Mathematical Programming 12 (1977) 241–254.
North-Holland Publishing Company

RESTART PROCEDURES FOR THE CONJUGATE GRADIENT METHOD

M.J.D. POWELL*
A.E.R.E., Harwell, England

Received 9 December 1975
Revised manuscript received 27 September 1976

The conjugate gradient method is particularly useful for minimizing functions of very many variables because it does not require the storage of any matrices. However the rate of convergence of the algorithm is only linear unless the iterative procedure is "restarted" occasionally. At present it is usual to restart every n or $(n+1)$ iterations, where n is the number of variables, but it is known that the frequency of restarts should depend on the objective function. Therefore the main purpose of this paper is to provide an algorithm with a restart procedure that takes account of the objective function automatically. Another purpose is to study a multiplying factor that occurs in the definition of the search direction of each iteration. Various expressions for this factor have been proposed and often it does not matter which one is used. However now some reasons are given in favour of one of these expressions. Several numerical examples are reported in support of the conclusions of this paper.

1. The conjugate gradient algorithm

The problem to be solved by the algorithm is to calculate the least value of a general differentiable function of several variables. We let n be the number of variables, we let $\boldsymbol{x}$ be the vector of variables and we let $F(\boldsymbol{x})$ be the objective function. Some values of the gradient vector

$$\boldsymbol{g}(\boldsymbol{x}) = \nabla F(\boldsymbol{x}) \tag{1.1}$$

have to be calculated, but the conjugate gradient algorithm does not require any explicit second derivatives. It is an iterative method and iteration numbers are shown as subscripts. For example the sequence of points $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \ldots$ is calculated by the successive iterations and it should converge to the point in the space of the variables at which $F(\boldsymbol{x})$ is least.

The conjugate gradient algorithm was first applied to the general unconstrained minimization problem by Fletcher and Reeves [5]. However now there are several versions of the algorithm for this calculation. The Fletcher and Reeves method without restarts is as follows.

Let $\boldsymbol{x}_1$ be a given starting point in the space of the variables and let k denote the number of the current iteration starting with $k = 1$. The iteration requires the gradient

$$\boldsymbol{g}_k = \boldsymbol{g}(\boldsymbol{x}_k). \tag{1.2}$$

* Presently at University of Cambridge, Cambridge, England.

If $k = 1$, let $\boldsymbol{d}_k$ be the steepest descent direction

$$\boldsymbol{d}_k = -\boldsymbol{g}_k. \tag{1.3}$$

Otherwise, for $k > 1$, we apply the formula

$$\boldsymbol{d}_k = -\boldsymbol{g}_k + \beta_k \boldsymbol{d}_{k-1}, \tag{1.4}$$

where β_k has the value

$$\beta_k = \|\boldsymbol{g}_k\|^2 / \|\boldsymbol{g}_{k-1}\|^2, \tag{1.5}$$

the vector norms being Euclidean. We obtain $\boldsymbol{x}_{k+1}$ by searching for the least value of $F(\boldsymbol{x})$ from $\boldsymbol{x}_k$ along the direction $\boldsymbol{d}_k$. Thus $\boldsymbol{x}_{k+1}$ is the vector

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \lambda_k \boldsymbol{d}_k, \tag{1.6}$$

where λ_k is the value of λ that minimizes the function of one variable

$$\phi_k(\lambda) = F(\boldsymbol{x}_k + \lambda \boldsymbol{d}_k). \tag{1.7}$$

This completes the iteration, and another one is begun if $F(\boldsymbol{x}_{k+1})$ or $\boldsymbol{g}_{k+1}$ is not sufficiently small.

Note that the iteration does not require the calculation of any second derivatives. Note also that for $k \geq 2$ the definition of λ_{k-1} and equations (1.4) and (1.7) imply that the derivative $\phi_k'(0)$ has the negative value $-\|\boldsymbol{g}_k\|^2$. Therefore the numerical procedures that are used for the search along $\boldsymbol{d}_k$ assume that the required value of λ_k is positive.

The main justification of the conjugate gradient algorithm is its properties when $F(\boldsymbol{x})$ is a convex quadratic function

$$F(\boldsymbol{x}) = \tfrac{1}{2}\boldsymbol{x}^{\mathrm{T}} A \boldsymbol{x} + \boldsymbol{b}^{\mathrm{T}} \boldsymbol{x} + c, \tag{1.8}$$

say, where the superscript T distinguishes a row vector from a column vector. In this case it may be proved (see [5] for instance) that the search directions are mutually conjugate, which means that the equations

$$\boldsymbol{d}_i^{\mathrm{T}} A \boldsymbol{d}_j = 0, \quad i \neq j, \tag{1.9}$$

are satisfied. It follows that $\boldsymbol{x}_{k+1}$ gives the least value of $F(\boldsymbol{x})$ in the k-dimensional affine set that contains the starting point $\boldsymbol{x}_1$ and the search directions $\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_k$, so the algorithm obtains the least value of the function (1.8) in at most n iterations. Further, because the construction of the search directions makes the affine set contain the directions of the gradients $\boldsymbol{g}_1, \boldsymbol{g}_2, \ldots, \boldsymbol{g}_k$, it also follows that the sequence of gradients $\{\boldsymbol{g}_j: j = 1, 2, 3, \ldots\}$ is mutually orthogonal. One consequence of this remark is that equation (1.5) is equivalent to the definition

$$\beta_k = \boldsymbol{g}_k^{\mathrm{T}}[\boldsymbol{g}_k - \boldsymbol{g}_{k-1}] / \|\boldsymbol{g}_{k-1}\|^2. \tag{1.10}$$

