AN UNCONSTRAINED OPTIMIZATION ALGORITHM WHICH USES FUNCTION AND GRADIENT VALUES*

J.E. Dennis and H.H.W. Mei

TR 75-246

June 1975

Department of Computer Science
Cornell University
Ithaca, New York 14853

*Research supported by National Science Foundation under grant number GJ-40903.

AN UNCONSTRAINED OPTIMIZATION ALGORITHM WHICH USES FUNCTION AND GRADIENT VALUES*

J.E. Dennis Jr. and H.H.W. Mei

ABSTRACT

A new method for unconstrained optimization is presented. It consists of a modification of Powell's 1970 dogleg strategy with the approximate Hessian given by Davidon's 1975 updating scheme which uses the projections of Δx and Δg in updating H and G and optimizes the condition number of $H^{-1}H_+$. This new algorithm performs well without Powell's special iterations and singularity safeguards. Only symmetric and positive definite updates to the Hessian are used.

*Research supported by National Science Foundation under grant number GJ-40903.

AN UNCONSTRAINED OPTIMIZATION ALGORITHM WHICH USES FUNCTION AND GRADIENT VALUES*

J.E. Dennis Jr. and H.H.W. Mei

1. Introduction

Our premise in this work is as follows: When one has a poor approximation to an unconstrained minimizer, quick improvement is likely to result from taking a step along the negative gradient. This approach bogs down as the minimum is approached. But when a good approximate minimizer is available the quasi-Newton methods often improve it quickly to acceptable accuracy.

M.J.D. Powell (1970 a,b) introduced a strategy, implemented in the unconstrained minimization routine MINFA (Powell (1970 c,d)), which can be viewed essentially as an elegant way to change over from gradient to quasi-Newton steps as the approximate minimizer is improved. Powell's program performs very well but one third of the iterations are so-called "special iterations" which experience shows to be necessary in practice although they serve no obvious purpose except to improve the particular quasi-Newton approximation to the Hessian used by the algorithm.

In Section 2 we describe Powell's strategy which has come to be known as the "Dogleg". An implementation which uses the Hessian rather than a quasi-Newton approximation, has been quite successful in applications at the National Bureau of Economic Research. Experience there indicates that it is worthwhile to modify the Hessian if necessary in order to ensure positive definiteness. Furthermore, users at NBER have suggested that the dogleg strategy

*Research supported by National Science Foundation under grant number GJ-40903.

could be improved by introducing an earlier bias toward the Newton direction. In Section 3 we set forth our way of doing this.

In Section 4 we briefly describe the scheme we will use for maintaining the quasi-Newton approximate Hessian. It is basically the new scheme given in Davidon (1976) although some changes are made since we do not always step in the quasi-Newton direction. This not only allows us to maintain a positive definite Hessian for our quadratic model function but we will also indicate in Section 5 why we first got the idea which is born out in practice that we can dispense with the special iterations which are important to the performance of MINFA.

Section 5 is devoted to a detailed description of our algorithm and a listing of the Fortran implementation MINOP is affixed to the end of this report. Section 6 gives test results on some standard problems and comparisons with Powell's MINFA, Davidon's OCOPTR and Fletcher's (1970) VMM01. Although our method seems at best marginally better in these tests, we feel intuitively that it will show in an even better light on more difficult problems. At any rate, we hope it will be regarded as a useful alternative to OCOPTR.

2. The Dogleg Strategy

Suppose a step bound Δ_k is given at the start of the kth iteration, let G_k and H_k denote the hessian approximation and its inverse respectively. Let

$$\phi_k(d) = f(x_k) + d^T g_k + \frac{1}{2} d^T G_k d$$

be the quadratic approximation to $f(x+d)$ in the kth iteration where $g_k = \nabla f(x_k)$ is the gradient of f at x_k. A useful device is to think of $\{x_k+d : ||d|| \le \Delta_k\}$ as the region in which we trust $\Phi_k(d)$ as an approximation to $f(x_k+d)$. We will use $||\cdot|| = ||\cdot||_2$. In order to find x_{k+1} Powell first calculates the "Cauchy point" i.e. the point $x_k - \alpha^* g_k$ where

$$\alpha^* = \frac{g_k^T g_k}{g_k^T G_k g_k} \text{ which minimizes } \Phi_k(-\alpha g_k) .$$

If the Cauchy point is outside the circle centered at x_k with radius Δ_k then $x_{k+1} = x_k - \beta\alpha^* g_k$ where β is chosen so that $||\beta\alpha^* g_k|| = \Delta_k$. The reasoning is that the gradient is still large so we should continue to move along it. If the Cauchy point is inside the trust region then we assume it is time to calculate the "Newton point" $x_k - H_k g_k$. If the Newton point is inside the circle, we pick $x_{k+1} = x_k - H_k g_k$ otherwise we pick $x_{k+1} = x_k+d_k$ where $||d_k|| = \Delta_k$ and $d_k = (1-\theta_n)(-a^* g_k) + \theta_k(-H_k g_k)$ with $0 \le \theta_k \le 1$. If $f(x_{k+1})$ is larger than $f(x_k)$, we reject x_{k+1}, halve Δ_k and repeat this step. Eventually we would try a short enough gradient step to reduce f. We then find Δ_{k+1}, update G_k, H_k and then proceed to the k+1th iteration until a point x^* such that $||\nabla f(x^*)|| \le \varepsilon$ is reached and the procedure stops. The number ε is a user supplied desired bound on the final gradient.

Any reader who is interested in details of how Δ_k and θ_k are revised and calculated should read Powell (1970).

Definition. A direction p_k is said to be a descent direction if there exists a $\lambda_k > 0$ such that $f(x_k + \lambda p_k) < f(x_k)$ for $\lambda \in (0, \lambda_k)$.

In Powell's (1970) MINFA routine, the Powell symmetric Broyden update, or the PSB update was used to maintain G_k, H_k. This technique did not attempt to preserve positive definiteness. Therefore the Newton direction $n_k = -H_k g_k$ does not have to be a descent direction. This was probably the reason why Powell had considered the Cauchy point before the Newton point (since the gradient direction is a descent direction). However, when positive definite updates are used, both directions are descent directions and one may wonder which strategy will yield a better result, strategy I, — the strategy that considers the Cauchy point first, or strategy II, — the strategy that takes a Newton step if $||n_k|| \leq \Delta_k$. Our analysis in the next section will give an answer to this question.

3. Double Dogleg Strategy

One of the reasons why people use Powell's dogleg method is that it has excellent global convergence properties. However when we are fairly close to the solution a dogleg step seems to have too much bias toward the gradient direction and results in apparently slower convergence. We modified Powell's dogleg step so that more bias toward the Newton step is introduced in it in the middle stages of the iteration.

We first prove a lemma:

Lemma 3.1: Let x and y be any two vectors in R^n and A be any $n \times n$ real symmetric positive definite matrix. The following inequality is true:

$$(x^T y)^2 \le (x^T A x)(y^T A^{-1} y).$$

Equality holds if and only if $y = \beta A x$ for some scalar β.

