Research Journal of Mathematics and Statistics 4(4): 84-88, 2012
ISSN: 2040-7505

Submitted: December 06, 2011 Accepted: June 29, 2012 Published: December 25, 2012

Improvements on Flow Efficiency and Further Properties of Fletcher-Powell Conjugate Gradient Algorithm

[1]S.O. Abdul-Kareem, [2]Y.A. Yahaya, [3]M.O. Ibrahim and [4]A. S. Mohammed
[1]Department of Mathematics and Statistics, Kara State Polytechnic, Ilorin, Nigeria
[2]Department of Mathematics and Statistics, Federal University of Technology, Mina, Nigeria
[3]Department of Mathematical Science, USANI Danfodiyo University, Soot, Nigeria

Abstract: An investigation was carried on the flow efficiency of Davison (Fletcher and Powell, 1963) procedure for minimization. Given some functions, the program implementing the procedure was found either prone to over flow problem or flowing for a while. With aim of overcoming these challenges, this study proposed a modification of some aspects of the procedure. New subroutines were introduced to perfect the original procedure. On implementing the new procedure, the program implementing it exhibited better quality of flow and further computational properties of the original procedure.

Keywords: Convergence routing, directed flow track, quadratic approximation, search and step size selection

INTRODUCTION

In the neighborhood of non degenerate minimum point z, a nonlinear function f(z) appears as an approximate quadratics. This appearance motivates having to seek for effective algorithms for minimizing quadratics. Walsh (1975) and Hestenese (1980) modified such algorithms to minimize non quadratic functions.
Having to minimize {f(z): z ε R^n), the tendency is to minimize:

$$f(z) = f_0 + \langle g, z\rangle + \tfrac{1}{2}\langle z, Hz\rangle \quad (1)$$

$g(z) = \partial f(z) / \partial z$, $H(z) = \partial g(z) / \partial z$, H ε $R^n \times R^n$ is positive definite, z, g ε R^n and f_0, f(z) ε R. Hestenese and Stiefel (1952) proposed Algorithm 1 (Appendix) for implementing minimization process. Later, Fletcher and Reeves (1964) and Polak and Ribiere (1969) introduced Algorithms 2 and 3 (Appendix) for computationally implementing Algorithm 1. Algorithms 2 and 3 yield identical results when minimizing quadratic functions. Both exhibit very slow convergences with the step size λ selected. Polak (1971) improved the convergence rate, choosing explicit form $\lambda = \langle g, g\rangle / \langle h, \langle H, h\rangle\rangle$ and Algorithm 4 (Appendix). Fletcher and Powell, 1963 provided Algorithm 6 (Appendix) for computationally implementing the minimizing algorithm. Noting that Algorithm 5 (Appendix) is incorporated in Algorithm 4 and both the search function a: T→T and stop rule c: T→R were utilized, this study is set out to perfect computational flow of the original (Fletcher and Powell, 1963) procedure. The primary objective is to modify the original search function and stop rule in such a manner that various options of computational implementation shall be provided. On this provision, the options were tested on some functions. The results revealed that desirable computational flow was achieved. New properties of the modified procedure were established with proofs.

MATHDOLOGY

Program flow improvements: Choose the search relation b: G→G, G = {g} and stop rule d: N→G U N, d (k) = g_k, k ε N = 1, 2, 3. Then consider making use of the following modifying procedure in subsequent needs for obtaining solutions to numerical examples.

Modification procedure:

Step 1: Select option. Opt = 1, 2, 3. If opt = 1 go to step 9
Step 2: Compute z_i ε T from Algorithm 5. If opt = 3 go to step 7
Step 3: Compute $b(g_i)$. If opt = 2 go to step 8
Step 4: Set $g_{i+1} = b(g_i)$
Step 5: Set $g_i = g_{ik}$. Reinitialize g after every (predetermined) k iterations
Step 6: Set i = i+1. Go to step 1
Step 7: z = z/ | z |. Go to step 2
Step 8: g = g/ | g |. Go to step 3 of Algorithm 6
Step 9: y = y/ | y |. Go to step 10 of Algorithm 6

Example 1: Minimize $f(z) = 100\,(z_{12} - z_2)^2 + (1 - z_1)^2$, given initial values z = (-1.2, 1).

