Numerical Optimisation Centre

TECHNICAL REPORT NO. 130 FEBRUARY 1983

A NEW THREE TERM CONJUGATE GRADIENT METHOD

by

L C W Dixon, P G Ducksbury and P Singh

The Hatfield Polytechnic

PAUL DUCKSBURY

THE HATFIELD POLYTECHNIC

NUMERICAL OPTIMISATION CENTRE

A NEW THREE TERM CONJUGATE GRADIENT METHOD

by

L C W Dixon, P G Ducksbury and P Singh

TECHNICAL REPORT NO. 130 FEBRUARY 1983

Abstract

In this paper we describe an implementation and give performance results for a conjugate gradient algorithm for unconstrained optimisation. The algorithm is based upon the Nazareth three term formula and incorporates Allwright preconditioning matrices and restart tests. The performance results for this combination compare favourably with existing codes.

and using equation (2)

$$f(x) - f(x^*) = \tfrac{1}{2} \sum_i (\beta_i^2 - \beta_i^{*2}) t^{(i)T} G t^{(i)} \qquad (5)$$

THEOREM 1: QUADRATIC TERMINATION

It follows that if we minimise f(x) by searching along each conjugate direction in turn, then each search is independent of the other and must set $\beta_i = \beta_i^*$. The minimum of a quadratic function is therefore obtained after searching along n mutually conjugate directions.

This is, of course, the basis of the conjugate gradient approach but if this were its only property the method would be hopelessly inefficient on most large problems.

In conjugate gradient algorithms the sequence of directions $t^{(k)}$ and a sequence of points $x^{(k)}$ are generated iteratively:-

$$x^{(k+1)} = x^{(k)} + \alpha t^{(k)} \qquad (6)$$

where the step α is chosen so that if the function is the quadratic (1) then $x^{(k+1)}$ would be the minimum along the line (6).

This implies that $x^{(k+1)}$ is then the minimum of the quadratic function in the space spanned by $[t^{(1)}, \ldots, t^{(k)}]$ and if the initial direction $t^{(1)} = g^{(1)}$ and the classic iterative method for generating $t^{(k)}$ is used this space is the same as that spanned by $[g^{(1)}, Gg^{(1)}, \ldots, G^{k-1} g^{(1)}]$.

From this two well known theorems have been derived which account for the practical usefulness of the method on large systems.

THEOREM 2

The number of iterations to obtain the minimum of (1) is bounded above by the number of independent vectors in the set $(g^{(1)}, Gg^{(1)}, \ldots, G^{n-1}g^{(1)})$.

THEOREM 3

The number of iterations to obtain the minimum of (1) is bounded above by the number of distinct eigenvalues of G.

2.2 On Nonquadratic Functions

When the function to be optimised is nonquadratic the above framework can still be used to generate an efficient algorithm. Many codes have been written for this purpose beginning with that described in Fletcher and Reeves [2]. In proposing a new conjugate gradient algorithm we wished to ensure its convergence pattern is based on Theorem 2 and 3 and not Theorem 1 above.

The structure of an early algorithm based on Theorem 1 consisted of seven basic steps:

ALGORITHM 1

Step 1 Select $x^{(1)}$, ε_o, jmax. Set k=1, j=1

Calculate $f^{(1)}$ and $g^{(1)}$

Step 2 If $\| g^{(k)} \| < \varepsilon_o$, STOP

If j $\geq$ jmax, STOP

If $\| x^{(k)} - x^{(k-n)} \| \leq \varepsilon_o$, STOP

Step 3 If k = 1 set $p^{(k)} = - g^{(k)}$

otherwise set $p^{(k)} = - g^{(k)} + bp^{(k-1)}$

Step 4 Perform an almost perfect line search along $p^{(k)}$

Step 5 Calculate $g^{(k+1)}$, b and set j = j+1, k = k+1

Step 6 If k = n set k = 1, to start a new cycle

Step 7 GOTO STEP 2.

In the above algorithm Step 3 generates the set of conjugate directions on (1) provided the line search undertaken in Step 4 is perfect. In as far as conjugacy is only defined for quadratic functions the line search in Step 4 ought to contain a parabolic interpolation to guarantee that it is exact on quadratic functions.