Although expressions (1.5) and (1.10) are equivalent when $F(\boldsymbol{x})$ is a quadratic function, they give different results in the general case. Polak [8] prefers the formula (1.10) and this version of the conjugate gradient method is called the

Polak–Ribière algorithm. Fletcher [3] made some numerical experiments to compare these two expressions for β_k. His results for non-quadratic objective functions were inconclusive. However in Section 2 of this paper a numerical example is reported where formula (1.10) is much better than formula (1.5). Also some new theoretical reasons are given in favour of expression (1.10).

Although it is assumed in this paper that exact line searches can be done when $F(\boldsymbol{x})$ is a general function, in practice we should accept errors in λ_k. We prefer to ignore this complication because we think that it does not affect the main conclusions of this paper. In the numerical results reported later the relative accuracy in each value of λ_k is at least five decimals.

Fletcher and Reeves [5] recommend restarting the conjugate gradient method by using equation (1.3) instead of equation (1.4) every n or $(n+1)$ iterations. There seems to be general agreement that occasional restarting is very helpful in practice. Crowder and Wolfe [2] give an example to show that without restarting the rate of convergence can be only linear and Powell [9] shows that without restarts a linear rate of convergence is usual when there are more than two variables. Therefore it is useful to study restarting procedures.

A disadvantage of restarting by searching along the steepest descent direction is that the immediate reduction in the objective function is usually less than it would be without the restart. Therefore we would like to find a version of the conjugate gradient method that does not restrict the search direction of a restarting iteration to the steepest descent direction. This question is the subject of Section 3. We note that a technique proposed by Beale [1] is suitable, and also that the application of linear transformations to the variables allows the restart direction to be any downhill direction. In Beale's method expression (1.4) is augmented by a multiple of the most recent restarting direction, one consequence being that $\boldsymbol{d}_k$ may no longer be a downhill search direction when $F(\boldsymbol{x})$ is a general function. If this happens another restart is made. Although this may appear to be a disadvantage, it is in fact helpful, because it provides an automatic criterion for restarting which is invoked less often when $F(\boldsymbol{x})$ is close to a quadratic function.

These ideas lead to a new conjugate gradient algorithm which is described in Section 4. Its restart procedure requires two more vectors of storage than are used by the Polak–Ribière and Fletcher–Reeves algorithms. Some standard test problems are used to compare these three methods. They show that the new algorithm usually requires fewest iterations, sometimes the gain being more than a factor of two. Therefore the new method should be a useful addition to optimization algorithms. A Fortran listing is available from the Harwell subroutine library, the name of the subroutine being VA14A.

2. The value of β_k

It was mentioned in Section 1 that numerical results obtained by Fletcher [3] do not show that the value (1.10) is much better than the value (1.5) in the definition of the search direction (1.4), so when the work of the present paper was begun the

choice between these two formulae seemed to be open. However during this work it was found that formula (1.10) is preferable. The purpose of this section is to justify this statement.

One reason comes from a real minimization problem that Dr. M.J. Norgett of the Theoretical Physics Division at Harwell wished to solve. It has 165 variables, the objective function gives the energy of a model of an atomic system, and the least value gives the structure of the nuclei when a gas condenses. The first attempt to solve it was to apply the Fletcher–Reeves conjugate gradient algorithm that is in the Harwell subroutine library, namely subroutine VA08A, which uses the definition (1.5) for β_k. The three function values obtained by iterations 48, 49 and 50 were -216.616, -217.693 and -218.388 respectively, the required minimum value being -268.276, which was obtained by Dr. Norgett by forcing the subroutine to restart more often than every n iterations. Then, using the same starting point x_1, the Polak–Ribière algorithm was applied. The required minimum value was found to six decimals accuracy in only twelve iterations without any restarts. Differences between the two algorithms that are as great as this one are very uncommon, but the fact that they can occur should be noted.

The following remarks show why the Fletcher–Reeves algorithm may be inefficient for several iterations if a search direction d_k occurs that is almost orthogonal to the steepest descent direction $-g_k$. We let θ_k be the angle between d_k and $-g_k$. Figure 1 shows this angle, the definition (1.4) and the orthogonality of g_k to d_{k-1}. The figure is useful because it gives the equation

$$\|d_k\| = \sec \theta_k \|g_k\|. \tag{2.1}$$

Further, if k is replaced by $(k+1)$ in the figure, we find the identity

$$\beta_{k+1}\|d_k\| = \tan \theta_{k+1} \|g_{k+1}\|. \tag{2.2}$$

We may eliminate $\|d_k\|$ from equations (2.1) and (2.2) and substitute the definition (1.5) to deduce the inequality

$$\begin{aligned} \tan \theta_{k+1} &= \sec \theta_k \|g_{k+1}\| / \|g_k\| \\ &> \tan \theta_k \|g_{k+1}\| / \|g_k\|. \end{aligned} \tag{2.3}$$

Now, if θ_k is close to $\frac{1}{2}\pi$, the iteration may take a very small step, in which case the change $(g_{k+1} - g_k)$ is small also. Thus the ratio $\|g_{k+1}\|/\|g_k\|$ is close to one. It follows from inequality (2.3) that θ_{k+1} is close to $\frac{1}{2}\pi$, so slow progress may occur again on the next iteration. Numerical calculations, in particular Dr. Norgett's

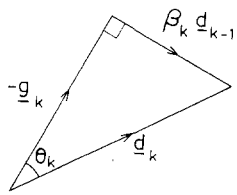

Fig. 1. The definition of θ_k.

example which was mentioned earlier, show that this inefficient behaviour can continue for several iterations when β_k is defined by equation (1.5). The following simple example provides a good demonstration of this effect.

