j = 0 \tag{5.2}$$

and

$$\bar{p}_i^T A p_j = p_i^T A^T \bar{p}_j = 0 \quad . \tag{5.3}$$

Proof Clearly the theorem holds for $i = 1$ since no condition is implied. For $i > 1$ an inductive argument is used, assuming that the theorem is true for i, and proving it when i is replaced by $i+1$. Firstly by (5.1a)

$$r_{i+1}^T \bar{r}_j = r_i^T \bar{r}_j - \alpha_i p_i^T A^T \bar{r}_j = r_i^T \bar{r}_j - \alpha_i p_i^T A^T (\bar{p}_j - \beta_{j-1} \bar{p}_{j-1}) \qquad (5.4)$$

by (5.1e), assuming $\beta_0 = 0$ if necessary. For $j=i$, it follows from (5.3) and (5.1c) that $r_{i+1}^T \bar{r}_i = 0$. For $j < i$ in (5.4), it follows from (5.2) and (5.3) that $r_{i+1}^T \bar{r}_j = 0$, so this result holds for all $j < i+1$. In a like manner $\bar{r}_{i+1}^T r_j = 0$ can be demonstrated so it follows that

$$\bar{r}_{i+1}^T r_j = r_{i+1}^T \bar{r}_j = 0 \qquad \text{for all} \quad j < i+1, \qquad (5.5)$$

which is (5.2) with i replaced by $i+1$.

It also follows from (5.1d) and (5.1b) that

$$p_{i+1}^T A^T \bar{p}_j = r_{i+1}^T A^T \bar{p}_j + \beta_i p_i^T A^T \bar{p}_j$$

$$= r_{i+1}^T (\bar{r}_j - \bar{r}_{j+1})/\alpha_j + \beta_i p_i^T A^T \bar{p}_j \, . \qquad (5.6)$$

When $j = i$, using (5.1c) gives

$$p_{i+1}^T A^T \bar{p}_i = r_{i+1}^T (\bar{r}_i - \bar{r}_{i+1})/\alpha_i + \beta_i \, r_i^T \bar{r}_i/\alpha_i \, .$$

Then by (5.5) and (5.1f), $p_{i+1}^T A^T \bar{p}_i = 0$. When $j < i$ in (5.6), then using (5.5) and (5.3) gives $p_{i+1}^T A^T \bar{p}_j = 0$, so this result holds for all $j < i+1$. Likewise $\bar{p}_{i+1}^T A p_j = 0$ can be established for all $j < i+1$, and these results extend (5.3) when i is replaced by $i+1$.

Q.E.D.

A direct consequence of (5.1d) is that $p_{k+1} = r_{k+1} + \beta_k r_k + \beta_k \beta_{k-1} r_{k-1} + \cdots$, so that (5.2) also implies that

$$r_i^T \bar{p}_j = \bar{r}_i^T p_j = 0 \qquad (5.7)$$

for $j < i$. Another simple corollary is that the algorithm must terminate with $r_{m+1} = \bar{r}_{m+1} = 0$ in at most n iterations, using the same argument as for the standard conjugate gradient method. It may seem unusual to state these results in detail for the biconjugate gradient algorithm but not for other conjugate gradient methods. However as will be demonstrated, almost all algorithms which are considered here are special cases of this algorithm, so these results carry over immediately.

In the form (5.1) the algorithm provides a similarity reduction of A to tridiagonal form, suitable for calculating eigenvalues. To solve a system of equation $Ax = b$, it is necessary to augment (5.1). To do this r_1 is chosen to be the residual $r(x_1)$, where x_1 is an initial estimate of the solution. Then the recurrence relation

$$x_{k+1} = x_k + \alpha_k p_k \qquad (5.8)$$

is carried out together with (5.1), and since $r_{m+1} = 0$, it follows that x_{m+1} solves the equations. For general unsymmetric A there is no guarantee that the algorithm will not break down or not be unstable.