To obtain a downhill direction we need $p^{(k)T} g^{(k)} < 0$ where

$$p^{(k)T} g^{(k)} = - g^{(k)T} g^{(k)} + b p^{(k-1)T} g^{(k)}.$$

The last term is, of course, zero for a perfect line search but otherwise this condition implies that the search must be sufficiently accurate for

$$p^{(k-1)T} g^{(k)} < \frac{g^{(k)T} g^{(k)}}{b} \tag{7}$$

which imposes one limit on the accuracy needed in the line search.

The form of b obtained immediately in Step 3 by insisting $p^{(k)}$ is conjugate to $p^{(k-1)}$, namely

$$b = \frac{g^{(k)T} y^{(k-1)}}{p^{(k-1)T} y^{(k-1)}} \tag{8}$$

can be simplified using the conjugacy and orthogonality properties and alternative formula are often used.

By the definition of $y^{(k-1)}$

$$b = \frac{g^{(k)T} (g^{(k)} - g^{(k-1)})}{p^{(k-1)T} (g^{(k)} - g^{(k-1)})} . \tag{9}$$

Using the perfect line search condition twice then gives the Polak-Ribierre [3] formula

$$b = \frac{g^{(k)T} (g^{(k)} - g^{(k-1)})}{g^{(k-1)T} g^{(k-1)}} \tag{10}$$

and applying the orthogonality property then gives the Fletcher Reeves formula

$$b = \frac{g^{(k)T} g^{(k)}}{g^{(k-1)T} g^{(k-1)}} . \tag{11}$$

Although both formulae are based on the assumption of a perfect line search they are frequently used in conjunction with imperfect searches. There is little conclusive numerical evidence on which of the alternative formulae for b is preferable but in [4] Powell argues in favour of the Polak-Ribiere form (10). He shows that if the conjugacy property

$g^{(k)T} g^{(k-1)} = 0$ has already been lost, then the Polak-Ribierre form bounds the angle between $p^{(k)}$ and $g^{(k)}$ and can produce a much more downhill direction.

Finite termination of the above type of algorithm follows from Wolfe's theorem [5, 6] provided the line search in Step 4 is such that Conditions II and III are satisfied. Condition I is automatically satisfied on the regular subset of directions with k = 1, as the search is then along the steepest descent direction.

Cohen [7] first proved that the above class of algorithms are superlinearly convergent and Dixon [8] contains one proof that there is a region round the solution in which it has n step second order convergence.

Most recent research work on the conjugate gradient algorithm has concentrated in four areas:

(1) the avoidance of the implied perfect line search at Step 4,

(2) the avoidance of the steepest descent step at the beginning of each cycle at Step 3,

(3) the introduction of an improved resetting test at Step 6 so that the cycle need not contain the full n iterations,

(4) preconditioning the objective function to obtain a better eigenstructure and hence improved performance.

3. Inexact Line Searches

The proof of finite termination of the conjugate gradient algorithm only requires that the line search at Step 4 satisfies Condition II and III of Wolfe's Theorem when k = 1. If, however, the direction of search is not reset regularly to the steepest descent direction then, to retain finite termination, it is necessary to test the direction of search against Wolfe's Condition I regularly and in the implementations discussed this is done at every iteration, and also Conditions II and III are satisfied at each iteration.

The n-step second order convergence property can be retained if a parabolic or cubic interpolation is included in the line search and the accuracy steadily improved so that v the angle between $g^{(k+1)}$ and $p^{(k)}$ satisfies

$$| \cos v | < R \, || \, g^{(1)} \, || \tag{12}$$

where $g^{(1)}$ is the gradient at the start of the cycle.

As this is till a stringent condition to achieve as $|| \, g^{(1)} \, || \to 0$, methods were sought for retaining n-step second order convergence for more relaxed line searches. In the earliest of these, the gradient prediction method Dixon [9], this rate was achieved by the introduction of two additional vectors, thus not greatly increasing the storage requirement. The basis of the method was to generate the same set of conjugate directions that would have resulted from the use of perfect line searches and to store a correction vector.