Suppose that the early iterations of the algorithm have made θ_k positive, but that a region in the space of the variables has been reached where $F(\boldsymbol{x})$ is the quadratic function

$$F(\boldsymbol{x}) = x_1^2 + x_2^2. \tag{2.4}$$

In this case the line search along $\boldsymbol{d}_k$ makes the ratio $\|\boldsymbol{g}_{k+1}\|/\|\boldsymbol{g}_k\|$ equal to $\sin\theta_k$. Therefore the first line of expression (2.3) shows that θ_{k+1} is equal to θ_k. Thus the angle between the search direction and the steepest descent direction remains constant for all iterations, which is highly inefficient if θ_k is close to $\frac{1}{2}\pi$. Note that this inefficient behaviour is corrected by a steepest descent restart.

Alternatively, if expression (1.10) is used to define β_k, then the iterations of the conjugate gradient method have never seemed to be less efficient than those of the steepest descent method. We use equations (2.1) and (2.2) to show that the behaviour described in the last two paragraphs does not occur. Now the definition of β_k provides the bound

$$\beta_{k+1} \leq \|\boldsymbol{g}_{k+1}\| \cdot \|\boldsymbol{g}_{k+1} - \boldsymbol{g}_k\| / \|\boldsymbol{g}_k\|^2, \tag{2.5}$$

so the elimination of $\|\boldsymbol{d}_k\|$ from the two equations gives the inequality

$$\tan\theta_{k+1} \leq \sec\theta_k \|\boldsymbol{g}_{k+1} - \boldsymbol{g}_k\| / \|\boldsymbol{g}_k\|. \tag{2.6}$$

It follows that, if θ_k is close to $\frac{1}{2}\pi$ and if this causes the step from $\boldsymbol{x}_k$ to $\boldsymbol{x}_{k+1}$ to be so small that the change $(\boldsymbol{g}_{k+1} - \boldsymbol{g}_k)$ is much less than $\|\boldsymbol{g}_k\|$, then $\tan\theta_{k+1}$ is much less than $\sec\theta_k$. Thus the search direction $\boldsymbol{d}_{k+1}$ is turned towards the steepest descent direction. Inequality (2.6) is sufficiently powerful to prove the following convergence theorem which, in contrast to a similar theorem given by Polak [8], does not require $F(\boldsymbol{x})$ to satisfy any convexity conditions.

Theorem 1. *If the set* $\{\boldsymbol{x}: F(\boldsymbol{x}) \leq F(\boldsymbol{x}_1)\}$ *is bounded, if* $\boldsymbol{g}(\boldsymbol{x})$ *is continuous, and if the step-lengths* $\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|$ *tend to zero, then the conjugate gradient algorithm without restarts that uses equation* (1.10) *provides the limit*

$$\liminf_{k\to\infty} \|\boldsymbol{g}_k\| = 0. \tag{2.7}$$

Proof. If the theorem is false there exists a positive constant ϵ and an integer l such that the bound

$$\|\boldsymbol{g}_k\| \geq \epsilon \tag{2.8}$$

holds for all $k \geq l$. Since $\boldsymbol{g}(\boldsymbol{x})$ is continuous and since the step-lengths tend to zero, there exists an integer $m \geq l$ such that the inequality

$$\|\boldsymbol{g}_{k+1} - \boldsymbol{g}_k\| \leq \tfrac{1}{2}\epsilon, \quad k \geq m, \tag{2.9}$$

is satisfied. Therefore, because the relation

$$\sec \theta_k \leq 1 + \tan \theta_k \tag{2.10}$$

is true for all θ_k in $[0, \frac{1}{2}\pi)$, expression (2.6) implies the bound

$$\tan \theta_{k+1} \leq \tfrac{1}{2}(1 + \tan \theta_k), \quad k \geq m, \tag{2.11}$$

which is applied recursively to give the inequality

$$\begin{aligned} \tan \theta_{k+1} &\leq \tfrac{1}{2} + \tfrac{1}{4} + \cdots + (\tfrac{1}{2})^{k+1-m}(1 + \tan \theta_m) \\ &< 1 + \tan \theta_m, \quad k \geq m. \end{aligned} \tag{2.12}$$

Therefore the angle between $\boldsymbol{d}_k$ and the steepest descent direction $-\boldsymbol{g}_k$ is bounded away from $\frac{1}{2}\pi$. This condition implies that $\|\boldsymbol{g}_k\|$ tends to zero [10], which contradicts the hypothesis (2.8), so the theorem is true.

