J. KSIAM Vol.8, No.2, 7-13, 2004

A NEW CONJUGATE GRADIENT MINIMIZATION METHOD BASED ON EXTENDED QUADRATIC FUNCTIONS

ISSAM.A.R. MOGHRABI

ABSTRACT. A Conjugate Gradient (CG) algorithm for unconstrained minimization is proposed which is invariant to a nonlinear scaling of a strictly convex quadratic function and which generates mutually conjugate directions for extended quadratic functions. It is derived for inexact line searches and is designed for the minimization of general nonlinear functions. It compares favorably in numerical tests with the original Dixon algorithm on which the new algorithm is based.

1. INTRODUCTION

Let A denote a symmetric and positive definite n by n matrix. For $x \in R^n$, we define

$$q(x) = (1/2)x^T Ax + b^T x + \gamma.$$

Let $F : R \longrightarrow R^+$ denote a strictly monotonic increasing function (with a non-vanishing first derivative) and define

$$f(x) = F(q(x)), \tag{1}$$

where such a function f is called an extended quadratic function.

When a minimization algorithm is applied to f, the ith iterate is denoted by x_i, the corresponding function value by f_i and its gradient by g_i. The function and gradient values of q are denoted by q_i and G_i, respectively, and the derivative of F at q_i is denoted by $F_i^{'}$. We note that $g_i = F_i^{'} G_i$ and define $\sigma_i = c_i/c_{i+1}$ for each i, where $c_j = F_j^{'}$. It is assumed here that when applying a minimization procedure to an extended quadratic function one only has specific knowledge of f_i, σ_i and g_i.

In order to improve the rate of convergence for more general functions, other than the quadratic form, new algorithms have been suggested by several authors [1,2,3,4,5]. All of these algorithms derive an expression for $\sigma_i g_{i+1}$, and hence include the property that the extended quadratic f is minimized in a finite number of iterations assuming exact line searches are used. In this paper this requirement is dropped, and the formula for $\sigma_i g_{i+1}$is based on inexact line searches.

2. Derivation of σ_i

For reference, we define a general algorithm for the classical CG-method as follows:

Given a starting point $x_0 \in R^n$, function $f(x)$ and its gradient $g(x)$, compute $p_0 = -g(x_0)$ initially .Then iterate using $x_{i+1} = x_i + \alpha_i p_i$, where α_i is determined by an exact line search routine such as cubic interpolation (see [6] and [7]) and for $i = 0, 1, 2, \ldots$ The new search direction is computed using the recurrence $p_{i+1} = -g_{i+1} + \beta_i p_i$, for $i \geq 0$.

Lemma 1.

Let $G_{i+\frac{1}{2}} = G(x_i + \frac{1}{2}\alpha_i p_i)$, which is consistent with the natural definition $G_{i+j} = G(x_i + j\alpha_i p_i)$. Then the following holds (for a quadratic function):

$G_{i+1} = 2G_{i+\frac{1}{2}} - G_i$.

The proof is similar to that presented in [3].

Now define x_i and $p_i \in R^n$ (for $i = 0, 1, ..$) and $x_{i+1} = x_i + \alpha_i p_i$, where α_i is an arbitrary positive real number such that $g_{i+1}^T p_i = 0$. We now define the following expressions for g_{i+1}^+ and $g_{i+1/2}^+$ as follows:

$$g_{i+1}^+ = g_{i+1} - (g_{i+1}^T g_i / g_i^T g_i) g_i \tag{2}$$

and

$$g_{i+1/2}^+ = (g_{i+1/2}^T g_i / g_i^T g_i) g_i \tag{3}$$

so that the following lemma holds for the extended quadratic model.

Lemma 2.