Suppose that a step $d^{(k)}$ is taken along the direction $p^{(k)}$ to a new point $x^{(k+1)}$ where the new gradient is $g^{(k+1)}$ but that, if a perfect step had been taken, the step would have been $d^{(k)}$. Then, from the properties of a quadratic function it can be shown that the following ratios are equal.

$$\theta_k = \frac{|| \, \tilde{d}^{(k)} \, ||}{|| \, d^{(k)} \, ||} = \frac{p^{(k)T} g^{(k)}}{p^{(k)T}(g^{(k)} - g^{(k+1)})} = -\frac{p^{(k)T} g^{(k)}}{p^{(k)T} y^{(k)}} \, . \tag{13}$$

This enables us to predict both the point that would have been reached with a perfect line search and the gradient at that point. Also due to the conjugacy of the directions generated these corrections are independent and we can define $x^{(k+1)*}$ and $g^{(k+1)*}$ the point and gradient that would have been reached after k perfect line searches

as $$g^{(k+1)*} = g^{(k+1)} - w^{(k+1)}; \quad x^{(k+1)*} = x^{(k+1)} - z^{(k+1)}$$

where $$w^{(k+1)} = w^{(k)} - (\theta-1)(y^{(k)}) \qquad w^{(1)} = 0 \tag{14}$$

$$z^{(k+1)} = z^{(k)} - (\theta-1)(d^{(k)}) \qquad z^{(1)} = 0$$

and $\theta - 1 = \dfrac{p^{(k)T} g^{(k+1)}}{p^{(k)T} y^{(k)}}$.

The prediction g* is then used in the conjugate gradient algorithm to calculate p with $y^{(k-1)}$ in the formula for b being replaced by $g^{(k)*} - g^{(k-1)*}$. On a quadratic function if $g^{(k)*} = 0$, this indicates that with perfect line searches $x^{(k)*}$ would be the solution and that a step z is necessary to obtain the solution. On a nonquadratic function it seems logical to include a line search along z if such a situation arises.

An alternative method for generating conjugate directions without performing accurate line searches was given by Nazareth [10].

Namely:

$$t^{(k+1)} = - y^{(k)} + \frac{y^{(k)T} y^{(k)}}{t^{(k)T} y^{(k)}} t^{(k)} + \frac{y^{(k)T} y^{(k-1)}}{t^{(k-1)T} y^{(k-1)}} t^{(k-1)} \qquad (15)$$

and in this form the new direction $t^{(k+1)}$ does not depend (apart from a scalar muliplier) on the length of the step along $t^{(k)}$. The same set of conjugate directions are therefore generated for arbitrary step sizes and the correction step z can be calculated as before. Again two additional vector stores are needed $t^{(k-1)}$ and z and again the value of $g^{(k)*}$ can be introduced to indicate when the step z would predict the solution.

4. Restarting Procedures

In the early algorithms the restarting strategy was usually to restart whenever k = n or n+1. When n is very large and the number of clusters of similar eigenvalues are very small this can be very inefficient. It is therefore often felt desirable to reset more regularly. If the gradient prediction algorithm is being used then it becomes natural to reset when $|| g^{(k)*} || \leq \varepsilon_o || g^{(1)} ||$ as this implies that enough

iterations have been undertaken to approximately minimise a quadratic function closely related to nonlinear objective function.

Powell [4] has suggested restarting whenever

$$| g^{(k)T} g^{(k-1)} | \geq 0.2 \, || g^{(k)} ||^2 . \qquad (16)$$

The left hand side would be zero if the conjugate gradient algorithm were working with perfect line searches on a quadratic function. Its size is therefore an indicator of the nonconjugacy of the search directions and so is an indicator that the cycle should be terminated.

Other measures could be similarly used, for instance, $t^{(k-1)T} y^{(k)}$ should be zero for conjugate directions and is available to the user of Nazareth's algorithm without additional store.

It also seems desirable to restart if the direction is not effectively downhill, Powell suggests restarting if

$$- 1.2 \, || g^{(k)} ||^2 \leq d^{(k)T} g^{(k)} \leq - 0.8 \, || g^{(k)} ||^2 \qquad (17)$$

is not satisfied, while in the codes reported in the tables the cycle was also restarted whenever

$$| g^{(k)T} d^{(k)} | \leq \varepsilon_1 \, || g^{(k)} || \, || d^{(k)} || \qquad \varepsilon_1 = 0.001 \qquad (18)$$

i.e. whenever Wolfe's Condition I was not satisfied.