The main purpose of the theorem is to show the power of the bound (2.6) that follows from the definition (1.10). Note that we are not claiming that the theorem is false if instead β_k has the value (1.5) because this is still an open question. One further comment on the value (1.5) is instructive. If the Fletcher–Reeves algorithm finds a small value of $\|\boldsymbol{g}_k\|$ because $\boldsymbol{x}_k$ is close to a stationary point of $F(\boldsymbol{x})$, and if subsequent values of $\|\boldsymbol{g}_k\|$ become larger due to the search path moving away from the stationary point, then the subsequent values of θ_k must be larger than the one that occurred when $\|\boldsymbol{g}_k\|$ was small. This is an unfortunate property, and it is a consequence of the fact that expression (2.3) can be written in the form

$$\tan \theta_{k+1}/\|\boldsymbol{g}_{k+1}\| < \tan \theta_k/\|\boldsymbol{g}_k\|. \tag{2.13}$$

For the reasons given in this section the use of expression (1.10) is recommended instead of the value (1.5).

3. Some restart procedures

The following example shows a disadvantage of using the steepest descent direction

$$\boldsymbol{d}_k = -\boldsymbol{g}_k \tag{3.1}$$

as a restarting direction in the conjugate gradient method. Thus it motivates the work of the remainder of this section which is to study techniques that allow other restarting directions.

The Polak–Ribière conjugate gradient algorithm is applied to the helical valley function

$$f(x_1, x_2, x_3) = 100\{(x_3 - 10\theta)^2 + (r - 1)^2\} + x_3^2, \tag{3.2}$$

where θ and r are defined by the equations

$$2\pi\theta = \begin{cases} \arctan(x_2/x_1), & x_1 > 0, \\ \pi + \arctan(x_2/x_1), & x_1 < 0, \end{cases}$$
$$r = \sqrt{x_1^2 + x_2^2} \tag{3.3}$$

and where the components of $\boldsymbol{x}_1$ are $(-1, 0, 0)$. This algorithm is described in Section 1, except that now we include the restarting procedure that sets $\boldsymbol{d}_k$ to the direction (3.1) instead of the direction (1.4) for certain values of k. In the example these values of k occur every t iterations, t being an integer from the range $1 \leq t \leq 5$. The calculated values of $F(\boldsymbol{x}_{k+1})$ are given in Table 1, where a horizontal line is drawn directly above each value of $F(\boldsymbol{x}_{k+1})$ that is obtained by a search along the steepest descent direction (3.1).

Table 1
Steepest descent restarts every t iterations

k	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$
1	520.451	520.451	520.451	520.451	520.451
2	519.849	123.724	123.724	123.724	123.724
3	11.164	10.193	9.794	9.794	9.794
4	7.205	6.696	6.920	6.920	6.920
5	6.355	6.563	2.637	6.145	6.082
6	6.177	6.431	2.348	5.455	5.597
7	6.087	6.317	1.487	3.446	5.501
8	6.025	6.182	1.167	0.700	3.000
9	5.965	6.029	1.088	0.504	2.218
10	5.904	5.935	1.023	0.433	1.402

The purpose of the table is to show that nearly always the short term effect of restarting is poor. By this we mean that the search direction (1.4) usually gives a better value of $F(\boldsymbol{x}_{k+1})$ than the search direction (3.1). For instance the table shows that if the first restart is made on the third iteration then the calculated value of $F(\boldsymbol{x}_4)$ is 10.193, but if no restart occurs then $F(\boldsymbol{x}_4)$ has the lower value 9.794. This kind of behaviour is certain when $F(\boldsymbol{x})$ is a quadratic function, because the search direction (1.4) leads to the least value of $F(\boldsymbol{x})$ in the two-dimensional affine set that contains the directions $\boldsymbol{d}_{k-1}$ and $-\boldsymbol{g}_k$, but the direction (3.1) does not take account of $\boldsymbol{d}_{k-1}$. Therefore we agree with McGuire and Wolfe [7] that we would like to find a restarting method that does not

abandon the second derivative information that is found by the search along $\boldsymbol{d}_{k-1}$.

Suitable methods can be obtained by letting $\boldsymbol{d}_k$ be the vector (1.4) when a restart is made, and by extending the definition of $\boldsymbol{d}_k$ on the ordinary iterations. In particular Beale's [1] paper gives a method of this type which is described below. This method has been tried by McGuire and Wolfe [7] who found their numerical results disappointing. However their difficulties can be turned to advantage because a restart is needed when they occur. Thus conditions for restarting are obtained that are suited to the current function $F(\boldsymbol{x})$. This approach gives the algorithm described in Section 4, which seems to be very successful in practice.