Proof: Since A is symmetric and positive definite, there exists a symmetric and positive definite matrix B such that $A = B^2$. Therefore,

$$\begin{aligned}(x^T y)^2 &= (x^T B\, B^{-1}\, y)^2 = [(Bx)^T (B^{-1}\, y)]^2 \\ &\le (x^T B\, B\, x)(y^T B^{-1}\, B^{-1}\, y) \\ &= (x^T A x)(y^T A^{-1}\, y)\end{aligned}$$

by the Cauchy-Schwarz inequality. /

Let p_k be the Cauchy step and n_k be the Newton step

i.e. $p_k = -\dfrac{||g_k||^2}{g_k^T G_k g_k}\, g_k$ and $n_k = -H_k g_k$.

When $||p_k|| < \Delta_k$ and $||n_k|| > \Delta_k$ we first look for the point in the Newton direction at which the reduction of the quadratic approximation to f resulting from taking this step is the same as taking the Cauchy step. That is, we determine t such that

$$\phi_k(-t\, H_k g_k) = \phi_k(p_k) = \phi_k\left(-\frac{||g_k||^2}{g_k^T G_k g_k}\, g_k\right)$$

which yields

$$t^2\left(\tfrac{1}{2}\, g_k^T\, H_k\, g_k\right) - t(g_k^T\, H_k\, g_k) + \frac{1}{2}\,\frac{||g_k||^4}{g_k^T G_k g_k} = 0,$$

and

$$t = \frac{g_k{}^T H_k g_k \pm \sqrt{(g_k{}^T H_k g_k)^2 - (||g_k||^4 \; g_k{}^T H_k g_k / g_k{}^T G_k g_k)}}{g_k{}^T H_k g_k}$$

$$= 1 \pm \sqrt{1 - \frac{||g_k||^4}{g_k{}^T H_k g_k g_k{}^T G_k g_k}} \quad .$$

$$= 1 \pm \sqrt{1-c} \;, \qquad \text{where} \quad c = \frac{||g_k||^4}{g_k{}^T H_k g_k \; g_k{}^T G_k \; g_k}$$

From lemma 3.1 we have $c \leq 1$ and therefore t is well defined.

We then take the dogleg step between p_k and tn_k if $||tn_k|| > \Delta_k$, and in practice it seems best to take t as $t = 1 - \alpha\sqrt{1-c}$ where α is some positive number less than 1 which we discuss below. But first see Figure 1 where point 1 is Powell's step and point 2 is the double dogleg step.

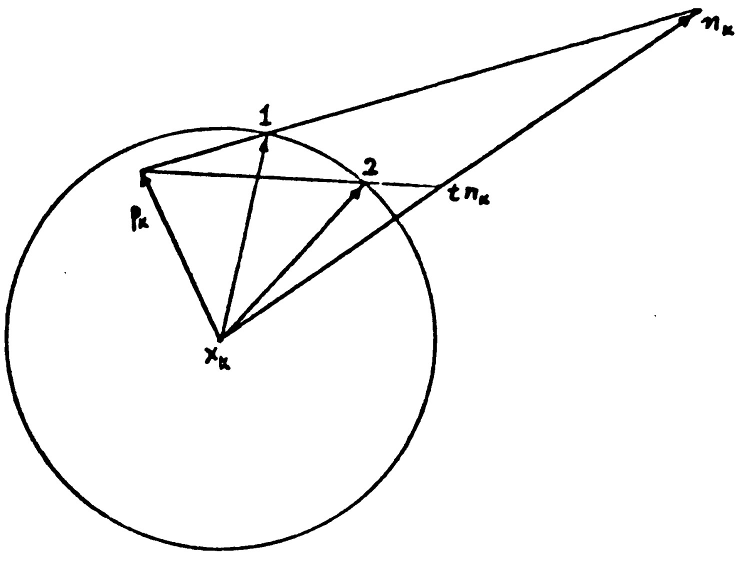

Figure 1

We would like to choose the parameter α in such a way so that Φ_k is monotonically decreasing along the line from $\delta = p_k$ to $\delta = tn_k$. We carry out the derivation as follows:

Let $t = 1 - \alpha\sqrt{1-c}$ and $\delta_k(\theta) = (1-\theta)p_k + \theta t n_k$. What we'd like to find is the value of α such that

$$\frac{\partial}{\partial\theta} \Phi_k(\delta(\theta)) \leq 0 \quad \forall\ \theta \in [0,1]$$

Now,
$$\delta(\theta) = -\left[(1-\theta)\frac{||g_k||^2}{g_k^T G_k g_k} g_k + \theta\, t\, H_k\, g_k\right]$$

$$\delta'(\theta) = \frac{||g_k||^2}{g_k^T G_k\, g_k}\ g_k - t\ H_k\ g_k$$

and

$$\frac{\partial}{\partial\theta} \Phi_k(\delta(\theta)) = g_k^T\ \delta'(\theta) + \delta(\theta)^T\ G_k\ \delta'(\theta)$$

$$= g_k^T\ \left[\frac{||g_k||^2}{g_k^T G_k\, g_k} g_k - t\ H_k\ g_k\right]$$

$$- \left[(1-\theta)\ \frac{||g_k||^2}{g_k^T G_k\, g_k} g_k + \theta\ t\ H_k g_k\right]^T G_k\left[\frac{||g_k||^2}{g_k^T G_k g_k} g_k - t\ H_k g_k\right]$$

$$= \frac{||g_k||^4}{g_k^T G_k\, g_k}\ [\theta(1-t) + (1-\theta)t] - g_k^T H_k g_k [t - \theta t^2]$$

Hence $\frac{\partial}{\partial\theta} \Phi_k\ (\delta(\theta)) \leq 0 \quad \forall\ \theta \in [0,1]$ if and only if

$$h(\theta) = c[\theta(1-t) + (1-\theta)t] - [t-\theta t^2] \leq 0 \quad \forall\ \theta \in [0,1].$$

Since

$$h'(\theta) = c\ [(1-t) - t] + t^2$$
$$= (t-c)^2 + c(1-c) > 0$$

This implies $h(\theta)$ is monotonically increasing and therefore $h(\theta) \leq h(1)$ $\forall\ \theta \in [0,1]$. We'll choose α such that

$$h(1) = t^2 - t + c(1-t) < 0 .$$

Since $$h(1) = \alpha\sqrt{1-c}[(c-1) + \alpha\sqrt{1-c}]$$

We can choose any α, $0 < \alpha < \sqrt{1-c}$. In our implementation we have used $\alpha = 0.8\ \sqrt{1-c}$.

The following theorem enables us to answer the question raised before about Strategy 1 versus Strategy 2.