If $\sigma_i = c_i / c_{i+1}$, where $c_j = F_j'$, then

$$\sigma_i = c_i / c_{i+1} = ((g_{i+1/2}^{+T} g_{i+1})(g_i^T g_i)) / ((g_{i+1}^{+T} g_{i+1})(g_{i+1/2}^T g_i) - (g_{i+1/2}^{+T} g_{i+1})(g_{i+1}^T g_i)).$$

Proof

For g_{i+1}, we have directly

$$g_{i+1} = c_{i+1} G_{i+1} = c_{i+1}(G_i + \alpha_i A p_i) \tag{4}$$

Using 2, 3 and 4, we have

$$g_{i+1}^+ = c_{i+1}(G_i + \alpha_i A p_i) - ((c_i c_{i+1}(G_i + \alpha_i A p_i)^T G_i) / c_i^2 G_i^T G_i) c_i G_i.$$

$$= c_{i+1}\alpha_i (A p_i - (G_i A p_i / G_i^T G_i) G_i) \tag{5}$$

Multiplying (5) by a non-zero vector g_{i+1} and dividing we get :

$(g_{i+1}^{+T} g_{i+1}) / (g_{i+1}^T g_{i+1/2}^+) = 2c_{i+1} / c_{i+1/2}$.

From Lemma 1 we have

$g_{i+1} = (2c_{i+1}/c_{i+1/2})g_{i+1/2} - (c_{i+1}/c_i)g_i$.
Therefore,

$g_{i+1} = [(g_{i+1}^{+T} g_{i+1})/(g_{i+1}^T g_{i+1/2}^+)]g_{i+1/2} - (c_{i+1}/c_i)g_i$.
On the assumption that α_i is an arbitrary positive value and p_i is chosen to be a non-zero vector, then
$g_{i+1}^T g_i = [(g_{i+1}^{+T} g_{i+1})/(g_{i+1}^T g_{i+1/2}^+)]g_{i+1/2}^T g_i - (c_{i+1}/c_i)g_i^T g_i$,
from which it follows that
$c_{i+1}/c_i = [(g_{i+1}^{+T} g_{i+1})(g_{i+1/2}^T g_i)/(g_{i+1}^T g_{i+1/2}^+)(g_i^T g_i)] - g_{i+1}^T g_i/g_i^T g_i$,
or, equivalently,

$$\sigma_i = c_i/c_{i+1} = \frac{(g_{i+1}^T g_{i+1/2}^+)(g_i^T g_i)}{(g_{i+1}^{+T} g_{i+1})((g_{i+1/2}^T g_i) - (g_{i+1}^T g_{i+1/2}^+)(g_{i+1}^T g_i)}, \tag{6}$$

as required.

3. ALGORITHM EDX

In this section we first describe the CG-methods based on the quadratic model proposed by Dixon in [6] which employs inexact line searches, termed (DX): this is modified to be invariant to a nonlinear scaling of $q(x)$. We call this the extended Dixon method (EDX).

If we let a '*' superscript denote an estimated quantity, then the search direction in Dixon's method p_{i+1} is given by the formula
$p_{i+1} = -g_{i+1}^* + \beta_i^* p_i$, for $i > 0$ and where $p_0 = -g(x_0)$.

Dixon presented several versions of the modified CG-method in [6]; we have selected here two for extension, the well-known Hestenes and Stiefel, and the Fletcher-Reeves versions, referred to as Dixon Versions **EDXA** and **EDXB**, respectively, with the necessary changes made to incorporate σ_i, defined in (6). We therefore define the respective EDX formulae

$$\beta_i^* = g_{i+1}^{*T}[\sigma_i g_{i+1}^* - g_i^*]/p_i^T[\sigma_i g_{i+1}^* - g_i^*] \tag{EDXA}$$

and

$$\beta_i^* = \sigma_i g_{i+1}^{*T} g_{i+1}^* / g_i^{*T} g_i^*, \tag{EDXB}$$

where the term g_i^* represents the estimated gradient term, namely the estimated value of the gradient at x_i^*, the point which would have been reached with exact line searches. For a quadratic function the gradients can be evaluated as follows:
$g_{i+1}^* = g_i^* + (1 - \frac{g_{i+1}^T p_i}{y_i^T p_i})y_i$, for $i > 0$, where $g_0^* = g_0$, and $y_i = g_{i+1} - g_i$. The search directions generated by this extended algorithm are not necessarily descent ones for an

arbitrary function. In our experiments, we use the following test to check whether the new direction is sufficiently downhill:

$$p_{i+1}^T g_{i+1} \geq 0.75 g_{i+1}^T g_{i+1} \tag{7}$$

Whenever relation (7) is not satisfied the iteration is restarted, as follows: the estimated error-vector term is used as in [6], i.e, $e_{n+1} = \sum_{i=0}^{n} \varepsilon_i p_i$, where $\varepsilon_i = (\alpha_i g_{i+1}^T p_i)/(y_i^T p_i)$. This estimated error-vector term is added to x_{n+1} to find x_{n+2}, i.e $x_{n+2} = x_{n+1} +\, e_{n+1}$ and the iteration is then restarted with $.-g_{n+2}$

4. Computational Results and Conclusion

Senevteen standard test functions are employed, in dimensions up to 1000, in order to examine the overall effectiveness of the two new algorithms.