Beale's technique comes from the following question. If $\boldsymbol{d}_t$ is an arbitrary downhill restarting direction, if $F(\boldsymbol{x})$ is a quadratic function, and if the search direction $\boldsymbol{d}_k$ $(k > t)$ is to be a linear combination of $\boldsymbol{d}_t$ and the calculated gradients $\boldsymbol{g}_{t+1}, \boldsymbol{g}_{t+2}, \ldots, \boldsymbol{g}_k$, what linear combination makes the sequence of search directions $\boldsymbol{d}_t, \boldsymbol{d}_{t+1}, \boldsymbol{d}_{t+2}, \ldots$ mutually conjugate? It is shown [1] that $\boldsymbol{d}_k$ has the form

$$\boldsymbol{d}_k = -\boldsymbol{g}_k + \beta_k \boldsymbol{d}_{k-1} + \gamma_k \boldsymbol{d}_t, \tag{3.4}$$

which can be regarded as expression (1.4) augmented by a multiple of the restarting direction $\boldsymbol{d}_t$. As in Section 1 the value of β_k is calculated to make $\boldsymbol{d}_k$ conjugate to $\boldsymbol{d}_{k-1}$, while the extra term $\gamma_k \boldsymbol{d}_t$ is present to provide conjugacy to $\boldsymbol{d}_t$ also. It is now appropriate to let β_k and γ_k have the values

$$\begin{aligned} \beta_k &= \boldsymbol{g}_k^{\mathrm{T}}[\boldsymbol{g}_k - \boldsymbol{g}_{k-1}]/\boldsymbol{d}_{k-1}^{\mathrm{T}}[\boldsymbol{g}_k - \boldsymbol{g}_{k-1}], \\ \gamma_k &= \boldsymbol{g}_k^{\mathrm{T}}[\boldsymbol{g}_{t+1} - \boldsymbol{g}_t]/\boldsymbol{d}_t^{\mathrm{T}}[\boldsymbol{g}_{t+1} - \boldsymbol{g}_t], \end{aligned} \tag{3.5}$$

except that γ_k is zero when $k = t+1$. No extra terms are needed to make $\boldsymbol{d}_k$ conjugate to the intermediate directions $\boldsymbol{d}_{t+1}, \boldsymbol{d}_{t+2}, \ldots, \boldsymbol{d}_{k-2}$ in the quadratic case so we do not introduce any extra terms when $F(\boldsymbol{x})$ is a general function, in order to retain the main advantage of the conjugate gradient algorithm which is that no matrices have to be stored.

However we noted in Section 1 that, when $F(\boldsymbol{x})$ is a general function, this method can cause the direction (3.4) to be uphill or orthogonal to the gradient $-\boldsymbol{g}_k$, even when all line searches are exact [7]. Therefore it is interesting to find techniques that provide quadratic termination, that allow $\boldsymbol{d}_k$ to be an arbitrary downhill restarting direction, and that satisfy the downhill condition

$$\boldsymbol{d}_k^{\mathrm{T}} \boldsymbol{g}_k < 0, \quad k > t, \tag{3.6}$$

in the general case. A suitable technique is described below but we will prefer Beale's method for the new algorithm.

This technique depends on the comment that, if W is a nonsingular matrix such that the equation

$$W\boldsymbol{d}_t = -(W^{\mathrm{T}})^{-1}\boldsymbol{g}_t \tag{3.7}$$

holds, then the change of variables

$$\boldsymbol{x} \leftarrow W\boldsymbol{x} \tag{3.8}$$

causes $\boldsymbol{d}_t$ to become the steepest descent direction in the transformed space.

Therefore for $k > t$ it is appropriate to apply the iteration of Section 1 in the transformed space. By straightforward algebra the resultant search directions can be expressed in terms of the original space. Thus we obtain the following method, where P is the matrix $(W^{\mathrm{T}}W)^{-1}$.

Let $\boldsymbol{d}_t$ be a downhill restart direction and let P be a constant positive definite matrix that satisfies the equation

$$\boldsymbol{d}_t = -P\boldsymbol{g}_t. \tag{3.9}$$

For $k > t$ let $\hat{\boldsymbol{g}}_k$ be the vector

$$\hat{\boldsymbol{g}}_k = P\boldsymbol{g}_k \tag{3.10}$$

and let $\boldsymbol{d}_k$ be the search direction

$$\boldsymbol{d}_k = -\hat{\boldsymbol{g}}_k + \beta_k \boldsymbol{d}_{k-1}, \tag{3.11}$$

where β_k has the value

$$\beta_k = \hat{\boldsymbol{g}}_k^{\mathrm{T}}[\boldsymbol{g}_k - \boldsymbol{g}_{k-1}]/\boldsymbol{d}_{k-1}^{\mathrm{T}}[\boldsymbol{g}_k - \boldsymbol{g}_{k-1}]. \tag{3.12}$$

For $k \geq t$ calculate $\boldsymbol{x}_{k+1}$ by a line search from $\boldsymbol{x}_k$ along $\boldsymbol{d}_k$. This method gives quadratic termination and it satisfies the condition (3.6) when $F(\boldsymbol{x})$ is a general function, because the line search along $\boldsymbol{d}_{k-1}$ and equations (3.10) and (3.11) imply the inequality

$$\boldsymbol{d}_k^{\mathrm{T}}\boldsymbol{g}_k = -\boldsymbol{g}_k^{\mathrm{T}}P\boldsymbol{g}_k < 0, \tag{3.13}$$

unless the algorithm terminates because $\boldsymbol{g}_k$ is zero.

In order to avoid storing matrices we require the form of P to be simple. There are many suitable choices that satisfy equation (3.9). A particularly interesting one can be found by letting the transformation matrix tend to singularity. It is the projection matrix

$$P = I - \frac{\boldsymbol{g}_t\boldsymbol{g}_t^{\mathrm{T}}}{\|\boldsymbol{g}_t\|^2}. \tag{3.14}$$

This choice gives quadratic termination, but it has a severe disadvantage in the general case, which we consider because it emphasizes a very nice property of Beale's method.