<u>Theorem 1</u>. Let G_k be symmetric and positive definite and $H_k = G_k^{-1}$. If $t = 1 - \alpha\sqrt{1-c}$ where α is such that $0 \leq \alpha \leq 1-c$ and $c = \dfrac{||g_k||^4}{g_k^T H_k g_k\ g_k^T G_k g_k}$. Then $||p_k|| \leq ||tn_k||$, where p_k, n_k denote respectively the Cauchy and Newton steps for Φ_k.

<u>Proof</u>: If $||g_k|| = 0$ we are finished so we consider the other case. Now,

$$||p_k|| = \frac{||g_k||^3}{g_k^T G_k g_k} = \frac{||g_k||^4\ ||H_k g_k||}{||g_k||\ ||H_k g_k||\,g_k^T G_k g_k}$$

$$\leq \frac{||g_k||^4 ||H_k g_k||}{g_k^T H_k g_k\ g_k^T G_k g_k} = [1-(1-c)]\ ||H_k g_k||$$

$$\leq [1 - \sqrt{1-c}]\ ||H_k g_k|| = ||tn_k|| \ ./$$

Since $t \leq 1$, one implication of this theorem is that the length of the Cauchy step is no greater than the t-multiple of the <u>Newton</u> step.

<u>Corollary</u>: If G_k is symmetric and positive definite and $H_k = G_k^{-1}$ then in the double dogleg method, strategy I and strategy II as described yield identical points.

<u>Proof</u>: The only time the two strategies can differ is when the Cauchy point $x_k + p_k$ is not strictly inside the trust region and the relaxed Newton point $x_k + tn_k$ is not strictly outside. Since $||p_k|| \leq ||tn_k||$, this condition could only be met if $||p_k|| = t||n_k||$. From the proof above and lemma 3.1, this occurs if and only if $||g_k|| \cdot ||H_k g_k|| = g_k^T H_k g_k$ and $c = 1$ which by the Cauchy-Schwarz inequality implies $g_k = \beta H_k g_k$ for some β.

So $p_k = tn_k$ and again the two strategies must yield the same point. /

Our new strategy can now be described as follows:

1. Compute n_k and $||n_k||$. If $||n_k|| \leq \Delta_k$ we take $\Delta x_k = n_k$. If $||n_k|| > \Delta_k$ we compute $||tn_k||$ where t is defined in Theorem 1. If $||tn_k|| \leq \Delta_k$ we take $x_k = \dfrac{\Delta_k}{||n_k||} n_k$.

2. If $||tn_k|| > \Delta_k$ we then compute the Cauchy step

$$p_k = \frac{-||g_k||^2}{g_k^T G_k g_k} g_k$$ and if $||p_k||_2 \geq \Delta_k$ we take

$$\Delta x_k = - \frac{\Delta_k}{||g_k||} g_k.$$

3. If $||p_k|| < \Delta_k$ we take $\Delta x_k = p_k + \theta_k tn_k$ where θ_k is chosen such that $0 < \theta_k < 1$ and $||\Delta x_k|| = \Delta_k$.

4. Updating H_k and G_k.

In 1967 Broyden introduced a one parameter family of updates using a parameter β. Fletcher (1970) showed that the β-class could be rewritten in the following form with a parameter $\phi = \beta\langle y,s\rangle$

$$H_\phi = H - \frac{Hyy^TH}{a} + \frac{ss^T}{b} + \phi a \left(\frac{s}{b} - \frac{Hy}{a} \right) \left(\frac{s}{b} - \frac{Hy}{a} \right)^T$$

where $a = y^T Hy$, $b = y^T s$, $c = s^T H^{-1} s = s^T Gs$. If we apply the Sherman-Morrison-Woodbury Theorem three times we can find $G_\phi = H_\phi^{-1}$ can be written as

$$G_\phi = G + \frac{(y-Gs)y^T + y(y-Gs)^T}{b} - \frac{\langle y-Gs,s\rangle}{b^2} yy^T$$

$$- \frac{\phi a}{\sigma} \left(\frac{Gs}{b} - \frac{cy}{b^2} \right) \left(\frac{Gs}{b} - \frac{cy}{b^2} \right)^T$$

where $\sigma = 1 + \phi \left[\frac{ac}{b^2} - 1\right]$.

Davidon [1976] used a scheme in which projections of Δx and Δg rather than the quantities themselves were used as s and y to update H and $\bar{G}$. The parameter ϕ was chosen such that the ratio of the largest to smallest eigenvalues is minimized. It is known that if H is symmetric and positive definite and ϕ satisfies $\frac{b}{a} + \phi(\frac{c}{b} - \frac{b}{a}) > 0$ then H_ϕ is symmetric and positive definite.

For Davidon's derivation, Δx was a multiple of the Newton step. We remove this restriction but since the scheme is essentially the same as Davidon's, we do not repeat it here. Any reader who is interested should consult Davidon's (1976) paper. However, because it will be useful in the next section, we state without proof the following theorem from that paper.

<u>Theorem 2</u>: Among all positive definite H_ϕ in the ϕ-class, there is a unique one which minimizes the condition number of $H^{-1}H_+$. When $b \le \frac{2ac}{a+c}$ it is the one with $\phi = \frac{b(c-b)}{ac-b^2}$ and when $b \ge \frac{2ac}{a+c}$ it is the one with $\phi = \frac{b}{b-a}$.

We note that for $\phi = \frac{b}{b-a}$, H_ϕ is the symmetric rank one or SR1 update.

5. The Algorithm

Our subroutine MINOP has the following parameters:

N , the dimension of the problem.

X , an Nx1 matrix which gives the initial guess of the solution vector.

F , a variable containing the current least calculated value of $f(x)$ corresponding to x .

G , a vector variable whose ith component contains the value of the ith component of the gradient $g(x)$ at x.

STEP , a positive number provided by user as the radius of the initial trust regions Δ_1.

ACC , an user provided desired bound on the magnitude of the gradient. MINOP stops when $||g(x_k)|| \leq$ ACC.

MAXFUN , a user provided integer which is an upper bound on the number of function calls allowed.

IPRINT , a positive integer that controls the frequency of output from MINOP. If IPRINT = 0 no printing is done, otherwise the best calculated x and f(x) are printed after every |IPRINT| iterations, and if IPRINT is positive, g(x), the gradient will be printed also.

The user has to provide two subroutines to calculate f(x) and g(x) as SUBROUTINE CALCF (N,X,F) and SUBROUTINE CALCG (N,X,G).

The implementation proceeds as follows:

START: Call f(x) and g(x) , $f = f(x)$ $g = g(x)$

$$G = \frac{0.01 \; ||g||_2}{\Delta} \cdot I \quad \text{and} \quad H = G^{-1}$$

$$z = -\frac{\Delta}{||g||_2} g , \qquad \text{FLAG} = 0.$$

STEP 1: If $||g|| \leq$ ACC then STOP;

Else compute $n = -Hg$ and $||Hg||$

If $||Hg||_2 < \Delta$ then $\Delta x = -Hg$ and go to STEP 4.

