International Journal of Pure and Applied Mathematics

Volume 18 No. 3 2005, 331-341

A QUADRATIC TERMINATION QUASI-NEWTON ALGORITHM WITHOUT EXACT LINEAR SEARCHES

Issam A.R. Moghrabi

Department of Computer Science
Faculty of Science
Beirut Arab University
P.O. Box 11-5020, Beirut, LEBANON
e-mail: imoghrabi@bau.edu.lb

Abstract: A new Quasi-Newton for unconstrained optimization that obtains the minimum for a quadratic function in a finite number of iterations is derived in this paper. The number of steps to obtain the minimum does not exceed the dimension of the problem being solved. The new algorithm competes favorably with Dixon's method [6], as our numerical results indicate. The algorithm presented here does not require any extra storage as in the case of Dixon's technique where an additional $O(n)$ storage is retained and updated at each iteration.

AMS Subject Classification: 65K10
Key Words: nonlinear programming, unconstrained optimization, quasi-Newton methods

1. Introduction

Unconstrained Optimization deals with minimizing a certain objective function with no constraints on the solution. This type of problems is of the form:

$$\text{minimize } f(x), \text{ where } f : R^n \to R),\ x \in R^n\,.$$

Received: October 18, 2004

The solution can be found by using a class of methods known as quasi-Newton method for unconstrained optimization. Quasi-Newton methods require only the function and its first partial derivatives (gradient) to be available. The Hessian is not required to be available or even coded. However, an approximating matrix to the Hessian is used and updated throughout the iterations to incorporate the changes in the function and its gradient.

Given B_k, current approximation to the Hessian, we need to find a new approximating matrix B_{k+1} to the new Hessian, evaluated at the newly computed iterate x_{k+1}. To determine B_{k+1} we may use the Taylor's series approximation of first order to the gradient about the iterate $\underline{x}_{k+1}$ to obtain a relation of the form (referred to as the secant equation):

$$B_{k+1}\underline{s}_k = \underline{y}_k\,,$$

where

$$\underline{s}_k = \underline{x}_{k+1} - \underline{x}_k,$$

and

$$\underline{y}_k = \underline{g}_{k+1} - \underline{g}_k.$$

Updating formulae used to find the new Hessian approximating matrix B_{k+1} using information from both the old approximation B_k and the vectors s_k and y_k, have the following form: $B_{k+1} = B_k + C_k$, where C_k is a correction matrix. Alternatively, it is preferable to use $H_{k+1} = H_k + D_k$, where D_k is a correction matrix and $H_{k+1} = B_{k+1}^{-1}$.

One particular rank-two formula is the well known BFGS formula. This formula is given by:

$$B_{k+1}^{BFGS} = B_k + \frac{\underline{y}_k\ \underline{y}_k^T}{\underline{y}_k^T\ \underline{s}_k} - \frac{B_k\ \underline{s}_k\ \underline{s}_k^T\ B_k}{\underline{s}_k^T\ B_k\ \underline{s}_k}$$

$$H_{k+1}^{BFGS} = H_k + \left[1 + \frac{\underline{y}_k^T\ H_k\ \underline{y}_k}{\underline{s}_k^T\ \underline{y}_k}\right]\frac{\underline{s}_k\ \underline{s}_k^T}{\underline{s}_k^T\ \underline{y}_k} - \frac{\underline{s}_k\ \underline{y}_k^T\ H_k\ + H_k\ \underline{y}_k\ \underline{s}_k^T}{\underline{s}_k^T\ \underline{y}_k}. \quad (1)$$

Numerical results indicate that this formula is superior to other updating formulae especially when inaccurate (non-exact) line searches are used. BFGS is considered to be a standard updating formula [4].

In this work, we introduce a new algorithm that terminates in m ($\leq n$) iterations in the case of inexact line search (which is normally required to obtain such finite termination). The method presented here is similar to Dixon's technique; however, in our case no extra storage is expended. Dixon's method

maintains and updates, at each iteration, a vector that accumulates certain quantity needed to achieve finite termination on quadratic functions. This vector is reset after n iterations to the null vector. Next, we briefly discuss Dixon's algorithm and use it to motivate our method. In Section 3, the derivation of the new algorithm is made. We finally present our numerical results that compare Dixon's algorithm to ours.