In the applications of interest here, A is a symmetric matrix. In this case the choice of $\bar{r}_1 = r_1$ leads to the standard conjugate gradient algorithm (2.1), and $\bar{r}_k = r_k$, $\bar{p}_k = p_k$ for all k. However in this section the main interest will lie in examining the consequences of making the choice

$$\bar{r}_1 = Ar_1 . \tag{5.9}$$

It then follows easily when A is symmetric that

$$\bar{r}_k = Ar_k , \qquad \bar{p}_k = Ap_k \tag{5.10}$$

for all k. Since all innerproducts in (5.1) are between either $(\bar{r}_k, r_k)$ or $(\bar{p}_k, Ap_k)$, it follows from (5.10) that the resulting algorithm is equivalent to the standard conjugate gradient algorithm but with (a,b) defined as a^TAb. Thus the biconjugate gradient algorithm (5.1) and (5.8), with the choice (5.9), is equivalent to the minimum residual algorithm described in section 4. That is to say the vectors x_k, r_k and p_k in (5.8), (5.1a) and (5.1d) recur as in the minimum residual algorithm. It is also possible to show (see section 6) that the vectors r_k are proportional to the search directions (p_k^{CG} say) which arise in the standard conjugate gradient algorithm (2.1).

Before pursuing these ideas further, it is noted that the choices $\bar{r}_1 = r_1$ and $\bar{r}_1 = Ar_1$ are special cases of a more general class of methods. Consider choosing $\bar{r}_1 = Mr_1$ where M is any square matrix. Then a sufficient condition for the properties $\bar{r}_k = Mr_k$, $\bar{p}_k = Mp_k$ to persist is that

$$A^TM = MA \tag{5.11}$$

This is easily proved by induction. When A is symmetric, (5.11) is the commutation condition $AM = MA$. The resulting method is then equivalent to using the generalized innerproduct $(a,b) \equiv a^TMb$ in (2.1). An example of a matrix which commutes is $M = A^{\mu}$, which has already been referred to, but it can be seen for instance that any matrix polynomial $M = \Pi(A)$ also commutes. A choice of possible interest might be the general linear polynomial $M = \lambda A + \mu I$.

Attention will henceforth be restricted to the case when A is symmetric and the choice $\bar{r}_1 = Ar_1$ is made. It will be noticed that the minimum residual algorithm makes no reference to the vectors $\{\bar{p}_k\}$ which are generated in the biconjugate gradient algorithm. Furthermore these vectors are mutually orthogonal, since the conditions (5.10) and the biconjugacy conditions (5.3) imply that $\bar{p}_i^T\bar{p}_j = \bar{p}_i^TAp_j = 0$, when $j \neq i$. It is interesting therefore to consider whether any further algorithms can be developed, in which the vectors $\bar{p}_k$ are used as

search directions, especially since stability might be expected to arise from the orthogonality property. Consider then introducing a new sequence of iterates $\{\hat{x}_k\}$ with $\hat{x}_1 = x_1$, obtained by searching the directions $\bar{p}_k$, that is by

$$\hat{x}_{k+1} = \hat{x}_k + \hat{\alpha}_k \bar{p}_k \tag{5.12a}$$

where $\hat{\alpha}_k$ is yet to be determined. Then the residuals $\hat{r}_k \equiv r(\hat{x}_k)$ recur like

$$\hat{r}_{k+1} = \hat{r}_k - \hat{\alpha}_k A\bar{p}_k \,. \tag{5.12b}$$

There is no difficulty in carrying out these recurrences in conjunction with those given by (5.1). Furthermore it is possible to choose the $\{\hat{\alpha}_k\}$ so that the sequence $\{\hat{x}_k\}$ terminates at the solution and $\{\hat{r}_k\}$ at zero. In fact $\hat{\alpha}_k$ is chosen so that $\hat{r}_{k+1}^T r_k = 0$, giving the equation