Otherwise compute

$$c = \frac{||g||^4}{(g^T Gg)(g^T Hg)}, \quad \alpha = 0.8\sqrt{1-c} \quad \text{and} \quad t = 1-\alpha\sqrt{1-c} = 0.2 + 0.8c$$

If $||tHg|| \le \Delta$ then $\Delta x = -\frac{\Delta}{||Hg||_2} Hg$ and go to STEP 4.

STEP 2: $n = tn$ and compute the Cauchy step

$$p = -\frac{||g||^2}{g^T Gg} g$$

If $||p||_2 = \frac{||g||^3}{g^T Gg} \ge \Delta$ then $\Delta x = -\frac{\Delta}{||g||} g$

and go to STEP 4.

STEP 3: $w = n - p$ compute

$$\text{THETA} = \frac{[\Delta^2 - ||p||^2]}{\langle p,w\rangle + [\langle p,w\rangle^2 + ||w||^2 (\Delta^2 - ||p||^2)]^{\frac{1}{2}}}$$

and

$$\Delta x = p + \text{THETA} * w$$

STEP 4: $xa = x + \Delta x$. Call $f(xa)$ and $fa = f(xa)$

If $fa < f$ go to STEP 5,

else $\Delta = \frac{1}{2}\Delta$ and go to STEP 1.

STEP 5: Call $g(xa)$ and $ga = g(xa)$, $\Delta g = ga - g$

If $fa - f > 0.1(g^T\Delta x + \frac{1}{2} \Delta x^T G \Delta x)$

then $\Delta = \frac{1}{2}||\Delta x||$ and go to STEP 6,

else $\Delta = ||\Delta x||$

If $||\Delta g - G\, \Delta x|| \le \frac{1}{2}||g||$

then $\Delta = 2||\Delta x||$ and go to STEP 6,

else if $g^T \Delta x \ge 2\, \Delta x^T\, ga$

then $\Delta = 2\, ||\Delta x||$.

STEP 6: $f = fa$, $x = xa$, $g = ga$

$b_0 = \Delta g^T \Delta x$, If $b_0 < 10^{-30}$ go back to STEP 1.

If this is the first iteration go to STEP 7.

If FLAG = 1 then $z = \Delta x$,

else define $c_2 = q^T \Delta x$,

$c_3 = \gamma^T \Delta x$,

and $z = c_1 \cdot [c_2 z + c_3 v] - \left(\frac{y^T \Delta x}{z^T y} \right) z$.

Remark: If $\Delta g^T \Delta x$ is positive but too small, we follow Davidon (1976) in declining to update H and G .

If $\Delta g^T \Delta x < 0$ there do not exist positive definite H_{k+1} and G_{k+1} in the class for which $H_{k+1}\, \Delta g_k = \Delta x_k$, we therefore do not update H and G and go back to STEP 1.

When FLAG = 1, z_{k-1} and v_{k-1} computed in the previous iteration are linearly dependent and we reassign z_{k-1} as Δx_k in the hope that they will become linearly independent. Of all the tests we made in Section 6, however, FLAG was never set to 1.

z_k is obtained by applying a projection matrix Q_k to Δx_k, i.e. $z_k = Q_k\, \Delta x_k$ where

$$Q_k = P_{k-1} - \frac{z_{k-1}\, y_{k-1}^T}{z_{k-1}^T\, y_{k-1}}$$

and P_{k-1} is a projection matrix whose column space is the linear span of z_{k-1} and v_{k-1}.

In fact $c_1 = \dfrac{1}{\alpha_1\alpha_3 - \alpha_2^2}$,

$$c_2 = (\alpha_3 z_{k-1}{}^T G_{k-1} - \alpha_2 v_{k-1}{}^T G_{k-1}) , \Delta x_k ,$$

and

$$c_3 = (-\alpha_2\, z_{k-1}{}^T G_{k-1} + \alpha_1\, v_{k-1}{}^T G_{k-1}) \quad \Delta x_k .$$

These numbers and vectors q, γ and the projection matrix P_{k-1} will all be constructed in the next step.

STEP 7: Compute the vectors $v = \Delta x - H\Delta g$, Gv and Gz.

Compute the numbers $\alpha_1 = z^T Gz$, $\alpha_2 = z^T Gv$, $\alpha_3 = v^T Gv$,

$$c_1 = \frac{1}{\alpha_1\alpha_3 - \alpha_2^2} ,$$

$$c_4 = \alpha_3\ (z^T\Delta g) - \alpha_2 (v^T\Delta g) ,$$

$$c_5 = - \alpha_2\ (z^T\Delta g) + \alpha_1 (v^T\Delta g) ,$$

$$y = c_1\ (c_4\ Gz + c_5\ Gv) ,$$

$$q = \alpha_3\ Gz - \alpha_2\ Gv ,$$

$$\gamma = -\alpha_2\ Gz + \alpha_1\ Gv ,$$

$$s = c_1\ ((q^T\Delta x) z + (\gamma^T\Delta x) v) .$$

<u>Remark:</u> A rank 2 projection matrix P_k is implicitly constructed in this step and

$$P_k = \frac{1}{\alpha_1\alpha_3 - \alpha_2^2} [z_k, v_k] \begin{bmatrix} \alpha_3 z_k{}^T G_k - \alpha_2 v_k{}^T G_k \\ -\alpha_2 z_k{}^T G_k + \alpha_1 v_k{}^T G_k \end{bmatrix} .$$

Any interested reader can easily verify that the following is true:

$$P_k z_k = z_k \ , \ P_k v_k = v_k \quad \text{and}$$

$$P_k P_k = P_k \ , \ P_k H_k = H_k P_k^T \ .$$

The vectors y and s are projections of Δg and Δx respectively that will be used in updating H and G and they are defined as $y = P^T \Delta g$, $s = P\Delta x$.

We can now observe that the projection Q defined in STEP 6 has the following properties:

(1) Since . $z_{k-1}^T y_{k-1} = z_{k-1}^T P_{k-1}^T \Delta g_{k-1} = z_{k-1}^T \Delta g_{k-1}$,

$$Q_k = P_{k-1} - \frac{z_{k-1} y_{k-1}^T}{z_{k-1}^T \Delta g_{k-1}} \ ,$$

(2) $Q_k Q_k = Q_k$,

(3)
$$\begin{aligned} \Delta g_{k-1}^T z_k &= g_{k-1}^T Q_k \Delta x_k \\ &= (\Delta g_{k-1}^T P_{k-1} - \frac{\Delta g_{k-1}^T z_{k-1} y_{k-1}^T}{z_{k-1}^T \Delta g_{k-1}}) \quad \Delta x_k \\ &= (y_{k-1} - y_{k-1})^T \Delta x_k \\ &= 0 \ . \end{aligned}$$

Since $H_k \Delta g_{k-1} = \Delta x_{k-1}$ we see that

$$\Delta x_{k-1}^T G_k z_k = 0$$

Hence the z_k defined in STEP 6 is orthogonal to the last gradient change and conjugate to the last step made. We make the admittedly hueristic assumption that this property can substitute for special iterations.

$$(4) \qquad Q_k z_{k-1} = P_{k-1} z_{k-1} - \frac{z_{k-1} y_{k-1}^T z_{k-1}}{z_{k-1}^T y_{k-1}} = z_{k-1} - z_{k-1} = 0.$$

STEP 8: Set $b = y^T s$. If $b \leq 10^{-30}$, then $b = b_0$, $u = \Delta g$, $s = \Delta x$

otherwise $u = y$.