5. Arbitrary Starting Directions

When it is decided to restart an algorithm it is questionable whether it is always advisable to restart by setting $p^{(1)} = - g^{(1)}$.

Beale [11] analysed the modifications that arose in the generation of conjugate directions if this assumption was not made and showed that conjugate directions were still obtained if

$$t^{(k+1)} = - g^{(k+1)} + b_1 t^{(1)} + b_k t^{(k)} \qquad (19)$$

where

$$b_1 = \frac{g^{(k+1)T} y^{(1)}}{t^{(1)T} y^{(1)}} . \qquad (20)$$

This approach allows a set of conjugate directions to be generated starting from any initial direction $t^{(1)}$, however, the fact that $t^{(1)}$ remains part of the formula for $t^{(k+1)}$ throughout the cycle may be undesirable. Although the storing of $t^{(1)}$ would enable the conjugacy check

$$t^{(1)T} g^{(k+1)} \approx 0 \tag{21}$$

to be made and used as an effective restarting strategy.

An alternative way of allowing an arbitrary starting direction was suggested by Allwright [12] who introduced a change of variables

$$x = Lz \tag{22}$$

where $H = LL^T$.

Then a quadratic function

$$F(x-x^*) = \tfrac{1}{2} (x-x^*)^T G(x-x^*)$$

is equivalent to

$$F(z-z^*) = \tfrac{1}{2} (z-z^*)^T L^T GL(z-z^*)$$

and

$$\nabla F_z = L^T GL(z-z^*)$$

$$= L^T G(x-x^*) = L^T g \tag{23}$$

which when transformed back into x space becomes simply Hg.

If the conjugate direction method is applied to make the directions conjugate in the z plane then the directions in the x plane are simply

$$p^{(1)} = t^{(1)} = - Hg^{(1)} \tag{24}$$

$$p^{(k+1)} = t^{(k+1)} = - Hg^{(k+1)} + bt^{(k)} \tag{25}$$

where

$$b = \frac{g^{(k)T} Hy^{(k-1)}}{p^{(k-1)T} Hy^{(k-1)}} \cdot \tag{26}$$

There are many alternative ways of selecting H so that $p^{(1)} = - Hg^{(1)}$ has any preselected value, p_h, one simple formula is to set

$$H = I - \frac{g^{(1)} g^{(1)T}}{g^{(1)T} g^{(1)}} - \frac{p_h p_h^T}{p_h^T g^{(1)}} \tag{27}$$

but this almost certainly not the best choice.

Indeed, if the Nazareth three term formula (15) is used to generate the set of directions, then the directions generated are conjugate for an arbitrary initial direction $t^{(1)}$, Nocedal [13]. However, if $t^{(1)}$ is not the gradient direction $t^{(k)}$ can become zero away from the minimum. This implies that provided the algorithm is restarted in the unlikely event that

$$|| \ t^{(k+1)} \ || \ < \varepsilon \ || \ g^{(1)} \ || \tag{28}$$

we can use the Allwright matrix to precondition the problem to improve the eigenstructure of $L^T GL$ and thus improve convergence. This was not however included in the algorithm when the numerical results were obtained, the matrix given by (27) being used.

6. The New conjugate Gradient Algorithm

The new conjugate gradient algorithm proposed combines the 3-term conjugate direction formula proposed by Nazareth, with a restarting procedure and includes a preconditioning matrix. The line search chosen is inexact but incorporates a parabolic interpolation and terminates at a point that satisfies Wolfe's Conditions II and III. If any direction is generated that does not satisfy Wolfe's Condition I the method is reset to ensure finite termination to a region around a solution defined by $g^T g \leq \varepsilon$.