$$\hat{\alpha}_k = \hat{r}_k^T r_k / \bar{p}_k^T A r_k \tag{5.13}$$

or equivalently, since $r_k = p_k - \beta_{k-1} p_{k-1}$ and using (5.3), by

$$\hat{\alpha}_k = \hat{r}_k^T r_k / \bar{p}_k^T \bar{p}_k \,. \tag{5.14}$$

A simple inductive proof shows that $\hat{r}_i^T r_j = 0$ for all $j < i$. First of all, $\hat{r}_{i+1}^T r_i = 0$ by choice of $\hat{\alpha}_i$. But for all $j < i$, $\hat{r}_{i+1}^T r_j = \hat{r}_i^T r_j - \hat{\alpha}_i \bar{p}_i^T A r_j = -\hat{\alpha}_i \bar{p}_i^T A r_j$, and eliminating $A\bar{p}_i$ using (5.1b) and using (5.2), shows that this scalar product is zero. Thus $\hat{r}_{i+1}^T r_j = 0$ for all $j < i+1$ which completes the induction. Now $\hat{r}_{n+1}^T r_j = 0$ for all $j \leq n$, implies that $\hat{r}_{n+1} = 0$, so it follows that the sequence $\{\hat{x}_k\}$ terminates at the solution in at most n iterations. Thus another algorithm has been derived from the biconjugate gradient algorithm. This new algorithm uses the recurrences (5.12a,b) and those for r_k, $\bar{r}_k$ and $\bar{p}_k$ from (5.1); note that p_k is not required.

In this form however, the algorithm fails when $\bar{r}_k^T r_k = 0$ (as also does the minimum residual algorithm). However the difficulties can be avoided by eliminating $\bar{r}_{k+1}$ from (5.1e) using (5.1b), and then eliminating $\bar{r}_k$ from the resulting expression using (5.1e). This gives $\bar{p}_{k+1}$ as a linear combination of $A\bar{p}_k$, $\bar{p}_k$ and $\bar{p}_{k-1}$, which on redefining the length of $\bar{p}_{k+1}$, can be written as

$$\bar{p}_{k+1} = A\bar{p}_k - \lambda_k \bar{p}_k - \mu_k \bar{p}_{k-1} \,. \tag{5.15}$$

By virtue of the orthogonality of the $\{\bar{p}_k\}$, the simple expressions

$$\lambda_k = \bar{p}_k^T A\bar{p}_k / \bar{p}_k^T \bar{p}_k \tag{5.16}$$

and

$$\mu_k = \bar{p}_{k-1}^T A\bar{p}_k / \bar{p}_{k-1}^T \bar{p}_{k-1} = \bar{p}_k^T \bar{p}_k / \bar{p}_{k-1}^T \bar{p}_{k-1} \tag{5.17}$$

follow readily. In this form, the algorithm requires the recurrences for $\hat{x}_k$, $\hat{r}_k$, $\bar{p}_k$ and r_k. A similar expression

$$p_{k+1} = Ap_k - \lambda_k p_k - \mu_k p_{k-1} \tag{5.18}$$

can be used to stabilize the minimum residual algorithm. However the expression (5.13) for $\hat{\alpha}_k$ is still potentially unsatisfactory. Therefore eliminating r_k using $r_k = r_{k-1} - \alpha_{k-1}\bar{p}_{k-1}$, and simplifying yields $\hat{\alpha}_k = \hat{r}_k^T\bar{p}_{k-1}/\bar{p}_k^T\bar{p}_k$ $(k>1)$. It now transpires that the recurrence r_k is no longer required in the algorithm, so an even more simple algorithm can be stated.

Given $\hat{x}_1$, set $\hat{r}_1 = r(\hat{x}_1)$, $\bar{p}_1 = A\hat{r}_1$ and let $\mu_1 = 0$ and $\bar{p}_0 = \hat{r}_1$ to allow for the initial special cases. Then repeat for $k = 1,2,\ldots$ the following

$$\hat{\alpha}_k = \hat{r}_k^T\bar{p}_{k-1}/\bar{p}_k^T\bar{p}_k \tag{5.19a}$$

$$\hat{x}_{k+1} = \hat{x}_k + \hat{\alpha}_k\bar{p}_k \tag{5.19b}$$

$$\hat{r}_{k+1} = \hat{r}_k - \hat{\alpha}_k A\bar{p}_k \tag{5.19c}$$

$$\lambda_k = \bar{p}_k^T A\bar{p}_k/\bar{p}_k^T\bar{p}_k \tag{5.19d}$$

$$\mu_k = \bar{p}_k^T\bar{p}_k/\bar{p}_{k-1}^T\bar{p}_{k-1} \tag{5.19e}$$

$$\bar{p}_{k+1} = A\bar{p}_k - \lambda_k\bar{p}_k - \mu_k\bar{p}_{k-1} \tag{5.19f}$$

This algorithm will be called the orthogonal direction algorithm. It requires five vectors of storage and is stable. Also the errors in the $\{\hat{x}_k\}$ decrease monotonically due to the orthogonality of the $\{\bar{p}_k\}$. The algorithm adapts well as a search algorithm for nonlinear problems, since $\hat{r}_{k+1}$ can alternatively be defined as $r(\hat{x}_{k+1})$, and then $A\bar{p}_k$ computed from differences in $\hat{r}_{k+1}$ and $\hat{r}_k$.