Compute $a = u^T H u$,

$b = u^T s$,

$c = s^T G s$.

If $b \leq \frac{2ac}{a+c}$ assign $\phi = \frac{b(c-b)}{ac-b^2}$.

If $b > \frac{2ac}{a+c}$ assign $\phi = \frac{b}{b-a}$.

Remark: If $y^T s$ is too small or negative we will not use them to update H and G for the same reason as in the remark following STEP 6. Since we must already have $\Delta g^T \Delta x > 10^{-30}$, we use Δg and Δx to update H and G. However we save y_k since we need it to compute z_{k+1} in the next iteration.

Because only symmetric and positive definite H_k and G_k are generated we see that a, b, c are positive numbers and

$$b^2 = (u^T s)^2 = (u^T H^{\frac{1}{2}} G^{\frac{1}{2}} s)^2 \leq (u^T H u)(s^T G s) = ac$$

by the Cauchy-Schwarz inequality. Also, $b^2 = ac$ if and only if $Hu = \lambda s$. Since $v_k^T G_k v_k > 0$ by the positive definiteness of G_k we get

$b < \frac{a+c}{2}$ and when $Hu = \lambda s$ occurs this is equivalent to $b > \frac{2ac}{a+c}$. The reader can easily verify that when $b = a$, $b < \frac{2ac}{a+c}$. Therefore, ϕ is well defined.

STEP 9: Set $m = z - \left(\frac{z^T v}{v^T v} \right) v$,

$$\beta = \frac{(m^T v)^2}{(v^T v)(m^T m)}$$

If $\beta \geq 10^{-6}$ assign Flag = 1.

Remark: This step tests linear independence of the vectors z and v. The vector m is the projection of z onto $\{v\}^{\perp}$ and if $\beta \geq 10^{-6}$ we decide z and v are dependent.

STEP 10: Set $\sigma = 1 + \phi \left[\frac{ac}{b^2} - 1 \right]$

$$w_1 = \frac{s}{b} - \frac{Hu}{a}$$

$$w_2 = \frac{Gs}{b} - \frac{c}{b^2} u$$

$$w_3 = u - Gs$$

$$G = G + \frac{w_3 u^T + u w_3^T}{b} - \frac{b-c}{b^2} uu^T - \frac{\phi a}{\sigma} w_2 w_2^T ,$$

$$H = H - \frac{H u u^T H}{a} + \frac{s s^T}{b} + \phi a w_1 w_1^T$$

GO TO STEP 1

Remark: This step is the final updating of H and G. The formula for G is an application of Sherman-Morrison-Woodbury's formula. H_k is singular if and only if $\sigma = 0$. Among all the tests we made σ was never less than 10^{-30}.

This completes the algorithm.

6. Numerical Results

The following functions have been tested:

1. Wood-Colville's function

$$f(x) = 100\,(x_2-x_1^2)^2 + (1-x_1)^2 + 90\,(x_4-x_3^2)^2$$
$$+ (1-x_3)^2 + 10.1\,[(x_2-1)^2 + (x_4-1)^2] + 19.8\,(x_2-1)(x_4-1)$$

with 4 different starting points

(-3, -1, -3, -1) , (-1.2, 1, 1.2, 1) ,

(-3, 1, -3, 1) and (-1.2, 1, -1.2, 1)

2. Rosenbrock's function

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

with 5 different starts (-1.2, 1) , (2, -2)

(-3.635, 5.621) , (6.39, -0.221) , and (1.489, -2.547)

3. Box's 2-variable function

$$f(x) = \sum_{i=1}^{10} [(e^{-x_1 \cdot \frac{i}{10}} - e^{-x_2 \cdot \frac{i}{10}}) - (e^{-\frac{i}{10}} - e^{-i})]^2$$

with 5 different starts (5,0) , (0,0)

(0,20) , (2.5,10) and (5,20)

4. Powell's 4-variable function

$$f(x) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4.$$

with start (3, -1, 0, 1) (-0.1, 1, -0.1, 1) and (-0.6, 1, -0.6, 1)

Comparisons with Powell's MINFA, Davidon's 1975 OCOPTR and Fletcher's 1970 VMM01 are made.

We have slightly modified Davidon's OCOPTR so that separate accounting on number of function calls and gradient calls are made as described in his original paper. This modification did not change its performance but gives us a more accurate accounting for comparing with our routine MINOP.

The data $\Delta_1 = 10$ in testing Wood's function and $\Delta_1 = 3$ in testing Rosenbrock's, Powell's 4 variables, and Box's function are separately listed simply because we have these results for MINFA and VMM01 from Shanno and Phua (1974). Tests using other values of Δ_1 were also tested in order to make the comparisons more meaningful, e.g. $\Delta = 1,2,3,4,5$ for Rosenbrock's, Box's and Powell's 4-variable function and $\Delta = 6,8,10,12,14$ for Wood's function. An average number of function and gradient calls were calculated and a total of equivalent function evaluations (which is equal to the number of function calls plus N times the number of gradient calls) for all the points was provided. According to this number the percentage of improvement of MINOP over other routines with respect to each function is shown below. Comparisons with MINFA are based on the same specified initial Δ while the average of MINOP is used in comparing with OCOPTR and VMM01.

	MINFA	OCOPTR	VMM01
Wood's	36%	10%	35%
Rosenbrock	3%	3.6%	2%
Box - 2	28%	1.6%	20%
Powell - 4	29.2%	-1.7%	N.A.

All tests used the same stopping criterion, that is when the magnitude of the gradient is less than or equal to 10^{-5}.

We have also tested Box 3 variable function

$$f(x) = \sum_{i=1}^{10} [(e^{\frac{-x_1 i}{10}} - e^{\frac{-x_2 i}{10}}) - x_3(e^{\frac{-i}{10}} - e^{-i})]^2$$

with 10 different starts. However we decided that there is no base for making any comparison since Box - 3 function is "flat" on the line $x_1 = x_2$ and $x_3 = 0$, and different solutions were found in different tests. For example: MINOP with $\Delta = 1,2,3,4,5$ starting at (0,0,10) converged to (-1,-1,0), (-1,-1,0) (0.7,0.7,0) (1.2,1.2,0), (2.7,2.7,0) with NF and NG 11(11), 15(13), 13(13), 15(15), 33(28) respectively, while OCOPTR converged to (4.5,4.5,0) with NF and NG 26(22).