2. Dixon's Algorithm

Dixon's algorithm is derived by considering the effect of non-accurate line searches on a quadratic function. A hat indicates an exact quantity that corresponds to an exact line search. Initially, x_0, H_0, f and g_0 are all given and hence the first direction is computed using $p_0 = -H_0 g_0$. Hence, at this iteration $s_0 = \theta \widehat{s}_0$.

As the function is quadratic, it follows that

$$y_0 = As_0 = \theta A\widehat{s}_0 = \theta \widehat{y}_0.$$

Also,

$$g_1 = g_0 + y_0 = g_0 + \theta \widehat{y}_0 = \widehat{g}_1 + (\theta - 1)\widehat{y}_0.$$

Thus,

$$H_1 g_1 = \widehat{H}_1 \widehat{g}_1 + (\theta - 1)\widehat{H}_1 \widehat{y}_0,$$

from which it follows that

$$p_1 = \widehat{p}_1.$$

If we now take an arbitrary step along p_1, we obtain

$$s_1 = \theta_1 \widehat{s}_1 \,,$$

so that

$$y_1 = \theta_1 \widehat{y}_1 \quad \text{and} \quad H_2 = \widehat{H}_2.$$

By induction, let us assume that $H_k = \widehat{H}_k$ and

$$p_j = \theta_j \widehat{p}_j, \text{ for all } j < k, \quad \text{then} \quad g_k = \widehat{g}_k + \sum_{j=0}^{k-1} (\theta_j - 1)\widehat{y}_j$$

and hence

$$H_k g_k = \widehat{H}_k \widehat{g}_k + \sum_{j=0}^{k-1} (\theta_j - 1)\widehat{H}_k \widehat{y}_j.$$

It can then be easily shown that $s_k = \theta_k \widehat{s}_k$ and $H_{k+1} = \widehat{H}_{k+1}$.

Dixon then proceeds to show that

$$\theta_j = (-y_j^T s_j)/(g_j^T s_j).$$

For the non quadratic case, Dixon retains a correction vector w that starts as null and is then updated at each iteration to adjust the search direction so that it matches the one that would have been obtained using exact line search. This is given by

$$p_k = -H_k g_k + w_k \,,$$

where

$$w_{k+1} = w_k + \theta_k s_k.$$

Dixon's algorithm possesses the following properties, when applied to quadratic functions (and so will our algorithm):

a) The directions p_k are the same conjugate directions taht would have been obtained with accurate line searches.

b) The matrices H_k obtained are the same ones that would have been computed using exact line searches and the BFGS formula.

c) The algorithm terminates in at most $n + 2$ iterations.

3. The New Algorithm

Before proceeding with our derivation, we need the results presented in the following lemma (again, a hat indicates an exact quantity that corresponds to an exact line search).

Lemma. *For a quadratic function, if inexact line search is used, then the constant* $\alpha_j (j \in [0, \infty])$ *in* $\widehat{y}_j = \widehat{g}_{j+1} - \widehat{g}_j = \alpha_j y_j$ *and* $\widehat{s}_j = \widehat{x}_{j+1} - \widehat{x}_j = \alpha_j s_j$, *where* $y_j = g_{j+1} - \widehat{g}_j$, *is given by*

$$\alpha_j = -(s_j^T \widehat{g}_j / s_j^T y_j). \tag{2}$$

Also, the exact step size along some search direction $\widehat{p}_j$ *is given as*

$$\widehat{t}_j = -t_j (s_j^T \widehat{g}_j / s_j^T y_j),$$

where t_j *is a non-exact step size found by some line search mechanism.*

Proof. We proceed by induction. The base case is

$$g_1 = g_0 + y_0 = g_0 + \widehat{y}_0/\alpha_0 = \widehat{g}_1 + (1/\alpha_0 - 1)\widehat{y}_0.$$