Corresponding Author: S.O. Abdul-Kareem, Department of Mathematics and Statistics, Kara State Polytechnic, Ilorin, Nigeria

Table 1: The values generated by Fletcher and Powell (1963)

k	z_1	z_2	f(z)
1	-1.20000E+00	1.00000E+00	2.43000E+01
2	-1.01284E+00	1.07639E+00	4.30709E+00
3	-8.73203E-01	7.41217E-01	3.55411E+00
4	-8.15501E-01	6.24325E-01	3.46184E+00
5	-6.72526E-01	3.88722E-01	3.20145E+00
6	-7.12159E-01	4.94380E-01	2.94785E+00
7	1.05488E+00	1.13391E+00	4.77510E+00
8	1.06641E+00	1.13774E+00	4.43687E-03
9	1.06413E+00	1.13140E+00	4.20725E-03
10	1.02291E+00	1.04313E+00	1.55704E-03
11	1.02772E+00	1.05350E+00	1.49666E-03

Solution: The program implementing Algorithm 6 overflows. If option 1 of our modification is selected, step 10 of the algorithm is adjusted to read "set $y_i = g_{i+1}$, $y_i = y_i/ |y_i|$, d = 0, $z_0 = 0$. Compute $d_i = d + y_i^T\Delta z_i$, $d_2 = d + y_i^T G_{ij} y_i$, $z_i = z_i G_{ij} y_i$" and the program flows, yielding values obtained by Fletcher and Powell (1963) (Table 1). Option 2 adjusts step 9 to read "set $\delta = \delta_0 z_i$. If $|z_i| <10^{-12}$ then $\delta = \delta_0$. Compute $z_i = z_i + \delta$, $g_{i+1} = (f_1 - f_0) /\delta$. $G_{i+1} = g_{i+1}/ |g_{i+1}|$ Go to step 2" and to consequently cause the program to flow for longer time (Table 2).

Example 2: Minimize $f(z) = (z_1z_2 -1)^2 + (z_1^2 + z_2^2 - 4)$, given initial values z = (0, 1).

Solution: The implementing program overflows. Option 1 adjusts Algorithm 6 to cause the program to flow: Values were obtained (Table 3). Option 3 adjusts Algorithm 6 to cause its step 7 to read "Set i = i+1. $z_i = z_i/ |z_i|$. Compute $\Delta z_i = \alpha S_1$, $z_{i+1} = z_i + \Delta z_i$, $f_1 = f_0$. If $f_0>f_1$ then $z_{i+1} = z_i + (\alpha_1 - \alpha)$ S1, $\Delta z_i = \alpha_1 S_1$" and to consequently cause the program to flow and yield values (Table 4). Options 2 and 3 together yield us better results (Table 5).

Example 3: Minimize $f(z) = 3z_1^4 - 2z_1^2z_2^2 + 3z_2^4$, given initial values z = (1, 0.1).

Table 2: The values generated by Abdul-Kareem (1993)

k	z_1	z_2	f(z)
1	-1.2000000000E+00	1.0000000000E+00	2.4300000000E+01
2	-1.2176337853E+00	1.0105819912E+00	2.7201024007E+01
3	-1.2334219272E+00	1.0184764639E+00	3.0274306224E+01
4	6.8522004820E-01	5.9155877958E-02	1.6939492348E+01
5	1.0214532759E-01	-2.3238176580E-01	6.7020764995E+00
6	-3.3900972059E-01	-1.1804423023E-02	3.3990473609E+00
7	-3.4913921401E-01	-1.5726309459E-01	9.6132789422E+00
8	2.0997695413E-01	1.2229450679E-01	1.2357340248E+00
9	-3.0624508388E-01	1.8581999191E-01	6.9779861961E+00
10	3.0954087063E-01	1.7222699965E-01	1.0573543400E+00
11	3.0964105436E-01	1.7222710679E-01	1.0579510352E+00
12	6.0401129342E-01	4.3683696318E-01	6.7531247761E-01
13	6.8631573227E-01	4.7738931743E-01	1.0324202584E-01
14	6.9782881569E-01	4.8374599397E-01	9.2343660667E-02
15	6.9800017054E-01	4.8383180623E-01	9.2341226650E-02
16	6.9801704015E-01	4.8384037586E-01	9.2341164367E-02