To be precise:-

The proposed method generates the new search direction by the following formula,

$$t^k = - Hg^k + b_k t^{k-1} + b_{k-1} t^{k-2} \tag{29}$$

where

$$b_{k-1} = \frac{y^{k^T} H \, y^{k-1}}{t^{k-1} \, H \, y^{k-1}} , \qquad k > 1 \tag{30}$$

$$b_k = \frac{y^{k^T} H \, y^k}{t^{k^T} H \, y^k} \tag{31}$$

and H is a positive definite matrix. Tests are reported for two alternative values of H:-

(i) $H = I$

$$\text{(ii)} \quad H = I - \frac{g_h g_h^T}{g_h^T g_h} - \frac{p_h p_h^T}{p_h^T g_h} \tag{32}$$

where $g_h = g^{(1)}$ and p_h is the step direction before resetting k.

The algorithm is restarted whenever either

(1) $k \geq N + 1$ or

(2) $(p_h^T H y^k)^2 > 0.2(p_h^T H p_h)(y^{k^T} H y^k)$ - Test RST - (33)

(3) Wolfe's Condition I does not hold,

(4) $\| g^* \| \leq \varepsilon_o \| g^k \|$ (34)

(5) $\| t^{(k)} \| \leq \varepsilon_o \| g^k \|$ (35)

The second test (33) checks whether the new direction is reasonably conjugate to the initial steepest descent direction in the frame being used.

The matrix H is updated at the restart of a cycle if

$$g_h^T p_h < - C_1 \sqrt{(p_h^T p_h)(g_h^T g_h)} \qquad C_1 = 0.001 \tag{36}$$

i.e. if the resulting matrix would be positive definite.

The structure of the algorithm based on the above philosophy consists of the following steps and at any stage requires 13-N-vector stores for Z, g^k, g^{k-1}, t^k, t^{k-1}, t^{k-2}, y^k, y^{k-1}, C^k, g_h, p_h, w^k and a work space. Let us assume the following notations $v_1 = v^{k+1}$, $v_o = v^k$, $v_{-1} = v^{k-1}$, etc. where v is any given vector.

ALGORITHM 2: OPCG

Step 1 Select x, $j = 0$, $ITER = 0$

Step 2 Evaluate g_{-1} $\quad g_h = + g_{-1}$, $\quad p_h = - g_{-1}$

Set $t^o = - g_{-1}$, $\quad H = I$ and $\quad a = 1/\sqrt{(g^T g)}$

Step 3 Perform an approximate line search, $d_o = at_o$

$x_1 = x_o + d_o$, iter = iter + 1, j = j + 1

$g_{-1} = g_o$

Evaluate $g_o = g(x_1)$

for $j > 2$ if the line search is along the correction vector z_o

then do one of the following:

(i) If $g_h^T p_h \geq - C\sqrt{(p_h^T p_h)(g_h^T g_h)}$ then go to Step 13

(ii) Update the matrix $H = I - \frac{g_h g_h^T}{g_h^T g_h} - \frac{p_h p_h^T}{p_h^T g_h}$

or use matrix H = I and continue.

Step 4 Calculate g_o

If $\| g_o \| \leq \varepsilon_o$ stop

if iter $\geq$ iter max, stop.

Step 5 Calculate

$$y_o = g_o - g_{-1},$$

$$t_o = - Hy_o + \beta t_{-2} + \gamma t_{-1}$$

$$\left.\begin{aligned} z_o &= z_{-1} + (\theta-1)t_{-1} \\ w_o &= w_{-1} + (\theta-1)y_{-1} \end{aligned}\right] \quad z_{-1} = w_{-1} = 0 \text{ if } j = 1$$

$$g_o^* = g_o - w_o$$

where $\theta - 1 = - \frac{g_o^T d_{-1}}{y_o^T t_{-1}}$ $\quad j \geq 1,$

$$\beta = \begin{cases} \dfrac{y_o^T H y_{-1}}{y_{-1}^T H t_{-2}} & j \geq 2 \\ 0 & j = 0,1 \end{cases}$$

$$\gamma = \frac{y_o^T H y_o}{y_o^T H t_{-1}}$$

Step 6 If $C_1 \| t_o \| . \| g_o \| \geq -t_o^T g_o$ then go to Step 8

For $j > 1$ if $(p_h^T H y_o)^2 > 0.2(p_h^T H p_h)(y_o^T H y_o)$ then go to Step 8

If $\| g_o^* \| \leq C_1 \| g_o \|$ or $\| t_o \| \leq C_1 \| g_o \|$ then go to Step 8

The first of these two conditions is Wolfe's Condition I while the others are tests (33), (34) and (35) mentioned above.