We, recursively, have

$$g_k = g_0 + \sum_{j=0}^{k-1} y_j = \widehat{g}_k + \sum_{j=0}^{k-1} (1/\alpha_j - 1)\widehat{y}_j.$$

Now, since $\widehat{y}_j = \alpha_j y_j$, it follows that

$$\alpha_j s_j^T y_j = s_j^T \widehat{y}_j = s_j^T \widehat{g}_{j+1} - s_j^T \widehat{g}_j = -s_j^T \widehat{g}_j.$$

Thus, we have

$$\alpha_j = -s_j^T \widehat{g}_j / s_j^T y_j.$$

Now, since $\widehat{s}_j = \alpha_j s_j$, it follows that

$$\widehat{s}_j = t_j (-s_j^T \widehat{g}_j / s_j^T y_j) p_j,$$

which implies that the exact minimum along p_j is

$$\widehat{t}_j = t_j (-s_j^T \widehat{g}_j / s_j^T y_j). \tag{3}$$

Hence, the proof is complete. □

We are now able to give an outline of the proposed algorithm as follows:

1. **set** $k = 0$ (where k is the iteration counter); given $\widehat{x}_0$, $\widehat{H}_0$ and $\widehat{g}_0$;
2. $\widehat{p}_k = -\widehat{H}_k \widehat{g}_k$;
3. **find** a minimum t_k along the search direction $\widehat{p}_k$ using some inexact line search algorithm (such as, cubic interpolation [1]);
4. **compute** $x_{k+1} = \widehat{x}_k + t_k \widehat{p}_k$;
5. **compute** $s_k = t_k \widehat{p}_k$, $\widehat{s}_k = \alpha_k s_k$ (where α_k is as in (2)) and $\widehat{x}_{k+1} = \widehat{x}_k + \widehat{s}_k$;
6. **If** $\widehat{x}_{k+1}$ is the minimum, **stop else** goto 7;
7. **let** $y_k = g_{k+1} - \widehat{g}_k$;
8. **let** $\widehat{g}_{k+1} = t_k \widehat{g}_k + \widehat{t}_k y_k$;
9. **compute** $\widehat{y}_k = \widehat{g}_{k+1} - \widehat{g}_k$;
10. **update** $\widehat{H}_k$ to obtain $\widehat{H}_{k+1}$ using

$$\widehat{H}_{k+1}^{BFGS} = \widehat{H}_k + \left[1 + \frac{\underline{\widehat{y}}_k^T \ \widehat{H}_k \ \underline{\widehat{y}}_k}{\underline{\widehat{s}}_k^T \ \underline{\widehat{y}}_k}\right] \frac{\underline{\widehat{s}}_k \ \underline{\widehat{s}}_k^T}{\underline{\widehat{s}}_k^T \ \underline{\widehat{y}}_k} - \frac{\underline{\widehat{s}}_k \ \underline{\widehat{y}}_k^T \ \widehat{H}_k \ + \widehat{H}_k \ \underline{\widehat{y}}_k \ \underline{\widehat{s}}_k^T}{\underline{\widehat{s}}_k^T \ \underline{\widehat{y}}_k},$$

or indeed any other formula that satisfies the Secant equation;
10. $k = k + 1$;
11. **goto** 2

It should be noted that the above algorithm terminates finitely on quadratic functions only. For more general functions, the proposed method is comparable to the standard BFGS formula and often improves slightly on it, as the numerical results in Table 1 indicate. The advantages of the new technique over the standard BFGS, on general functions, are detected most when the algorithm moves closer to the minimum, where near quadratic behavior is displayed.

Theorem. *The above algorithm, when applied to quadratic function, generates the sequence of iterates as any quasi-Newton method that satisfies the Secant Equation, that is.*

$$\widehat{x}_k^{new} = \widehat{x}_k^{QN}, \quad \forall k \geq 0,$$

provided the matrices $\widehat{H}_k$ are positive definite so that $\widehat{t}_k > 0$, $\forall k \geq 0$. Consequently, the algorithm locates the minimum in $m (m \leq n)$ iterations, for exact line search.