There are some unanswered questions about implementation, for instance about propagation of round-off error and practical conditions for termination. To this extent it might be convenient to recur the minimum residual vector r_k, at the expense of a vector of storage. Alternatively certain collinearity properties which exist between the orthogonal direction algorithm, the minimum residual algorithm and the conjugate gradient algorithm (given in section 6) might be used. When the minimum residual vector is available, there are also certain alternatives for nonlinear problems in regard to calculating $\hat{\alpha}_k$, which might be explored.

6. Interrelationships

Recently, Paige and Saunders [6] suggested another new method based on the use of a QR factorization. In this section their algorithm is described and is shown to give the same sequence of iterates $\{\hat{x}_k\}$ as the algorithm (5.19) derived from biconjugate gradients. Interrelationships between this algorithm, the minimum residual algorithm and the conjugate gradient algorithm are then stated.

Consider the analysis of section 2, and in particular the extended tridiagonal matrix $\mathfrak{T}_k = \mathfrak{Q}_{k+1}^T A\,\mathfrak{Q}_k$ defined in the conjugate gradient method. Paige and

Saunders [6] base their method on a QR factorization of $\bar{T}_k$. If an orthogonal matrix Q_{k+1} is defined as the product of k plane rotations in the $(1,2), (2,3),\ldots,(k,k+1)$ planes

$$Q_{k+1} = \begin{bmatrix} c_1 & s_1 & & & \\ s_1 & -c_1 & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix} \cdots\cdots\cdots \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & & c_k & s_k \\ & & & s_k & -c_k \end{bmatrix}, \tag{6.1}$$

then it is possible to choose $c_1, c_2, \ldots, c_k$ to eliminate the subdiagonal elements of $\bar{T}_k$ in order, so that

$$Q_{k+1}^T \bar{T}_k = \begin{pmatrix} U_k \\ 0 \end{pmatrix} = \begin{bmatrix} x & x & x & & \\ & x & x & x & \\ & & x & x & x \\ & & & x & x \\ & & & & x \\ 0 & \cdots & \cdots & \cdots & 0 \end{bmatrix}. \tag{6.2}$$

Then

$$\mathcal{R}_{k+1} \bar{T}_k = \mathcal{R}_{k+1} Q_{k+1} \begin{pmatrix} U_k \\ 0 \end{pmatrix} = W_k U_k \tag{6.3}$$

where W_k is the first k columns of $\mathcal{R}_{k+1} Q_{k+1}$. Note that the columns of W_k are mutually orthogonal. Now (2.15) shows that the iterates $\{x_k^{CG}\}$ in conjugate gradients satisfy

$$x_{k+1}^{CG} - x_1^{CG} = \mathcal{R}_k T_k^{-1} e_1 \|r_1\|. \tag{6.4}$$

If $\mathcal{R}_k T_k^{-1}$ is replaced by $\mathcal{R}_{k+1} \bar{T}_k^{+T}$, then the solution x_{m+1}^{CG} is not affected, since $\bar{T}_m = \begin{pmatrix} T_m \\ 0 \end{pmatrix}$ so that $\bar{T}_m^{+T} = \begin{pmatrix} T_m^{-T} \\ 0 \end{pmatrix} = \begin{pmatrix} T_m^{-1} \\ 0 \end{pmatrix}$. However different intermediates, $\{x_k^{PS}\}$ say, are obtained, which always exist since $\bar{T}_k$ always has full rank. It is easily verified that $\bar{T}_k^{+T} = Q_{k+1} \begin{pmatrix} U_k^{-T} \\ 0 \end{pmatrix}$ so it follows that