Step 7 If $j \geq n + 1$ go to Step 8

If $j \leq n$ go to Step 3.

Step 8 Test z_o for Wolfe's Condition I. If satisfied go to Step 11.

Step 9 Set $g_h = g_o$, $p_h = z_o$. This effectively updates the matrix H.

Step 10 If $g_h^T p_h \geq - C_1 \sqrt{(p_h^T p_h)(g_h^T g_h)}$ then go to Step 13 (i.e. H is unsatisfactory), otherwise to go Step 5 with $j = 1$.

Step 11 Set $t_o = z_o$, $g_h = g_o$, $p_h = z_o$, $a = 1.0$, $j = 0$.

Step 12 Go to Step 3.

Step 13 Set $g_h = g_o$, $p_h = t_{-1}$, $j = 0$. This effectively updates the matrix H.

Step 14 If $g_h^T p_h \geq - C_1 \sqrt{(p_h^T p_h)(g_h^T g_h)}$ then go to Step 15. This tests if H is acceptable.

Step 15 Set $t_o = - g_o$, $j = 0$, $g_h = - g_o$, $p_h = -g_o$. This effectively sets H = I. Go to Step 3.

The line search at Step 3 consists of first performing a parabolic interpolation using the values of the function and its derivative at the starting point and the function value at an offset point. If the predicted function value does not satisfy Wolfe's Conditions II and III, then the Armijo [14] procedureis adopted based on that step, i.e. the predicted step is halved or doubled until

$$f(x + at) < f(x) + .1af'(x)^T t$$

$$f(x + 2at) > f(x) + .2af'(x)t.$$

7. Numerical Results

7.1 Comparison Algorithms

In order to justify proposing a new conjugate gradient code at this time it was felt necessary to demonstrate that it was indeed an improvement over existing available codes. We therefore chose for comparison purposes first the two standard Harwell routines VA08A and VA14A [4], second the quoted results in Buckley and Le Nir [15] and also the two earlier Hatfield Codes CONGRA, Dixon [6] and CONLS, Dixon [9].

Results for four versions of the new code are included for completeness, with and without the preconditioning matrix H and with and without the restart test (33), as it was felt of interest to indicate the effect of these innovations.

For each algorithm the amount of computational effort is measured in terms of the effective function evaluation (EFE)s given by EFE = $N_F + n\, N_G$

n = number of unknowns

N_F = number of function calls

N_G = number of gradient calls.

In all cases the termination criterion was set at

$$g^T g < \varepsilon = 1E\text{-}4$$

and the constant C_1 in Wolfe's Condition I was set at 1E-3.

7.2 Test Problems

The following are the details of the test functions used.

1. PWSING (POW n)

$$I(x) = \sum_{j=1}^{n/4} \left[(x_{4j-3} + 10x_{4j-2})^2 + 5(x_{4j-1} - x_{4j})^2 + (x_{4j-2} - 2x_{4j-1})^4 + 10(x_{4j-3} - x_{4j})^4 \right]$$

$$\underline{x}^{(0)} = (3, -1, 0, 3, \ldots)^T$$

2. EXTROS

$$I(x) = \sum_{i=1}^{n/2} \left[100(x_{2i} - x_{2i-1}^2)^2 + (1 - x_{2i-1})^2\right]$$

$$\underline{x}^{(0)} = (-1.2, 1, \ldots)^T$$

3. TRIDIA

$$I(x) = \sum_{i=2}^{n} \left[i(2x_i - x_{i-1})^2\right]$$

$$\underline{x}^{(0)} = (1, 1, 1, 1, \ldots)^T$$

4. ROS2

$$I(x) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2$$

$$\underline{x}^{(0)} = (-1.2, 1)^T$$

5. ROS8

$$I(x) = 100(x_1^2 - x_2)^8 + (1 - x_1)^8$$

$$\underline{x}^{(0)} = (-1.2, 1)^T$$

6. TRIGNM

$$I(x) = \sum_{i=1}^{n} \left[n + i - \sum_{j=1}^{n} (A_{ij}\mathrm{Sin}(x_j) + B_{ij}\mathrm{Cos}(x_j))\right]^2$$

$$A_{ij} = \delta_{ij} \qquad B_{ij} = i\delta_{ij} + 1$$

$$\delta_{ij} = \begin{cases} 0 \text{ if } i \neq j \\ 1 \text{ if } i = j \end{cases} \qquad \underline{x}^{(0)} = (\tfrac{1}{n}, \tfrac{1}{n}, \ldots)$$