Proof. We proceed inductively. The following is basically true

$$\widehat{x}_0 = \widehat{x}_0^{QN}, \ \widehat{H}_0 = \widehat{H}_0^{QN}, \ \widehat{g}_0 = \widehat{g}_0^{QN} \text{ and, hence, } \widehat{p}_0 = \widehat{p}_0^{QN}.$$

We, then, assume that the following is true for some iteration k

$$\widehat{x}_k = \widehat{x}_k^{QN}, \ \widehat{H}_k = \widehat{H}_k^{QN}, \ \widehat{g}_k = \widehat{g}_k^{QN} \text{ and, hence, } \widehat{p}_k = \widehat{p}_k^{QN}.$$

Suppose now that t_k is the step size found using some line search method along the search direction $\widehat{p}_k$. Then, for an exact line search, we have

$$\widehat{g}_{k+1}^T \widehat{p}_k = 0.$$

In the proposed algorithm, the value of $\widehat{t}_k$ in Step 8 in the above algorithm is chosen such that

$$\widehat{g}_{k+1}^T \widehat{p}_k = \widehat{g}_{k+1}^{QN^T} \widehat{p}_k^{QN} = 0.$$

Since $\widehat{p}_k = \widehat{p}_k^{QN}$, it follows that

$$\widehat{g}_{k+1} = \widehat{g}_{k+1}^{QN},$$

and, therefore,

$$\widehat{s}_k^{QN} = \widehat{s}_k \text{ since } \widehat{t}_k^{QN} = \widehat{t}_k.$$

Thus, it directly follows that

$\widehat{x}_{k+1}^{new} = \widehat{x}_{k+1}^{QN}$, $\widehat{y}_k^{new} = \widehat{y}_k^{QN}$ and, hence, $\widehat{H}_{k+1}^{new} = \widehat{H}_{k+1}^{QN}$, as required. □

4. Computational Results and Conclusions

In our experiments, we have tested the new algorithm on seventeen standard test functions in dimensions up to 1000. The algorithm has been tested using *C++* on a *PIV 200* processor, using double precision. The cubic interpolation line search used in the algorithm was required to produce a new iterate x_{i+1} satisfying the following standard stability conditions (see Fletcher [1], for example):

$$\begin{aligned} f(x_{i+1}) &\leq f(x_i) + 10^{-4} s_i^T g(x_i); \\ s_i^T g(x_{i+1}) &\geq 0.9 s_i^T g(x_i) \end{aligned}$$

(as in [1]). Table 1 contains the respective numerical results for the Dixon's algorithm (DIX) and the NEW algorithm. The table below reports the number of function calls (NOF) and the number of iterations (NOI) for each test function. Overall totals are also given for NOF and NOI.

Comparisons are affected by the choice of test function, accuracy required and linear search. Nevertheless, the computational results indicate clearly that the extended version gives overall improvement of at least 14% on NOF and/or NOI, although on individual functions some failures have been reported. It is generally evident that the new algorithm has a clear advantage on large dimensions and non-quadratic functions.

All algorithms terminate when $|f - f_{min}| < 1 \times 10^{-10}$.