$$x_{k+1}^{PS} - x_1^{PS} = W_k U_k^{-T} e_1 \|r_1\| = W_k z_k \tag{6.5}$$

say. Forward substitution in $U_k^T z_k = e_1 \|r_1\|$ gives one element of z_k (ζ_k say) at each iteration. Since W_{k-1} is a submatrix of W_k, it follows from (6.5) that

$$x_{k+1}^{PS} = x_k^{PS} + \zeta_k w_k \tag{6.6}$$

where w_k is the k^{th} column of W_k. This is the algorithm which Paige and Saunders suggest. It requires the three-term recurrence relationship derived from (2.12) to be used, which generates the columns of $\mathcal{R}_m$ directly from A. The computation of the elements $c_1, c_2, \ldots$ in the plane rotation matrices requires square roots, and this calculation and that in forming the elements of U_k^T are quite intricate. The process apparently can be carried out using only five storage vectors (see Paige and Saunders' subroutine in [6]), but this does not explicitly include the vector $r_k^{PS} \equiv r(x_k^{PS})$. Since the algorithm does not

introduce new information at each stage through Aw_k (but rather through the three-term recurrence relation for the columns of $\mathcal{R}_k$), it is not conveniently adapted as a search algorithm for nonlinear problems. However I must acknowledge a great debt to this work, since it made me re-examine the idea of searching the orthogonal directions $\{\tilde{p}_k\}$ in the biconjugate gradient algorithm, which I had previously abandoned.

In demonstrating the interrelationships between the various methods, it is first pointed out that the reduction (6.2) equivalently implies that $\tilde{\mathcal{L}}_k$ (if it exists) is reduced to upper bidiagonal form on premultiplication by Q_{k+1}^T. This is most conveniently written

$$Q_{k+1}^T \tilde{\mathcal{L}}_k \mathcal{D}_k^{\frac{1}{2}} = \begin{pmatrix} B_k \\ 0 \end{pmatrix} = \begin{bmatrix} x & x & & & \\ & x & x & & \\ & & x & x & \\ & & & x & x \\ & & & & x \\ 0 & & & & 0 \end{bmatrix} . \tag{6.7}$$

Consider now the minimum residual algorithm. The vectors r_k are uniquely defined (apart from length) by being in the linear space S_k (see section 2) and satisfying the conditions $r_k^T A r_j = 0$, for all $j < k$. Therefore in the biconjugate gradient algorithm, with A symmetric and $\tilde{r}_1 = Ar_1$; if the matrices

$$R_k = \left[\frac{r_1}{\rho_1}, \frac{r_2}{\rho_2}, \ldots, \frac{r_k}{\rho_k} \right] \tag{6.8a}$$

$$\tilde{R}_k = \left[\frac{\tilde{r}_1}{\rho_1}, \frac{\tilde{r}_2}{\rho_2}, \ldots, \frac{\tilde{r}_k}{\rho_k} \right] \tag{6.8b}$$

are defined, where $\rho_i = \sqrt{r_i^T r_i}$, then R_k and $\tilde{R}_k$ are uniquely defined by the conditions

$$AR_k = \tilde{R}_k \tag{6.9a}$$

$$\tilde{R}_k^T R_k = I \tag{6.9b}$$

$$r_k/\rho_k \in S_k \tag{6.9c}$$

for all k. It is readily verified that the definitions of R_k and $\tilde{R}_k$ given by the expressions

$$\mathcal{R}_k = R_k \mathcal{D}_k^{\frac{1}{2}} \mathcal{L}_k^T \tag{6.10a}$$

and

$$\tilde{R}_k = \mathcal{R}_{k+1} \tilde{\mathcal{L}}_k \mathcal{D}_k^{\frac{1}{2}} \tag{6.10b}$$

satisfy the conditions (6.9), and so relate the biconjugate gradient method to the conjugate gradient method.