The disadvantage is that, if P is the matrix (3.14), if the algorithm that uses equations (3.10)–(3.12) is applied to a general function, and if no restarts are made after the tth iteration, then usually the calculated sequence of points $\boldsymbol{x}_k$ $(k = 1, 2, 3, \ldots)$ converges to a limit, $\boldsymbol{x}^*$ say, where the gradient $\boldsymbol{g}(\boldsymbol{x}^*)$ is a non-zero multiple of $\boldsymbol{g}_t$. Therefore $\boldsymbol{x}^*$ is not the required vector of variables that minimizes $F(\boldsymbol{x})$. The following example demonstrates this behaviour.

Let the objective function be the quadratic

$$F(\boldsymbol{x}) = x_1^2 + x_2^2 + x_3^2 \tag{3.15}$$

when $F(\boldsymbol{x}) \leq F(\boldsymbol{x}_l)$, where l is some integer that is greater than t, but let the form of $F(\boldsymbol{x})$ otherwise be such that $\boldsymbol{g}_t$ is a multiple of the first coordinate vector and

$\boldsymbol{d}_l$ and $\boldsymbol{x}_{l+1}$ are the vectors

$$\boldsymbol{d}_l = \begin{pmatrix} 1 \\ \mu_{l+1} \\ \mu_l \end{pmatrix}, \quad \boldsymbol{x}_{l+1} = \begin{pmatrix} \mu_l/\mu_{l+1} \\ 0 \\ -1/\mu_{l+1} \end{pmatrix}, \tag{3.16}$$

where $\mu_{l+1} \geq \mu_l > 0$. Then, since $\boldsymbol{d}_{l+1}$ is a linear combination of $\boldsymbol{d}_l$ and $\hat{\boldsymbol{g}}_{l+1}$ that is conjugate to $\boldsymbol{d}_l$, we can introduce a scaling factor so that it is the direction

$$\boldsymbol{d}_{l+1} = \begin{pmatrix} 1 \\ \mu_{l+1} \\ -\mu_{l+2} \end{pmatrix}, \tag{3.17}$$

where μ_{l+2} is defined by the recurrence relation

$$\mu_{k+2} = (1 + \mu_{k+1}^2)/\mu_k, \quad k \geq l. \tag{3.18}$$

It follows that $\boldsymbol{x}_{l+2}$ is the vector

$$\boldsymbol{x}_{l+2} = \begin{pmatrix} \mu_{l+1}/\mu_{l+2} \\ -1/\mu_{l+2} \\ 0 \end{pmatrix}. \tag{3.19}$$

By continuing this process we find that the first component of $\boldsymbol{x}_k$ is μ_{k-1}/μ_k $(k > l)$, where the sequence μ_k $(k = l, l+1, l+2, \ldots)$ is defined by expression (3.18). We also find that the other two components of $\boldsymbol{x}_k$ are zero and $\pm 1/\mu_k$. Thus it is straightforward to prove that $\boldsymbol{x}^*$ is a positive multiple of the first coordinate vector. For example, if $\mu_l = \mu_{l+1} = 1$, it happens that μ_k is the integer

$$\mu_k = \frac{2}{5+\sqrt{5}}(\tfrac{3}{2}+\tfrac{1}{2}\sqrt{5})^{k-l} + \frac{2}{5-\sqrt{5}}(\tfrac{3}{2}-\tfrac{1}{2}\sqrt{5})^{k-l}, \quad k \geq l, \tag{3.20}$$

so μ_{k+1}/μ_k tends to the limit $\frac{1}{2}(3+\sqrt{5})$. Thus the sequence $\boldsymbol{x}_k$ $(k = 1, 2, 3, \ldots)$ converges to the point $(0.381966, 0, 0)$, which does not provide the least value of the objective function (3.15).

McGuire and Wolfe [7] point out that Beale's method has a similar property when $F(\boldsymbol{x})$ is a general function. This fact is easy to prove because equations (3.4) and (3.5) show that $\boldsymbol{d}_k$ is orthogonal to the change in gradients $(\boldsymbol{g}_{t+1} - \boldsymbol{g}_t)$ for all $k \geq t+1$. Thus, if Beale's method is not restarted after the tth iteration, it is usual for the limiting gradient $\boldsymbol{g}(\boldsymbol{x}^*)$ to be a non-zero multiple of $(\boldsymbol{g}_{t+1} - \boldsymbol{g}_t)$.

However there is a very important difference between the two restart procedures, which is that the need to restart again can be identified automatically when Beale's method is applied, but there is no similar technique when the projection matrix (3.14) is used. This remark depends on the fact that, when $F(\boldsymbol{x})$ is a quadratic function, then Beale's method makes the gradients $\boldsymbol{g}_k$ $(k = t+1, t+2, \ldots)$ mutually orthogonal, but the projection method makes only the projected gradients $P\boldsymbol{g}_k$ $(k = t+1, t+2, \ldots)$ mutually orthogonal. Therefore

in Beale's method a suitable time to restart is when the expression

$$\boldsymbol{g}_k^{\mathrm{T}}\boldsymbol{g}_{k+1}/\|\boldsymbol{g}_{k+1}\|^2 \tag{3.21}$$

is much different from zero. This criterion prevents $\boldsymbol{g}_k$ from tending to a non-zero limit. However, when the projection method is applied, then expression (3.21) can be much different from zero even when termination will occur because $F(\boldsymbol{x})$ is exactly quadratic. Therefore instead it may seem appropriate to test the quantity

$$(P\boldsymbol{g}_k)^{\mathrm{T}}P\boldsymbol{g}_{k+1}/\|P\boldsymbol{g}_{k+1}\|^2, \tag{3.22}$$

but this expression is zero for all $k > l$ in the example (3.15)–(3.20), even though $\boldsymbol{g}_k$ tends to a non-zero limit. It would be interesting to study other choices of P that require only a few vectors of storage.