TEST		**NEW**	**Dixon**
FUNCTION	N	NOI	NOI
		(NOF)	(NOF)
ROSEN	2	22(54)	25(57)
CUBE	2	22(52)	24(55)
BEALE	2	8(20)	9(43)
BOX	2	9(31)	9(44)
FREUD	2	6(15)	7(24)
BIGGS	3	11(30)	13(37)
HELICAL	3	18(39)	18(43)
RECIPE	3	6(19)	6(21)
MIELE	4	30(83)	30(93)
POWELL	4	31(67)	29(69)
WOOD	4	19(42)	18(56)
DIXON	10	17(37)	18(46)
OREN	10	12(64)	12(55)
NON-DIGN	20	20(46)	22(54)
TRI-DIGN	30	28(50)	28(63)
OREN	30	21(90)	21(85)
SHALLOW	40	6(18)	6(30)
FULL	40	39(75)	39(81)
EX-ROSEN	60	23(57)	26(77)
EX-POWELL	60	40(83)	35(89)
EX-WOOD	60	17(42)	18(66)
EX-POWELL	80	43(88)	39(90)
WOLFE	80	48(75)	37(81)
NON-DIGN	90	23(51)	22(55)
EX-WOOD	100	19(42)	18(46)
EX-ROSEN	100	23(55)	26(60)
WOOD	200	29(60)	30(62)
POWELL	200	52(103)	44(95)
POWELL	200	89(140)	93(188)
TOTAL	NOI	731	722
	NOF	1628	1865

Table 1

References

[1] M.C. Biggs, Minimization algorithms making use of non-quadratic properties of the objective function, *Journal of the Institute of Mathematics and its Applications*, **8** (1971), 315-327.

[2] M.C. Biggs, A note on minimization algorithms which make use of non-quadratic properties of the objective function, *Journal of Institute of Mathematics and its Applications*, **12** (1973), 337-338.

[3] K.W. Brodlie, *Some Topics in Unconstrained Minimization*, Ph.D. Thesis, University of Dundee (1973).

[4] C.G. Broyden, The convergence of a class of double rank minimization algorithms II. The new algorithm, *Journal of the Institute of Mathematics and its Applications*, **6** (1970), 221-231.

[5] B. Bunday, *A Basic Optimization Methods*, Edward Arnold, Bedford Square, London (1984).

[6] L.C. W. Dixon, Conjugate directions without linear searches, *Journal of Institute of Mathematics and its Applications*, **11** (1973), 317-328.

[7] Dixon, L.C.W. Conjugate gradient algorithm quadratic termination without line searches, *Journal of the Institute of Mathematics and its Applications*, **15** (1975).

[8] R. Fletcher, M.J.D. Powell, A rapidly convergent descent method for minimization, *Computer Journal*, **6** (1963), 163-168.

[9] H.R. Hestenes, E. Stiefel, Methods of conjugate gradients for solving linear systems, *Journal of Research of the National Bureau of Standards*, **49** (1952), 409-436.

[10] G.P. McCormick, K. Ritter, Methods of conjugate directions versus quasi-Newton methods, *Mathematical Programming*, **3** (1972), 101-l16.

[11] L. Nazareth, A relationship between BFGS and conjugate-gradient algorithms and its implementations for new algorithms, *SIAM Journal on Numerical Analysis*, **16** (1979), 794-800.

[12] S.S. Oren, Self-scaling variable metric algorithm without line search for unconstrained minimization, *Mathematics of Computation*, **27** (1973), 873-885.

[13] S.S. Oren, Self-scaling variable metric algorithm, Part II, *Management Science*, **20** (1974), 863-874.

[14] S.S. Oren, On the selection of parameters in self-scaling variable metric algorithms, *Mathematical Programming*, **3** (1974), 351-367.

[15] S.S. Oren, D.G. Luenberger, Self-scaling variable metric algorithm, Part I, *Management Science*, **20** (1974), 845-862.

[16] S.S. Oren, E. Spedicato, Optimal conditioning of self-scaling variable metric algorithms, *Mathematical Programming*, **10** (1976), 70-90.

Appendix

All the test functions used in this paper are from general literature.

1. Rosenbrock banana function, $n = 2$, $f = 100(x_2 - x_1^2)^2 + (1 - x_1)^2, x_0 = (-1.2, 1.0)^T$.

2. Cube function, $n = 2$, $f = 100(x_2 - x_1^3)^2 + (1 - x_1)^2$, $x_0 = (-1.2, 1.0)^T$.

3. Scale function, $n = 2$, $f = (1.5 - x_1(1 - x_2))^2 + (2.25 - x_1(1 - x_2^2))^2 + (2.625 - x_1(1 - x_2^3))^2$, $x_0 = (0, 0)^T$.