Table 3: The generated values, making use of option 2

k	z_1	z_2	f(z)
1	0.0000000000E+00	1.0000000000E+00	1.0000000000E+01

Table 4: The generated values, making use of option 3

k	z_1	z_2	f(z)
1	0.0000000000E+00	1.0000000000E+00	1.0000000000E+01
2	-2.0522944001E+00	-2.6077507493E-02	9.4102231193E-01

Table 5: The generated values, making use of options 2 and 3 together

K	z_1	z_2	f(z)
1	0.0000000000E+00	1.0000000000E+00	1.0000000000E+01
2	5.9555923790E-01	1.7977796189E+00	1.7581120037E-01
3	6.7279863470E-01	1.8364008174E+00	8.6089334189E-02
4	6.7586550213E-01	1.8379357911E+00	8.5949944231E-02
5	6.7601548620E-01	1.8380122431E+00	8.5948970036E-02
6	6.7611453512E-01	1.8380632676E+00	8.5948248405E-02

Table 6: The generated values, making use of options 2

k	z_1	z_2	f(z)
1	1.0000000000E+00	1.0000000000E-01	2.9803000000E+00
2	8.9873959993E+01	-4.0630690034E-01	1.7723669852E+00
3	1.1193988698E-01	-1.2907000806E-02	4.6695148513E-04
4	1.9561333540E-02	3.3282318979E-02	3.2726128082E-06

Table 7: The generated values, making use of options 1 and 2 together

k	z_1	z_2	f(z)
1	1.0000000000E+00	1.0000000000E-01	2.9803000000E+00
2	2.9797863868E-02	-3.8410146807E-01	6.5720469316E-02
3	-2.1549766646E-01	-2.6245407310E-01	1.4306388961E-02
4	2.8164045327E-02	-1.4062337191E-01	1.1436599708E-03
5	-83158838384E-02	-8.4962056601E-02	1.9995216817E-04
6	1.0895306490E-02	-3.7935027547E-02	5.9133536185E-06
7	1.0895367735E-02	-3.7935014787E-02	5.9133425987E-06
8	-1.1302962588E-02	-2.6835882113E-02	1.4208644666E-06
9	1.5780561177E-02	1.3294141415E-02	1.9172408102E-07
10	1.6111062322E-02	-6.2094193462E-03	4.2799519823E-09

Solution: Option 2 adjusts Algorithm 6 to cause program to yield values (Table 6). On switching to options 1 and 2, they together cause the algorithm to yield us values (Table 7).

SUMMARY OF RESULTS

We are still left with further combinations of options 1, 2, 3. We have not made use of options 1 and 3 together and options 1, 2, 3 together for improving Algorithm 6, to consequently cause the program to flow more efficiently. Unlike in Table 1 (Fig. 1), the values in Table 2 provide useful information on the presence of intermediate sub optimal values (Fig. 2). The presence of sub minimal values of the objective function is noted at iterations k = 6, 8, 10. Our modification procedure improves the flow efficiency of Algorithm 6, judging from results achieved for the three examples considered.

Revealed computational properties: The success achieved from the improvements on the flow efficiency of the Fletcher and Powell Algorithm 6 gears our interest in exploring for further hiding properties of the algorithm. While reviewing literatures, the following new results were revealed. Let A be an n × n positive definite and symmetric matrix operator. Choosing a step size λ along specified descent direction (Ibiejugba, 1980), subsequent optimal point z is computed such that:

$$z_{i+1} = z_i + \lambda_i p_i$$
$$p_i = -g_i,\ i = 0$$
$$p_i = -g_i - \langle g_i, g_i\rangle\, g_{i-1} / \langle g_{i-1}, g_{i-1}\rangle,\ i>0 \qquad (2)$$

The modifications earlier introduced to achieve better flow efficiency agree with the following new results:

Result 1: $\langle A(z_i - z_{i-1}), (z_{i+1} - z_i)\rangle = 0$

Proof: From Eq. (2), $A(z_{i+1} - z_i) = \lambda_i A p_i$ is obtained and:

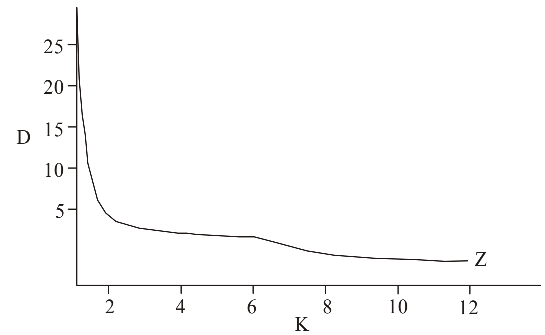

Fig. 1: Illustration of the flow pattern exhibited with Table 1

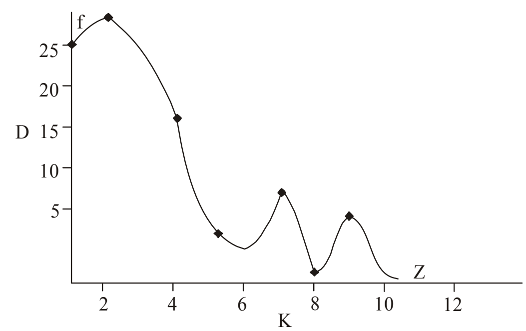

Fig. 2: Illustration of the flow pattern exhibited with Table 2

$$\langle A(z_i - z_{i-1}), (z_{i+1} - z_i)\rangle = \langle (z_i - z_{i-1}), A(z_{i+1} - z_i)\rangle$$
$$= \langle \lambda_{i-1} p_{i-1}, \lambda_i A p_i\rangle$$
$$= \lambda_{i-1} \lambda_i \langle p_{i-1}, A p_i\rangle$$
$$= 0$$

This is so because p_{i-1} and p_i are A-conjugate descent directions. The proof of this result is established.

Result 2: $(z_{i+1} - z_i)(z_i - z_{i-1}) \ldots (z_1 - z_{-0}) = 0$

Proof: For i>0, any sub product:

$$A(z_{i+1} - z_i)(z_i - z_{i-1}) = \langle A(z_i - z_{i-1}), (z_{i+1} - z_i)\rangle = 0$$

is in agreement with Result 1and the proof of this result is established.

Result 3: For j = 1, 2, 3,… $A(z_{i+j} - z_i)(z_i - z_0) = 0$

Proof: From existing results (Ibiejugba, 1980), $z_{i+j} - z_i = \Sigma_{k=1}^{i+j-1} \lambda_k p_k$ and $z_i - z_0 = \Sigma_{k=0}^{i-1} \lambda_k p_k$. Consequently:

$A(z_{i+j} - z_i)(zi - z_0) = <(z_{i+j} - z_i), (zi - z_0)>$
$= <\Sigma_{k=1}^{i+j-1} \lambda_k p_k, \Sigma_{k=0}^{i-1} \lambda_k p_k>$
$= <\Sigma_{k=1}^{i+j-1} \lambda_k p_k <Ap_k, p_{k-1}>, j = 1, 2, 3,\ldots$
$= 0$

This is so because p_k and p_{k-1} are A-conjugate directions. Hence, the proof of this result is established.

Result 4:
$\Sigma_{k=0}^{i-1} 1/[g_k, g_k] = <\Sigma_{k=0}^{i-1} g_k/<g_k, g_k>, \Sigma_{k=0}^{i-1} g_k/<g_k, g_k>>$

Proof: From the existing results (Ibiejugba, 1980), $<p_{i-1}, p_i> = <g_{i-1}, g_{i-1}> <g_i, g_i> <\Sigma_{k=0}^{i-1} g_k/[g_k, g_k], \Sigma_{k=0}^{i-1} g_k/[g_k, g_k] p_k, p_{k-1}>$
This result and Eq. (2) yield:

$<p_0, p_1> = <-g_0, -g_1 - <g_1, g_1> g_0/<g_0, g_0>>$
$= <-g_0, -g_1> + <-g_0, - <g_1, g_1> g_0/<g_0, g_o>>$
$= <g_1, g_1>$