We claim that restarting Beale's method when expression (3.21) is much different from zero is more than a device to prevent convergence to the wrong point – it is a positive advantage, because the test for restarting is suited to the particular form of $F(\boldsymbol{x})$. We have in mind that sometimes $F(\boldsymbol{x})$ contains so much symmetry that it is appropriate to restart more often than every n iterations. In this case any underlying quadratic characteristics of $F(\boldsymbol{x})$ are exploited in far fewer than n iterations, and when these are exhausted then $\boldsymbol{g}_k$ is likely to contain a relatively large component of $(\boldsymbol{g}_{t+1} - \boldsymbol{g}_t)$, in which case another restart is usually made automatically. Also we may wish to restart automatically after fewer than n iterations in a case discussed by Luenberger [6], where the second derivative matrix of $F(\boldsymbol{x})$ has a few large eigenvalues and the remaining ones are clustered. Alternatively, when $F(\boldsymbol{x})$ is a general quadratic function and exact arithmetic is used, then expression (3.21) is zero for all $k > t$, so the algorithm continues until termination is achieved. Finding an automatic method that can treat all these situations appropriately was a main goal when the work described in this paper was started. Therefore this feature of Beale's method is very satisfactory.

4. The new algorithm

The work described already provides a new algorithm that is defined in this section. Some numerical examples are given also to show that the new method compares favourably with the versions of the Polak–Ribière and Fletcher–Reeves conjugate gradient algorithms that restart by using the steepest descent direction (3.1) every n iterations.

One parameter that is needed by the new algorithm is a quantity to compare with expression (3.21) in order to decide whether enough orthogonality between $\boldsymbol{g}_{k-1}$ and $\boldsymbol{g}_k$ has been lost to warrant another restart. Since expression (1.10) occurs naturally the quantities $\boldsymbol{g}_{k-1}^{\mathrm{T}}\boldsymbol{g}_k$ and $\|\boldsymbol{g}_k\|^2$ are comparable. Therefore it is appropriate to set this parameter to a constant. Numerical experiments suggest that all values in the range $(0.1, 0.9)$ are satisfactory and for definiteness the value 0.2 is chosen. The same numerical constant occurs in a test to decide whether the direction $\boldsymbol{d}_k$ is sufficiently downhill. The new algorithm is as follows.

We are given $\boldsymbol{x}_1$, we define $\boldsymbol{d}_1$ to be the steepest descent direction $-\boldsymbol{g}_1$, we let $k = t = 1$ and we begin the iterations. If $k \geq 2$ we test the inequality

$$|\boldsymbol{g}_{k-1}^{\mathrm{T}} \boldsymbol{g}_k| \geq 0.2\|\boldsymbol{g}_k\|^2. \tag{4.1}$$

If it holds we set $t = k - 1$. We also restart by setting $t = k - 1$ if $(k - t) \geq n$ because in this case n mutually conjugate search directions will have been used when $F(\boldsymbol{x})$ is a quadratic function. For $k \geq 2$ the search direction is defined by equations (3.4) and (3.5), except that γ_k is zero when $k = t + 1$. If $k > t + 1$ we must check that the search direction $\boldsymbol{d}_k$ is sufficiently downhill. Therefore we try the inequalities

$$-1.2\|\boldsymbol{g}_k\|^2 \leq \boldsymbol{d}_k^{\mathrm{T}} \boldsymbol{g}_k \leq -0.8\|\boldsymbol{g}_k\|^2. \tag{4.2}$$

If they are not satisfied we let $t = k - 1$ and we redefine $\boldsymbol{d}_k$ by letting $\gamma_k = 0$ in equation (3.4). Thus we obtain a suitable search direction for all values of k. As usual $\boldsymbol{x}_{k+1}$ is the vector (1.6) where λ_k is calculated to minimize the function (1.7). The iterations finish if $\|\boldsymbol{g}_{k+1}\|$ or $F(\boldsymbol{x}_{k+1})$ is sufficiently small. Otherwise k is increased by one and another iteration is begun.

It should be noted that the storage required by the new algorithm exceeds that of the Polak–Ribière and Fletcher–Reeves methods, because after a restart we need the vectors $\boldsymbol{d}_t$ and $(\boldsymbol{g}_{t+1} - \boldsymbol{g}_t)$. Thus at least six vectors of storage are needed altogether compared with four vectors in the Polak–Ribière method and only three vectors in the Fletcher–Reeves method. Moreover at all stages in the line search one should retain the value of $\boldsymbol{x}$ that gives the least calculated value of $F(\boldsymbol{x})$ and also the corresponding gradient $\boldsymbol{g}(\boldsymbol{x})$, in case the line search cannot be completed satisfactorily because of computer rounding errors. Thus each algorithm may require an extra two vectors. All these demands do not prevent problems of twenty thousand variables from fitting into the core store of many computers, so only rarely will the usefulness of the new algorithm be impaired by its storage requirements.