MINOP TEST RESULTS: NF(NG)

Wood's function	Δ = 6	8	10	12	14	Average.
(-3,-1,-3,-1)	57(50)	70(61)	71(61)	62(57)	64(57)	62.8(57)
(-1.2,1,1.2,1)	57(50)	60(51)	59(50)	60(52)	64(53)	60(51)
(-3,1,-3,1)	32(25)	64(61)	43(35)	32(26)	59(47)	49.4(38.8
(-1.2,.,-1.2,1)	107(95)	128(118)	117(103)	120(104)	121(111)	118.6(106.
Rosenbrock's function	**Δ = 1**	**2**	**3**	**4**	**5**	**Average**
(-1.2,1)	50(45)	49(45)	28(25)	44(36)	50(43)	44.2(38.8
(2,-2)	21(21)	35(35)	43(42)	37(35)	40(39)	35(34.4
(-3.635,5.621)	75(69)	55(53)	56(54)	55(50)	52(50)	58.6(55)
(6.39,-0.221)	60(58)	60(60)	61(60)	51(51)	33(32)	53(52)
(1.489,-2.547)	30(30)	27(27)	39(36)	29(28)	35(33)	32(30.8
Box's 2-vari. func.	**Δ = 1**	**2**	**3**	**4**	**5**	**Average**
(5,0)	24(22)	24(20)	23(20)	20(16)	23(19)	22.8(19.4
(0,0)	18(17)	19(17)	18(16)	28(22)	19(16)	20.4(17.6
(0,20)	14(14)	19(17)	28(25)	25(25)	27(25)	22.6(21)
(2.5,10)	8(6)	10(9)	8(5)	12(10)	12(7)	10(7.4)
(5,20)	16(14)	26(19)	23(17)	21(15)	24(18)	22(16.6
Powell's 4-vari. func.	**Δ = 1**	**2**	**3**	**4**	**5**	**Average**
(3,-1,0,1)	28(27)	23(21)	37(35)	38(35)	34(30)	32(29)
(-0.1,1,-0.1,1)	27(25)	30(28)	25(23)	31(27)	25(21)	27.6(24.8
(-0.6,1,-0.6,1)	33(32)	23(21)	27(25)	32(29)	33(29)	29.6(27.2

TEST FUNCTIONS	MINOP Average over Δ=6,8,10,12,14	MINOP Specific Δ	POWELL'S MINFA	DAVIDON'S OCOPTR	FLETCHE VMM01
Wood's function	NF(NG)	Δ = 10 NF(NG)	Δ=10 NFG	NF(NG)	NFG
(-3,-1,-3,-1)	62.8(57)	71(61)	117	72(50)	129
(-1.2,1,1.2,1)	60(51)	59(50)	73	83(61)	64
(-3,1,-3,1)	49.4(38.8)	43(35)	94	83(62)	115
(-1.2,1,-1.2,1)	118.6(106.2)	117(103)	119	138(95)	94
Total equivalent function evaluation	1304	1286	2015	1448	2010
Rosenbrock's function	Average over Δ = 1,2,3,4,5	Δ = 3	Δ = 3		
(-1.2,1)	44.2(38.8)	28(25)	37	54(43)	28
(2,-2)	35(34.4)	43(42)	39	45(41)	20
(-3.635,5.621)	58.6(55)	56(54)	39	56(50)	70
(6.39,-0.221)	53(52)	61(60)	68	35(32)	74
(1.489,-2.547)	32(30.8)	39(36)	43	54(47)	28
Total equivalent function evaluation	or 646	661	678	670	660
Box's 2-variable function	Average over Δ = 1,2,3,4,5	Δ = 3	Δ = 3		
(5,0)	22.8(19.4)	23(20)	20	16(14)	24
(0,0)	20.4(17.6)	18(16)	27	18(17)	21
(0,20)	22.6(21)	28(25)	21	19(16)	21
(2.5,10)	10(7.4)	8(5)	19	15(14)	16
(5,20)	22(16.6)	23(17)	36	26(25)	28
Total equivalent function evaluation	261.8	266	369	266	330

TEST FUNCTIONS	MINOP		POWELL'S MINFA	DAVIDON'S OCOPTR	FLETCHER VMM01
		Specific Δ			
Powell's 4-variable function	Average over Δ=1,2,3,4,5	Δ = 3	Δ = 3		
(3,-1,0,1)	32(29)	37(35)	43	31(26)	N.A.
(-0.1,1,-0.1,1)	27.6(24.8)	25(23)	41	35(28)	
(-0.6,1,-0.6,1)	29.6(27.2)	27(25)	35	30(23)	
Total equivalent function evaluation	413	421	595	404	

References

Davidon, Williams C, (1976), "New Optimization Algorithms Without Linear Searches", in Mathematical Programming 9, pp 1-30.

Powell, M. J. D., (1970a), "A Hybrid Method for Nonlinear Equations", in Numerical Methods for Nonlinear Algebraic Equations, P. Rabinowitz, ed., Gordon and Breach, London.

Powell, M. J. D., (1970b), "A Fortran Subroutines for Solving Systems of Nonlinear Algebraic Equations", in Numerical Methods for Nonlinear Algebriac Equations, P. Rabinowitz ed., Gordon and Breach, London.

Powell, M. J. D., (1970c), "A Fortran Subroutine for Unconstrained Minimization, Requiring First Derivatives of the Objective Function", A.E.R.E., Harwell, R.6469.

Powell, M. J. D., (1970d), "A New Algorithm for Unconstrained Optimization", in Nonlinear Programming, edited by J. B. Rosen, O. L. Mangasarian, K. Ritter, Academic Press, New York.

```
      SUBROUTINE MINOP(N,X,F,G,STEP,ACC,MAXFUN,IPRINT)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION X(N), G(N),XA(10), GA(10),H(10,10), GG(10,10),
     1          V(10),U(10), WA(10), WB(10), WC(10), WT(10),WS(10),
     1          Y(10), Z(10), GZ(10), GV(10), PZ(10)
C     ******************************************************************
C     * INITIATLIZE THE FOLLOWING PARAMETERS                           *
C     * NFCALL & NGCALL = NO. OF FUNCTION & GRADIENT CALLS RESP..      *
C     ******************************************************************
      NFCALL = 1
      NGCALL = 1
      IUPD = 1
      EPSSQ = ACC * ACC
      IPP = IPRINT * IPRINT
      IFTEST = 1
      DSS = STEP*STEP
      CALL CALCF (N,X,F)
      CALL CALCG (N,X,G)
      GSQ = 0
      DO 3 I = 1,N
    3 GSQ = GSQ + G(I) * G(I)
C
C     ******************************************************************
C     * PREPARE FOR THE FIRST ITERATION BY INITIALIZING MATRICES H&GG  *
C     ******************************************************************
C
      C = -DSQRT( DSS / GSQ )
      GGDIAG = 0.01 * DSQRT( GSQ) / STEP
      HDIAG = 1. / GGDIAG
      DO 6 I= 1,N
      DO 5 J = 1,N
      H(I,J) = 0.0D0
      GG(I,J) = 0.0D0
    5 CONTINUE
      H(I,I) = HDIAG
      GG(I,I) = GGDIAG
    6 Z(I) = C * G(I)
C
      PRINT 7
    7 FORMAT(1H1,5X,'THE FOLLOWING OUTPUT IS PROVIDED BY ROUTINE MINOP')
      GO TO 10
    8 GSQ = 0
      DO 9 I = 1, N
    9 GSQ = GSQ + G(I) * G(I)
   10 IF (GSQ-EPSSQ) 11, 11, 18
C
C     ******************************************************************
C     * PRINT THE FINAL VALUE OF FUN. AND GRADIENT                     *
C     ******************************************************************
C
   11 IF (IPP) 17, 17, 12
   12 PRINT 13,NFCALL,NGCALL,F
   13 FORMAT (/8X,'AFTER',I4,' FUNCTION CALLS AND',I4,' GRADIENT CALLS
     1 THE FINAL FUNCTION VALUE IS:  F= ',E20.12)
      PRINT 14,(X(I), I = 1, N)
```