Relationships can also be derived for the directions p_k and $\tilde{p}_k = Ap_k$ in the biconjugate gradient algorithm. Defining

$$P_k = \left[\frac{p_1}{\|\bar{p}_1\|}, \frac{p_2}{\|\bar{p}_2\|}, \ldots, \frac{p_k}{\|\bar{p}_k\|} \right] \tag{6.11a}$$

and

$$\bar{P}_k = \left[\frac{\bar{p}_1}{\|\bar{p}_1\|}, \frac{\bar{p}_2}{\|\bar{p}_2\|}, \ldots, \frac{\bar{p}_k}{\|\bar{p}_k\|} \right] = AP_k \tag{6.11b}$$

then by virtue of (5.1d,e) there exists an upper bidiagonal matrix, L_k^T say, such that

$$R_k = P_k L_k^T \tag{6.12a}$$

and

$$\bar{R}_k = \bar{P}_k L_k^T . \tag{6.12b}$$

Then it follows using the biorthogonality conditions that

$$\bar{R}_k^T A R_k = L_k \bar{P}_k^T A P_k L_k^T = L_k L_k^T . \tag{6.13}$$

But from (6.10),

$$\begin{aligned} \bar{R}_k^T A R_k &= \mathcal{D}_k^{\frac{1}{2}} \tilde{\mathcal{L}}_k^T \mathcal{R}_{k+1}^T A \mathcal{R}_k \mathcal{L}_k^{-T} \mathcal{D}_k^{-\frac{1}{2}} \\ &= \mathcal{D}_k^{\frac{1}{2}} \tilde{\mathcal{L}}_k^T \tilde{\mathcal{T}}_k \mathcal{L}_k^{-T} \mathcal{D}_k^{-\frac{1}{2}} \\ &= \mathcal{D}_k^{\frac{1}{2}} \tilde{\mathcal{L}}_k^T \tilde{\mathcal{L}}_k \mathcal{D}_k^{\frac{1}{2}} = B_k^T B_k \end{aligned} \tag{6.14}$$

from (6.7). Since Choleski factors are unique, it follows that

$$L_k^T = B_k . \tag{6.15}$$

Thus, using (6.12b), (6.10b) and (6.7) in turn,

$$\bar{P}_k = \bar{R}_k B_k^{-1} = \mathcal{R}_{k+1} \tilde{\mathcal{L}}_k \mathcal{D}_k^{\frac{1}{2}} B_k^{-1} = \mathcal{R}_{k+1} Q_{k+1} \begin{Bmatrix} I \\ 0 \end{Bmatrix} = W_k$$

showing the equivalence of the directions $\bar{p}_k$ in biconjugate gradients and the directions w_k in the Paige and Saunders method. Since the methods using the sequence $\hat{x}_k$ in section 5 start and terminate at the same points as the Paige and Saunders method, it follows that $\hat{x}_k = x_k^{PS}$ for all k.

Another feature which follows from equation (6.10a) is that by virtue of (2.11)

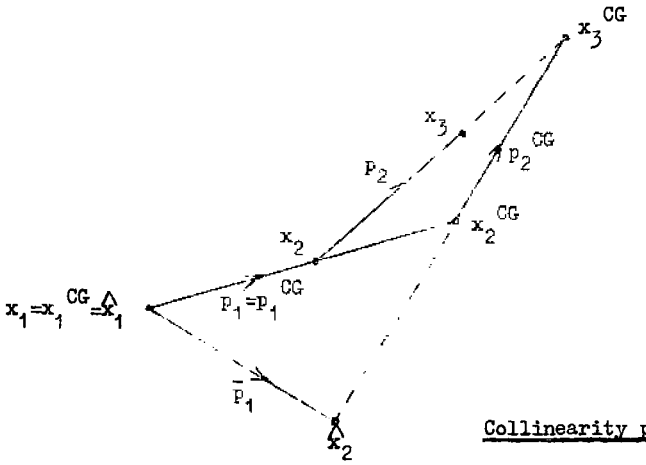
$$\mathcal{P}_k = R_k \mathcal{D}_k^{\frac{1}{2}} , \tag{6.16}$$

that is to say, the residuals r_k in the minimum residual algorithm are proportioned to the search directions p_k^{CG} in the conjugate gradient algorithm (2.1).

Finally two collinearity properties between vectors in different methods are described. The first is that both x_k, x_{k+1} and x_{k+1}^{CG} are collinear in the direction p_k of the minimum residual algorithm. This result is given for example by Hestenes and Stiefel [2] but is readily deduced inductively using the fact that both r_{k+1} and r_{k+1}^{CG} are orthogonal to $\bar{p}_{k-1}$. The second

property is that $\hat{x}_k$, x_k^{CG} and x_{k+1}^{CG} are collinear in the direction p_{k+1}^{CG} of of the conjugate gradient algorithm. This result is given by Paige and Saunders. These properties are illustrated in the figure.