4. Box function, $n = 2$, $f = \sum_{i=1}^{n}(e^{-x1}z_i - e - x^2 z_i - e^{-zi} + e{-10zi})^2$, where $z_i = (0.1)^i$ and $x_0 = (5, 0)^T, i = 1, ...$

5. Frudenstein and Roth function, $n = 2$, $f = [-13 + x_1 + ((5 - x_2)x_2 - 2)x_2]^2 + [-29 + x_1 + ((1 + x_2)x_2 - 14)x_2]^2$, $x_0 = (30, 3)^T$.

6. Recipe function, $n = 3$, $f = (x_1 - 5)^2 + x_2^2 + x_3^2/(x_1 - x_2)^2, x_0 = (2, 5, 1)^T$.

7. Biggs function, $n = 3$, $f = \sum_{i=1}^{n}(e^{-x1zi} - x_3 e^{-x2zi} - e^{-zi} + 5e^{-10zi})^2$, where $z_i = (0.1)i$ and $x_0 = (1, 2, 1)^T, i = 1....$

8. Helical Valley function, $n = 3$, $f = 100\{[x_3 - 1.0]^2 + [r - 1]^2\} + x_3^2$, where $r = 1/2 \arctan(x_2/x_l)$, for $xi > 0$ and $r = 1/2 + 1/2 \arctan(x_2/x_l)$ for $x_l < 0$, $x_0 = (-1, 0, 0)^T$.

9. Miele and Cornwell function, $n = 4$, $f = (e^{x_1} - 1)^2 + \tan 4(x_3 - x_4) + 100(x_2 - x_3)^2 + 8x_1 + (x_4 - l)^2$, $x_0 = (1, 2, 2, 2)^T$.

10. Dixon function, $n = 10$, $f = (1 - x_l)^2 + (1 - x_{10})^2 + \sum_{i=1}^{n}(x_i^2 - x_i + 1)^2$, $x_0 = (-1; ..)^T$, $i = 2, ...$

11. Oren and Spedicato power function, $n = 10, 30$, $f = \sum_{i=1}^{n}(i - x_i^2)^2$, $x_0 = (1, ...)^T$.

12. Non diagonal variant of Rosenbrock function, $n = 20, 90$, $f = \sum_{i=1}^{n}[100(x_l - x_i^2)^2 + (1 - x_i)^2]$, $x_0 = (-1, ..)^T, i = 1...$

13. Tri-diagonal function, $n = 30$, $f = [\sum_{i=2}^{n}(2x_i - x_{i-1})^2]$, $x_0 = (1; ....)^T$.

14. Full set of distinct eigenvalues problem, $n = 40$, $f = (x_l - 1)^2 + \sum_{i=2}^{n}(2x_i - xi_{-1})^2$, $x_0 = (1; ..)^T$.

15. Shallow function (generalized form), $n = 40$, $f = \sum_{i=1}^{n/2}(x_{2i-1}^2 - x_{2i})^2 + (1 - x_{2i-1})^2$, $x_0 = (-2; .)^T$.

16. Powell function (generalized form), $n = 60, 80$, $f = \sum_{i=1}^{n/4}[(x_{4i-3} + l0x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2 + (x_{4i-2} - 2x_{4i-1})^4 + 10(x_{4i-3} - x^{4i})^4]$, $x_0 = (3, -1, 0, 1; ...)^T$.

17. Wood function (generalized form), $n = 60, 100$, $\sum_{i=1}^{n/4} f = [100(x_{4i-2} - x_{4i-3}^2)^2 + (1 - x_{4i-3})^2 + 90(x_{4i} - x_{4i-1}^2)^2 + (1 - x_{4i-1})^2 + 10.1(x_{4i-2} - 1)^2 + (x_{4i} - 1)^2 + 19.8(x_{4i-2} - 1)(x_{4i-l})$, $x_0 = (-3, -1; -3, -1, ...)^T$.