This is so because g_0 and $g_1$are mutually conjugate. It can be similarly shown that:

$<p_1, p_2> = <-g_1, - <g_1, g_1> g_0/<g_0, g_0>, -g_2 - <g_2, g_2> g_1/<g_1, g_1> - <g_2, g_2> g_o/<g_0, g_0>$
$= <g_2, g_2> + <g_1, g_1> <g_2, g_2>/<g_0, g_0>$
$<p_2, p_3> = <-g_2, - <g_2, g_2>g_1/<g_1, g_1> - <g_2, g_2> g_0/<g_0, g_0>, -g_3 - <g_3, g_3>g_2/<g_2, g_2> - <g_3, g_3> g_1/<g_1, g_1> - <g_3, g_3> g_0/<g_0, g_0>$
$= <g_3, g_3> + <g_2, g_2> <g_3, g_3> / <g_1, g_1> + <g_2, g_2> <g_3, g_3> / <g_0, g_0>$

This order continues. It then generally followed that:

$<p_{i-1}, p_i> <g_{i-1}, g_{i-1}> / <g_{i-3}, g_{i-3}> + \ldots <g_i, g_i> <g_{i-1}, g_{i-1}> / <g_0, g_0>$
$= <g_i, g_i> <g_{i-1}, g_{i-1}> <1/<g_{i-1}, g_{i-1}> + 1/<g_{i-2}, g_{i-2}> + \ldots + 1/<g_0, g_0>$
$= <g_i, g_i> <g_{i-1}, g_{i-1}> \Sigma_{k=0}^{i-1} 1/<g_k, g_k>$

Noting the regular pattern of the right hand sides of our outcomes, we conclude that:

$\Sigma_{k=0}^{i-1} 1/<g_o, g_k> = <\Sigma_{k=0}^{i-1} g_k/<g_k, g_k>, \Sigma_{k=0}^{i} g_k/<g_k, g_k>>.$

Thus the proof of this result is established by inspection.

Result 5:
$<p_{i-1}, p_i> / <p_{i-2}, p_{i-1}> \geq \Sigma_{k=0}^{i-1} <g_i, g_i> / <g_k, g_k>, i\geq 2$

Proof: From the foregoing result, it follows that:

$<P_{i-2}, p_{i-1}> = <g_{i-2}, g_{i-2}> <g_{i-1}, g_{i-1}> \Sigma_{k=0}^{i-2} 1/<g_k, g_k>$

Dividing this and the foregoing result, we obtain:

$<p_{i-1}, p_i> / <p_{i-2}, p_{i-1}> = <g_i, g_i> \Sigma_{k=0}^{i-1} 1/<g_k, g_k> / (<g_{i-2}, g_{i-2}> \Sigma_{k=2}^{i-2} 1/<g_k, g_k>)$

For i = 2:

$<p_1, p_2> / <p_0, p_1> = [<g_2, g_2> / <g_0, g_0>] [(1/<g_0, g_0> +1/<g_1, g_1>) / (1/<g_0, g_0>)]$
$= <g_2, g_2> [1+ <g_0, g_0> / <g_1, g_1>] / <g_0, g_0>$
$= <g_2, g_2> / <g_0, g_0> + <g_2, g_2> / (<g_1, g_1>$

For i = 3:

$<p_2, p_3> / <p_1, p_2> = [<g_3, g_3> / <g_1, g_1>] [(<g_0, g_0> +1/<g_1, g_1> +1/<g_2, g_2>) / (1/<g_0, g_0> + 1/<g_1, g_1>)]$
$= [<g_3, g_3> / <g_0, g_0> + <g_3, g_3> / <g_1, g_1> + <g_3, g_3> / <g_2, g_2>] / [1+ <g_1, g_1> / <g_0, g_0>]$
$\geq [<g_3, g_3> <g_3, g_3> <g_3, g_3>] / [<g_0, g_0> <g_1, g_1> <g_2, g_2>]$

This order continues. It then generally follows that, for i = j:

$<p_{j-1}, p_j> / <p_{j-2}, p_{j-1}> \geq <g_j, g_j> \Sigma_{k=0}^{j-1} 1/<g_k, g_k>$

Applying the principle of mathematical induction, this result is established.