PRINT -- LISTING OF INPUT DATA SET --

```
   14 FORMAT(10X,'X=',(5E20.12))
      IF (IPRINT) 17, 17, 15
   15 PRINT 16, (G(I), I = 1, N)
   16 FORMAT (10X,'G=',(5E20.12))
   17 RETURN
C
C     ******************************************************************
C     * TEST WHETHER MAXFUN CALLS OF CALCFG HAVE BEEN MADE             *
C     ******************************************************************
C
   18 IF (NFCALL-MAXFUN) 21, 19,19
   19 PRINT 20, NFCALL
   20 FORMAT(18X,'MINOP HAS EXHAUSTED',I5,'CALLS OF CALCFG')
      GO TO 11
C     * PRINT THE CURRENT BEST VALUE OF F, G, X, ETC.                  *
   21 IPTEST = IPTEST - IABS(IPRINT)
      IF (IPTEST) 22,22,26
   22 IPTEST = IPP
      NF = NFCALL - 1
      PRINT 23, NF, F
   23 FORMAT (40X,'AFTER',I4,' ITERATIONS',15X,'***** F= ',E20.12)
      BOUND = DSQRT (DSS)
      WRITE(6,24 ) BOUND, THETA
   24 FORMAT(' STEP-BOUND=',F20.12,9X,'LATEST VALUE OF THETA=',E20.12)
      PRINT 14, (X(I), I = 1, N)
      IF (IPRINT) 26, 26, 25
   25 PRINT 16, (G(I), I = 1, N)
C
C     ******************************************************************
C     * CALCUALTE NEWTON POINT                                         *
C     ******************************************************************
   26 WASQ = 0.0D0
      DO 28 I = 1,N
      WA(I) = 0.0D0
      DO 27 J= 1,N
   27 WA(I) = WA(I) - H(I,J) * G(J)
   28 WASQ = WASQ + WA(I) * WA(I)
      IF ( WASQ .LE. DSS) GO TO 39
      GGGG = 0
      HGGG = 0.0D0
      DO 30 I = 1, N
      WB(I) = 0.0D0
      DO 29 J = 1,N
   29 WB(I) = WB(I) + GG(I,J) * G(J)
      HGGG = HGGG - WA(I) * G(I)
   30 GGGG = GGGG + WB(I) * G(I)
      D = HGGG * GGGG
      D = GSQ * GSQ / D
      D = 0.2 + 0.8 * D
      IF ( D * D * WASQ .GT. DSS ) GO TO 32
      D = DSQRT ( DSS / WASQ )
      DO 31 I = 1,N
   31 WA(I) = D * WA(I)
      GO TO 41
   32 IF ( GGGG * GGGG * DSS .GT. GSQ ** 3 ) GO TO 34
```

```
C
C     ******************************************************************
C     * THE CAUCHY POINT IS OUTSIDE CIRCLE, SO PICK THE POINT ON THE
C     * BOUNDARY
C     ******************************************************************
      C = -DSQRT ( DSS / GSQ)
      DO 33 I = 1,N
   33 WA(I) = C * G(I)
      GO TO 41
C
C     ******************************************************************
C     * THE CAUCHY POINT IS INSIDE CIRCLE, USE DOG LEG STEP.
C     ******************************************************************
   34 DO 35 I = 1,N
   35 WA(I) = D * WA(I)
      CA = 0.0
      CB = 0
      C = -GSQ / GGGG
C     ******************************************************************
C     * SET THE STEEPEST DESCENT CORRECTION IN WB AND THE DIFFERENCE
C     * BETWEEN WA AND WB IN WC.
C     ******************************************************************
      DO 36 I = 1, N
      WB(I) = C * G(I)
      WC(I)= WA(I) - WB(I)
      CA = CA + WB(I) * WC(I)
   36 CB = CB + WC(I) * WC(I)
C
C     * FIND CORRECTION VECTOR AT THE INTERSECTION OF WA-WB AND CIRCL
      C = DSS -C * C * GSQ
      THETA = C / ( CA + DSQRT( CA * CA + C * CB))


      DO 38 I = 1, N
   38 WA(I) = WB(I) + THETA * WC(I)
      GO TO 41
   39 WRITE(6,40)  NFCALL
   40 FORMAT(' GENERALIZED NEWTON STEP TAKEN AT THE',I4,'TH ITERATION
C
   41 DO 42 I = 1,N
   42 XA(I) = X(I) + WA(I)
      NFCALL = NFCALL + 1
      CALL CALCF(N,XA,FA)
      DSQ = 0.0D0
      DO 43  I = 1,N
   43 DSQ = DSQ + WA(I) * WA(I)
C     *******************************************************************
C     * TEST IF FUNCTION VALUE IS DECREASED
C     *******************************************************************
      IF (FA .LT. F) GO TO 45
      WRITE (6,44)  NFCALL
   44 FORMAT(' FUNCTION INCREASED ON',I4,' TH ITERATION,NO UPDATE IS
     1E, STEP-BOUND UPDATED AND TRY AGAIN')
      DSS = 0.25 * DSQ
      GO TO 18
```

```
C
C      * CALCULATE SOME NUMBERS FOR REVISING STEP BOUND
    45 NGCALL = NGCALL + 1
       CALL CALCG(N,XA,GA)
       DG = 0.0
       DGA = 0.0
       DGGD = 0.0
       WBSQ = 0.0
       DO 47 I = 1, N
       WC(I) = 0.0D0
       DO 47 J = 1,N
    47 WC(I) = WC(I) + GG(I,J) * WA(J)
C
C      * SET  (Y-GS) IN WB
       DO 48 I = 1,N
       U(I) = WC(I)
       WB(I) = GA(I) -G(I) - WC(I)
       DG = DG + G(I) * WA(I)
       DGA = DGA + GA(I) * WA(I)
       DGGD = DGGD + WC(I) * WA(I)
    48 WBSQ = WBSQ + WB(I) *WB(I)
C      ******************************************************************
C      * TEST WHETHER TO DECREASE THE STEP-BOUND
C      ******************************************************************
       IF (FA-F-0.1 * DG-0.05 * DGGD) 51, 51, 50
    50 DSS = 0.25 * DSQ
       GO TO 54
C      ******************************************************************
C      * TEST WHETHER TO INCREASE THE STEP-BOUND
C      ******************************************************************
    51 DSS = DSQ
       IF (WBSQ - 0.25 * GSQ) 53, 53, 52
    52 IF (DG -DGA -DGA) 54, 53, 53
    53 DSS = 4.0 * DSQ
C      * SET THE DIFFERENCE BETWEN GRADIENTS IN WC
    54 DO 55 I = 1,N
    55 WC(I) = GA(I) - G(I)
       F = FA
       B0 = 0.0
       DO 56 I = 1,N
       X(I) = XA(I)
       G(I) = GA(I)
    56 B0 = B0 + WC(I) * WA(I)
C      ******************************************************************
C      * TEST WHETHER WE NEED TO UPDATE H AND G
C      ******************************************************************
       IF (B0 .GE. 1.0D-30) GO TO 58
       WRITE(6,57)  NFCALL
    57 FORMAT('FUNCTION DECREASED ON',I4,'TH ITERATION, NO UPDATE IS MAD
      1 ,SINCE Y*Z IS NEG. OR TOO SMALL')
       GO TO 8
    58 IF (IUPD .EQ. 1) GO TO 63
       IF (FLAG .EQ. 0.0) GO TO 60
       DO  59 I = 1, N
    59 Z(I) = WA(I)
```

```
      GO TO 63
   60 C2=0.0
      C3 = 0.0
      SUM = 0.0
      DO 61 I = 1, N
      C2 = C2 + WS(I) * WA(I)
      C3 = C3 + WT(I) * WA(I)
   61 SUM = SUM +Y(I) * WA(I)
      DO 62 I = 1, N
   62 Z(I) = C1 *(C2 * Z(I) + C3 * V(I)) - (SUM/ZWC) * Z(I)
   63 IUPD = IUPD+1
      DO 64 I = 1, N
      GV(I) = U(I) - WC(I)
      V(I) = WA(I)
      GZ(I) = 0.0
      DO 64 J = 1, N
      GZ(I) = GZ(I) + GG(I,J) * Z(J)
   64 V(I) =V(I) - H(I,J) * WC(J)
      ZGV = 0.0
      VGV = 0.0
      ZWC = 0.0
      ZGZ = 0.0
      VWC = 0.0
      DO 65 I = 1, N
      ZWC = ZWC + Z(I) * WC(I)
      ZGZ = ZGZ + Z(I) * GZ(I)
      VWC = VWC + V(I) * WC(I)
      ZGV = ZGV + Z(I) * GV(I)
      VGV = VGV + V(I) * GV(I)
   65 CONTINUE
      C1 = 1. / (ZGZ * VGV - ZGV * ZGV)
      C2 = ZWC * VGV - VWC * ZGV
      C3 = -ZWC * ZGV + VWC * ZGZ
      VSQ =0.0
      ZV  =0.0
      A = 0.0
      E = 0.0
      DO 66 I = 1,N
      Y(I) = C1 * (C2 * GZ(I) + C3 * GV(I))
      WS(I) = VGV * GZ(I) - ZGV * GV(I)
      WT(I) = -ZGV * GZ(I) + ZGZ * GV(I)
      ZV  =  ZV + Z(I) * V(I)
      VSQ = VSQ + V(I) * V(I)
      A = A + WS(I) * WA(I)
   66 E = E + WT(I) * WA(I)
      B = 0.0
      DO 67 I = 1,N
      PZ(I) = C1 * ( A * Z(I) + E * V(I))
   67 B = B +PZ(I) * Y(I)
      IF (B .GE. 1.0D-30) GO TO 70
      B = B0
      WRITE (6,68)
   68 FORMAT(15X,'IMAGE PRODUCT NEG.OR TOO SMALL,ORIGINAL STEP USED')
      DO 69 I = 1, N
      PZ(I) = WA(I)
```