7. BOXFUN

$$I(x) = \sum_{j=1}^{n/2} \sum_{i=1}^{10} \left[\exp(-x_{2j-1}.t_i) - 5\exp(-x_{2j}t_i) - \exp(-t_i) + 5\exp(-10t_i)\right]^2$$

$$t_i = \frac{i}{10} \qquad i = 1, 2, \ldots, 10 \qquad \underline{x}^{(0)} = (5, 0, 5, 0, \ldots)$$

8. EXP2

$$I(x) = \sum_{i=1}^{10} (e^{-x_1 z_i} - 5e^{-x_2 z_i} - e^{-z_i} + 5e^{-10z_i})^2$$

where $z_i = i/10$ $\qquad x^{(0)} = (1, 2)^T$

9. EXP3

$$I(x) = \sum_{i=1}^{10} (e^{-x_1 z_i} - x_3 e^{-x_2 z_i} - e^{-z_i} + 5e^{-10 z_i})^2$$

where $z_i = {}^{i}/_{10}$ $\qquad x^{(0)} = (1, 2, 1)^T$

10. EXP4

$$I(x) = \sum_{i=1}^{10} (x_3 e^{-x_1 z_i} - x_4 e^{-x_2 z_i} - e^{-z_i} + 5e^{-10 z_i})^2$$

where $z_i = {}^{i}/_{10}$ $\qquad x^{(0)} = (1, 2, 1, 1)^T$

11. WOOD4

$$I(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2$$
$$+ 10.1\{(x_2 - 1)^2 + (x_4 - 1)^2\} + 19.8(x_2 - 1)(x_4 - 1)$$
$$x^{(0)} = (-3, -1, -3, -1)^T$$

12. NONDIA

$$I(x) = \sum_{i=2}^{n} \left[100(x_1 - x_i^2)^2 + (1 - x_2)^2\right]$$
$$x^{(0)} = (-1.2, 1, \ldots)^T$$

7.3 The Numerical Results

The numerical results have been presented in two tables. In Table 1 the results are for the classic small dimensional test problem. In Table 2 results are presented for some larger dimensional problems. This latter set of results are the more interesting. Surprisingly the results without the preconditioning matrix H (i.e. OPCG 2 and 4) are better than those with it and this led to the discovery that the 3 term conjugate gradient formula generates conjugate directions for any given starting direction t° [reported in Dixon [16] and previously in Nocedal [13]]. As the choice of H used only had the effect of making the first direction the steepest descent direction in the transformed space, it is no longer necessary for this purpose and could be chosen to have other desirable properties. The results

with the reset test were usually but not universally better than those without and it has been retained; the code OPCG2 being adopted as the definitive version. On comparing the results of this code with those of other codes on the large dimensional problems in Table 2 it obviously well outperforms VA08A and VA014A and the results are also an improvement on those reported in Buckley and Le Nir [15]. On the less interesting small dimensional problems the situation is less clear. Out of interest an algorithm was written combining the Fletcher Reeves conjugate gradient formula with the line search strategy incorporated in the new code. This is reported in the tables as FRCG and, as expected, the new code (OPCG2) also regularly performed better than this modification, though this simple code did surprisingly well.

8. Conclusions

In this paper a modified form of the conjugate gradient algorithm is motivated, having desirable restart properties that outperforms a wide subset of current conjugate gradient codes. The new code contains provision for a preconditioning matrix that ought to lead to a further improvement in performance.