Since the main purpose of this paper is to show the merits of Beale's restart procedure, we prefer not to dwell on the details of the line search techniques. However these details are very important to the total number of function and gradient evaluations that are needed to complete a minimization calculation. A line search method that is well suited to the present algorithm is given in the listing of subroutine VA14A, which can be obtained from the Harwell subroutine library. One feature of the line search that is special to conjugate gradient methods is that not only do we require a decrease in the directional derivative, for instance by satisfying the inequality

$$|\boldsymbol{d}_k^{\mathrm{T}} \boldsymbol{g}_{k+1}| < 0.1|\boldsymbol{d}_k^{\mathrm{T}} \boldsymbol{g}_k|, \tag{4.3}$$

but also we require the angle between $\boldsymbol{g}_{k+1}$ and $\boldsymbol{d}_k$ to be close to ninety degrees. This extra precaution is necessary because the direction of $\boldsymbol{d}_{k+1}$ is sensitive to the direction of $\boldsymbol{g}_{k+1}$ and the conjugate gradient algorithms are less good at recovering from errors than are the variable metric methods for minimization.

To compare the present algorithm numerically with the Polak–Ribière and Fletcher–Reeves methods in a way that almost avoids any dependence on the

line search, we made every line search correct to 10^{-5} relative accuracy. Note that this is not an optimal strategy in terms of the total amount of computation, but it does allow a comparison between the number of iterations required by each algorithm. Results for two test functions are reported, namely the helical valley function (3.2) with $x_1 = (-1, 0, 0)$ and the trigonometric function of the form

$$F(x) = \sum_{i=1}^{n} \left\{ \sum_{j=1}^{n} A_{ij} \sin x_j + B_{ij} \cos x_j - E_i \right\}^2 \tag{4.4}$$

which was introduced by Fletcher and Powell [4]. The method of generating the parameters and the vector x_1 for the function (4.4) is described in reference [4].

The number of iterations required to reduce $F(x)$ below 10^{-8} in the case of the helical valley function is given in the first row of Table 2. The other rows of the table show the number of iterations required to reduce the function (4.4) below 10^{-5} for various even values of n, except that if more than two hundred iterations are needed the value of $F(x_{201})$ is given instead, which occurs four times in the table. These figures were calculated in double-length arithmetic on an IBM 370 computer so any effects from rounding errors are very slight. However the numbers given by Fletcher [3] were calculated in single-length arithmetic, which prevented him from obtaining the same accuracy in $F(x)$. Therefore his figures differ greatly from those of Table 2.

The table shows very clearly that the restart procedure we have studied is a valuable technique. Note that the given numerical examples do not include any symmetry properties which, for reasons given at the end of Section 3, can be even more advantageous to the new algorithm.

Table 2
A comparison of three algorithms

n	Polak–Ribière	Fletcher–Reeves	New algorithm
3	30	33	24
2	3	3	4
4	12	12	9
6	24	24	13
8	56	64	35
10	70	121	40
20	1.3×10^{-3}	1.2×10^{-3}	83
30	7.3×10^{-4}	4.7×10^{-4}	122

Acknowledgment

I wish to thank Dr. M.J. Norgett for the interesting example that is reported in Section 2. Also I am grateful to Dr. J.K. Reid for studying a draft of the paper and suggesting several improvements. The interest shown in this work by the staff and visitors at the Argonne National Laboratory during their 1975 workshop on Optimization was very encouraging and helpful.

References

[1] E.M.L. Beale, "A derivation of conjugate gradients", in: F.A. Lootsma, ed., *Numerical methods for nonlinear optimization* (Academic Press, London, 1972) pp. 39–43.
[2] H.P. Crowder and P. Wolfe, "Linear convergence of the conjugate gradient method", *IBM Journal of Research and Development* 16 (1972) 431–433.
[3] R. Fletcher, "A Fortran subroutine for minimization by the method of conjugate gradients", Report R-7073, A.E.R.E., Harwell, 1972.
[4] R. Fletcher and M.J.D. Powell, "A rapidly convergent descent method for minimization", *Computer Journal* 6 (1963) 163–168.
[5] R. Fletcher and C.M. Reeves, "Function minimization by conjugate gradients", *Computer Journal* 7 (1964) 149–154.
[6] D.G. Luenberger, *Introduction to linear and nonlinear programming* (Addison-Wesley, New York, 1973).
[7] M.F. McGuire and P. Wolfe, "Evaluating a restart procedure for conjugate gradients", Report RC-4382, IBM Research Center, Yorktown Heights, 1973.
[8] E. Polak, *Computational methods in optimization: a unified approach* (Academic Press, London, 1971).
[9] M.J.D. Powell, "Some convergence properties of the conjugate gradient method", *Mathematical Programming* 11 (1976) 42–49.
[10] G. Zoutendijk, "Nonlinear programming, computational methods", in: J. Adabie, ed., *Integer and nonlinear programming* (North-Holland, Amsterdam, 1970) pp. 37–86.