VI

```
   69 GA(I) = WC(I)
      GO TO 72
   70 DO 71 I = 1, N
   71 GA(I) = Y(I)
   72 CC = 0.0
      A  = 0.0
      DO 74 I = 1, N
      WB(I) = 0.0
      GZ(I) = 0.0
      DO  73 J = 1, N
      GZ(I) = GZ(I) + GG(I,J) *PZ(J)
   73 WB(I) = WB(I) + H(I,J) * GA(J)
      A = A + GA(I) * WB(I)
   74 CC = CC + PZ(I) * GZ(I)
C     ***************************************************************
C     *  OPTIMAL CONDITIONING
C     ***************************************************************
      IF ( B .LE. 2.0D0 * A * CC / (A + CC)) GO TO 75
      PHI = B / (B - A)
      GO TO 76
   75 PHI = B * (CC - B)/(A * CC - B * B)
   76 USQ = 0.0
      UV  = 0.0
C     ***************************************************************
C     * TEST WHETHER Z & V ARE LINEARLY INDEPENDENT
C     ***************************************************************
      DO 77 I = 1, N
      U(I) = Z(I) - ( ZV / VSQ ) * V(I)
      USQ = USQ + U(I) * U(I)
   77 UV  = UV  + U(I) * V(I)
      FLAG = 0.0
      IF (1.0D6 * UV * UV  .GE. VSQ * USQ ) FLAG =1.0
C     ***************************************************************
C     *  IF FLAG = 1 THEN Z & V  ARE LINEARLY DEPENDENT
C     ***************************************************************
      SIGMA = 1.+ PHI * (A * CC/ (B * B) - 1.)
      IF ( SIGMA .LT. 1.0D-30) WRITE (6,78)
   78 FORMAT(20X,' G IS ALMOST SINGULAR')
      DO 79 I = 1, N
      WA(I) = (1./B) * PZ(I) - (1./A) * WB(I)
      WC(I) = (1./B) * GZ(I) -CC/(B * B) * GA(I)
   79 U(I) = GA(I)- GZ(I)
C     ***************************************************************
C     *  START TO UPDATE MATRICES H & GG
C     ***************************************************************
      DO 80 I = 1, N
      DO 80 J = 1, N
      GG(I,J) = GG(I,J) + (1./B) * (U(I) * GA(J) + GA(I) * U(J))
     1           -(( B- CC ) / (B * B)) * GA(I) * GA(J)
     1           -(PHI* A / SIGMA) * WC(I) * WC(J)
      H(I,J) = H(I,J) - (1./A) * WB(I) * WB(J) + (1./B) * PZ(I) * PZ(J
     1           + PHI * A * WA(I) * WA(J)
   80 CONTINUE
      GO TO 8
      END
```