CONCLUSION

If achieving the maximal values of the objective functions had been our primary target, then we would have adjusted step 3 of Algorithm 6 and caused it to read " Step 3: Set $S_0 = 0$ and $S_{i+1} = S_i + G_{ij}g_i$". Agreeing that constrained optimization problems can be reduced to unconstrained type (Ibiejugba *et al*., 1986; Abdul-Kareem, 1993), our results hold for all cases of optimization. We must have previously not been very strict on routing our search for optimal cost function along descent direction. Now that we adhere only to searching along descent direction we begin to achieve tremendous success.

APPENDIX

Algorithm 1: (Hestenese and Stiefel, 1952):
Step 0: Select z_0 ε R^n. Set i = 0.
Step 1: Compute $\delta f(z)/\delta z = \nabla f(z)$.
Step 2: If $\nabla f(z) = 0$ stop, else, compute h_i ε $F(z_i)$ and go to step 3.
Step 3: Compute $\lambda_i>0$ such that $f(z + \lambda_i h_i) = \min \{f(z_i + \lambda_i h_i), \lambda_i>0\}$.
Step 4: Set $z_{i+1} = z_i + \lambda_i h_i$, i = i + 1 and go to step 1.
Algorithm 2: (Fletcher-Reeves, 1964):
Step 0: Select z_0 ε R^n. If $\nabla f(z) = 0$ stop. Else, go to step 1.
Step 1: Set i = 0, $g_0 = h_0 = -\nabla f(z_0)$.
Step 2: Compute $\lambda_i>0$ such that $f(z_0 + \lambda_i h_i) = \min \{f(z_i + \lambda_i h_i), \lambda_i>0\}$.
Step 3: Set $z_{i+1} = z_i + \lambda_i h_i$,
Step 4: Compute $\nabla f(z_{i+1})$
Step 5: If $\nabla f(z_{I+1}) = 0$ stop, else set:
$g_{i+1} = -\nabla f(z_{I+1})$, $h_{i+1} = g_{i+1} + \lambda_i h_i$ with $\lambda_i = <g_{i+1}, g_{i+1}> / <g_i, g_i>$ Set i = i+1 and goto step 2

Algorithm 3: (Polak and Ribiere, 1969):
Step 0: Select z_0 ε R^n. If ▼f (z) = 0 stop. Else, go to step 1.
Step 1: Set i = 0, $g_0 = h_0$ = -▼f (z_0).
Step 2: Compute λ_i>0 such that f($z_0 + \lambda_i h_i$) = min {f ($z_i + \lambda_i h_i$), λ_i>0}.
Step 3: Set $z_{i+1} = z_i + \lambda_i h_i$,
Step 4: Compute ▼f (z_{i+1})
Step 5: If ▼f(z_{I+1}) = 0 stop, else set g_{i+1} = -▼f(z_{I+1}), $h_{i+1} = g_{i+1} + \lambda_i h_i$ with λ_i = <g_{i+1}, g_{i+1}> / <g_i, g_i>
Set i = i+1 and goto step 2

Algorithm 4: (Polak, 1971):
Step 0: Select ε_0>0, α>0, βε(0, 1) and δ>0. Select z_0εR^n and set i = 0
Step 1: Set ε = ε_0
Step 2: Compute ▼f (z_{i+1})
Step 3: If ▼f (z_{I+1}) = 0 stop, else compute h_i (consistently as in Algorithm 2 or 3)
Step 4: Select θ (Γ) = f (z + Γh_i) and use the following Algorithm 5 with the current value of ε to compute Γ
Step 5: If θ (Γ) ≤αε set λ_i = Γ and goto Step 6, else set ε = βε and got Step 4
Step 6: Set $z_{i+1} = z_i + \lambda_i h_i$, i = i+1 and goto step 3

Algorithm 5:
Step 0: Compute z_0εT, TcR^n
Step 1: Set i = 0
Step 2: Compute a (z_i)
Step 3: Set z_{I+1} = a (z_i)
Step 4: If c (z_{i+1}) ≥x (z_i) stop, else set i = i+1 and goto step 2