Figure



Collinearity properties

Note that $\hat{x}_3$ cannot be drawn on this figure (assuming $m > 2$), but in three dimensions would be at the foot of a perpendicular dropped from $\hat{x}_2$ to the line joining x_3^{CG} and x_4^{CG}.

Collinearity properties can be important in the design of a practical algorithm; for instance it may be possible to recur expressions for $\|r_k^{CG}\|$ or $\|r_k\|$ whilst implementing the orthogonal directions algorithm. Then the algorithm could be terminated if either of $\|r_k^{CG}\|$ and $\|r_k\|$ (in addition to $\|\hat{x}_k\|$) were below a given tolerance. The corresponding value of x_k^{CG} or x_k would then be determined using the appropriate collinearity property.

7. Summary

Various conjugate gradient methods have been examined which can be applied generally to solving symmetric indefinite systems of linear or nonlinear equations. The modification of Luenberger's work has not been followed up because of the potential stability problems. This leaves two methods, the minimum residual algorithm and the orthogonal direction algorithm. Both of these have been shown to be special cases of the biconjugate gradient algorithm. The orthogonal direction algorithm has also been shown to be equivalent to the Paige and Saunders [6] algorithm, although the recurrences given here are quite different, and much more simple. It is thought that in the form given here, the orthogonal direction algorithm has wide potential application, both to linear and nonlinear problems. No substantial numerical experience is yet available, the recurrences merely having been verified on two small problems. However Paige and Saunders

[6] report good although somewhat limited computational experience with their algorithm.

Further work will include trying to apply the orthogonal directions algorithm to large scale partial differential equation problems. However I am conscious that no specific use of the structure in the matrix (1.1) is being used, and it might be profitable to take more account of this. In the context of this paper this might involve other stable reductions of $\tilde{T}_k$ to upper triangular form, or perhaps using a linear combination of p_k and $\bar{p}_k$ as a search direction. There is also a different conjugate gradient algorithm based on the existence of a reduction of A to the form PLQ^T, where P and Q are orthogonal and L is lower bidiagonal. Lawson [4] gives a list of references to this idea. Unfortunately this algorithm requires two multiplications by A per iteration. However it is not simply related to any of the algorithms in his paper, and might therefore be worth considering.

I would finally like to acknowledge the help of Dr T L Freeman and A S M Halliday in carefully reading the manuscript and pointing out a number of mistakes.

References

1. Bunch, J.R., and Parlett, B.N., Direct methods for solving symmetric indefinite systems of linear equations, S.I.A.M. J. Numer. Anal., Vol.8, 1971, pp.639-655.

2. Hestenes, M.R., and Stiefel, E., Methods of Conjugate Gradients for Solving Linear Systems, J. Res. Nat. Bur. Standards, Vol.49, 1952, pp.409-436.

3. Lanczos, C., An Iteration Method for the Solution of the Eigenvalue Problem of Linear Differential and Integral Operators, J. Res. Nat. Bur. Standards, Vol. 45, 1950, pp.255-282.

4. Lawson, C.L., Sparse Matrix Methods Based on Orthogonality and Conjugacy, Jet Propulsion Lab., Tech. Memo. 33-627, 1973.

5. Luenberger, D.G., Hyperbolic pairs in the method of conjugate gradients, S.I.A.M. J. Appl. Math., Vol.17, 1969, pp.1263-1267.

6. Paige, C.C., and Saunders, M.A., Solution of sparse indefinite systems of equations and least squares problems, Stanford University Report, STAN-CS-73-399, 1973.

7. Reid, J.K., On the Method of Conjugate Gradients for the Solution of Large Sparse Systems of Linear Equations, pp.231-254 in Large Sparse Systems of Linear Equations, ed. J.K. Reid, Academic Press, London, 1971.

8. Rutishauser, H., Theory of gradient methods, Chapter 2 of Refined iterative methods for computation of the solution and the eigenvalues of self-adjoint boundary value problems, by M. Engeli, Th. Ginsburg, H. Rutishauser, and E. Stiefel, Birkhäuser, Basel, 1959.