	BOX2	BOX4	EXP2	EXP3	EXP4	ROS2	ROS8	WOOD4	POW4	TRIGNM4
OPCG1	-	76	-	-	-	386	90	-	127	68
OPCG2	81	70	59	84	143	170	37	F4	119	56
OPCG3	-	136	-	-	-	386	90	-	127	53
OPCG4	81	70	59	67	140	170	37	F4	126	53
VA14A	36	80	45	92	120	195	F1	F2	345	44
VA08A	63	280	156	147	690	276	300	640	365	170
CONGRA	272	620	84	600	1251	366	99	390	446	1035
CONLS	80	690	62	134	362	170	247	1680	472	1060
FRCG	178	502	104	87	111	495	F3	1111	104	81

Table 1. Small Functions

F1 the run terminated with f = 0.018 (f* = 0)

F2 the run terminated with f = 1.6 (f* = 0)

F3 the run terminated with f = 1.93E-3 (f* = 0)

F4 the run terminated with f = 7.8, a stationary point which satisfies the convergency criteria.

	POW		EXTROS		TRIDIA			NONDIA	TRIGNM
	60	80	10	20	10	20	30	10	40
OPCG1	2111	6646	974	4111	119	439	831	-	225
OPCG2	1688	2055	393	801	119	439	831	702	225
OPCG3	2183	1566	782	2326	119	439	831	-	225
OPCG4	2122	2964	454	888	119	439	831	708	225
BLALN	7991	5913	671	1281	209	819	1891	858	-
VA14A	2623	3483	671	1281	297	819	1643	935	410
VA08A	9577	16443	957	2100	440	1029	2015	1100	4100
FRCG	3906	7463	1362	2511	119	439	863	F5	224

Table 2. Large Functions

F5 the run terminated with f = 1.8 (f* = 0)

Algorithm Code

OPCG1 = with preconditioner H and RST

OPCG2 = without preconditioner H and with RST

OPCG3 = with preconditioner H and without RST

OPCG4 = without preconditioner H and without RST

BLALN = Buckley and Le Nir [9] method

VA14A = Harwell subroutines

VA08A = Harwell subroutines

CONGRA = Fletcher/Reeves [10]

CONLS = based upon Hestenes and Stiefel but incorporating the gradient prediction method

FRCG = Fletcher/Reeves with the new line search

References

1. Hestenes, M and Stiefel, E
Methods of conjugate gradients for solving linear systems, Journal of Research of National Bureau of Standards, Vol.49, pp 409-436, 1952.

2. Fletcher, R and Reeves, C M
Function minimisation by conjugate gradients, Computer Journal, Vol.17, pp 149-154, 1964.

3. Polak, E and Ribiere, G
Note sur la convergence de methodes de directions conjugées, Rev. Francaise Informat Recherche Operationnelle, 3e Année, No.16, pp 35-43, 1969.

4. Powell, M J D
Restart procedures for the conjugate gradient method, Mathematical Programming, Vol.12, pp 241-254, 1977.

5. Wolfe, P
Convergence conditions for ascent methods, SIAM Review, Vol.11, pp 226-235, 1969.

6. Dixon, L C W
Nonlinear optimisation: A survey of the state of the art, in Software for Numerical Mathematics, (ed. D J Evans), Academic Press, London and New York, 1974.

7. Cohen, A
Rate of convergence of several conjugate gradient algorithms, SIAM Journal, Vol.9, pp 248-259, 1972.

8. Dixon, L C W
On quadratic termination and second order convergence, two properties of unconstrained optimisation algorithms, in Towards Global Optimisation, (eds. L C W Dixon and G P Szegö), North Holland Press, 1975.

9. Dixon, L C W
Conjugate directions without line searches, Journal of Institute of Mathematics and its Applications, Vol.11, pp 317-328, 1973.

10. Nazareth, L
A conjugate gradient algorithm without line searches, Journal of Optimisation Theory and Applications, Vol.23, No.3, pp 373-387, 1977.

11. Beale, E M L
A derivation of conjugate gradients, in Numerical Methods for Nonlinear Optimisation, (ed. F A Lootsma), Academic Press, London, 1972.

12. Allwright, J C
Improving the conditioning of optimal control problems using simple methods, in Recent Mathematical Developments in Control, (ed. O J Bell), Academic Press, 1972.

13. Nocedal, J
On the method of conjugate gradients for function minimisation, PhD thesis, Rice University, 1978.

14. Armijo, L
Minimization of functions having Lipschitz continuous first partial derivative, Pacific Journal of Mathematics, Vol.16, 1966.