Algorithm 6: (Fletcher and Powell, 1963)
Step 0: Set i = 0. Select $\delta_0 = 10^{-6}$ and x_0εR^n. Compute f (z), f_i = f (z_i)
Step 1: Set δ = $\delta_0 z_i$. If $|z_i|$ <10^{-12} then δ = δ_0. Set $z_{i+1} = z_i$ + δ. Compute f_{i+1} and g = (f_i -f_0)/δ]
Step 2: Set G_{ij} = 0, i # 0, G_{ij} = 1, i = j
Step 3: Set S_0= 0 and $S_{i+1} = S_i - G_{ij} g_i$
Step 4: Set α_1 = 1. Compute $z_{1i} = z_i + \alpha_1 S_1$and set $f_{1i} = f_1$. If f_1<f_0 goto Step 5, else set $\alpha_1 = 0.5\alpha_1$
Step 5: Set $\alpha_2 = 2\alpha_1$. Compute $z_{2i} = z_i + \alpha_2 S_i$ and set $f_{2i} = f_2$. If f_2>f_1 goto Step 6, else set $\alpha_2 = \alpha_1$ and $f_2 = f_1$
Step 6: Compute α* = ($\alpha_1^2 - \alpha_2^2$) f_0 + $\alpha_2^2 f_1$ - $\alpha_1^2 f_2$. α = 0.5α*/ [(α_1 - α_2) f_0 + $\alpha_2 f_1$ - $\alpha_1 f_2$]
Step 7: Set i = i+1. Compute $\Delta z_i = \alpha S_i$ and set $z_{i+1} = z_i + \Delta z_i$, $f_1 = f_0$. If f_0>$f_1$then set $z_i = z_i$ + (α_1 - α) Si and $\Delta z_i = \alpha_1 S_1$
Step 8: Set k = 1. $g_i = g_{ik}$. If k = 5 goto Step 1
Step 9: Set δ = $\delta_0 z_i$. If $|z_i|$ <10^{-12} then δ = δ_0. Compute $z_i = z_i$ + δ, g_{i+1} = (f_i -f_0) /δ and go to step 2
Step 10: Set $Y_i = g_{i+1}$, d = 0 and z_0 = 0. Compute d_1 = d + $Y_i^T \Delta z_i$, d_2 = d + $Y_i^T G_{ij} Y_i$ and $z_i = z_i G_{ij} Y_j$
Step 11: Compute $G_{i+1\ J+1} = G_{ij} + \Delta z_i^T \Delta z_i$, /$d_1$ - $z_i^T z_i$, /d_2 and goto Step 3

REFERENCES

Abdul-Kareem, S.O., 1993. Computational improvements on Fletcher-Powell conjugate gradient method. M.Sc. Thesis, University of Ilorin, Nigeria.

Fletcher, R. and M.J.D. Powell, 1963. A rapidly convergent descent method for minimization. Comp. Jour., 6: 149-154.

Fletcher, R. and C.M. Reeves, 1964. Function minimization by conjugate gradients. Comp. J., 7: 149-154.

Hestenese, M.R. and E. Stiefel, 1952. Methods of conjugate gradients for solving linear systems. J. Res. Nat. Bur Standards, 49: 409-436.

Hestenese, M.R., 1980. Conjugate Direction Methods in Optimization. Springer-Verlas, New York.

Ibiejugba, M.A., 1980. Computing methods in optimal control theory. Ph.D. Thesis, University of Leeds.

Ibiejugba, M.A., I. Orisamolu and J.E. Rubio, 1986. A penalty optimization technique for class of regular optima control problems. Abacus: J. Math. Assoc. Nigeria, 17(1).

Polak, E. and G. Ribiere, 1969. Note on methods of convergencie conjugates. Rev. Fr. Inr. Rech. Oper. 16: 35-43.

Polak, E., 1971. Computational Methods in Optimization: A Unified Approach. Academic Press, London.

Walsh, G.R., 1975. Methods of Optimization. John Wiley and Sons Ltd., New York.