The algorithms were tested using C++ on a PIV 200 processor, using double precision. The effectiveness of modifying DX to EDX is tested with the two different updating formulae (A and B) for the search directions. In each case the line search accuracy parameter α is chosen to satisfy (7). The line-search algorithm used is a standard cubic interpolation (as in [9]). Table 1 contains the respective numerical results for the EDX and DX algorithms, with versions A and B. The table reports the number of function calls (NOF), the number of iterations (NOI) and the corresponding function value F are given for each test function. Overall totals are also given for NOF and NOI.

Comparisons are affected by the choice of test function, accuracy required, line search and restarting criterion. Nevertheless, the computational results indicate clearly that the extended versions give overall improvements of at least 9% on NOF or NOI, although on individual functions there can be a loss of efficiency. The Hestenes-Stiefel Version (A) is clearly inferior to the Fletcher-Reeves Version (B).

It is generally evident that the new algorithms have a clear advantage on higher dimensions and general non-quadratic functions.

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

TABLE : (1)

TEST FUNCTIONS	**N**	**EDXA NOI(NOF)**	**EDX NOI(NOF)**	**EDXB NOI(NOF)**
ROSEN	2	22(54)	25(57)	23(59)
CUBE	2	22(57)	24(55)	23(60)
BEALE	2	8(20)	9(43)	9(30)
BOX	2	9(41)	9(44)	9(41)
FREUD	2	6(18)	7(24)	6(21)
BIGGS	3	11(31)	13(37)	11(30)
HELICAL	3	18(39)	18(43)	17(36)
RECIPE	3	6(19)	6(21)	7(19)
MIELE	4	30(83)	30(93)	28(77)
POWELL	4	31(67)	29(69)	23(59)
WOOD	4	19(42)	18(56)	20(41)
DIXON	10	17(37)	18(46)	23(49)
OREN	10	12(64)	12(55)	13(56)
NON-DIGN	20	20(46)	22(54)	21(47)
TRI-DIGN	30	28(57)	28(63)	29(59)
OREN	30	21(95)	21(85)	23(99)
SHALLOW	40	6(18)	6(30)	6(20)
FULL	40	39(79)	39(81)	38(83)
EX-ROSEN	60	23(57)	26(77)	24(60)
EX-POWELL	60	40(83)	35(89)	42(70)
EX-WOOD	60	17(42)	18(66)	18(48)
EX-POWELL	80	43(88)	39(90)	41(82)
WOLFE	80	48(75)	37(81)	41(79)
NON-DIGN	90	23(53)	22(55)	23(59)
EX-WOOD	100	19(42)	18(46)	19(39)
EX-ROSEN	100	23(57)	26(60)	23(48)
WOOD	200	29(65)	30(62)	30(58)
POWELL	200	52(133)	44(95)	41(93)
POWELL	1000	89(240)	93(198)	85(186)
TOTAL	**NOI**	731	722	716
	NOF	1802	1875	1708

REFERENCES

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

[2] S.S. Oren, "Self-scaling variable metric algorithm, Part II", Management Science 20 (1974), 863-874.
[3] S.S. Oren, "On the selection of parameters in self-scaling variable metric algorithms", Mathematical Programming 3 (1974) 351-367.
[4] S.S. Oren and D.G. Luenberger, "Self-scaling variable metric algorithm, Part I", Management Science 20 (1974) 845-862.
[5] S.S. Oren and E. Spedicato, "Optimal conditioning of self-scaling variable metric algorithms", Mathematical Programming l0 (1976) 70-90.
[6] Dixon, L.C.W. Conjugate gradient algorithm quadratic termination without line searches, Journal of the Institute of Mathematics and its Applications 15, 1975.
[7] 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.
[8] S.S. Oren, "Self-scaling variable metric algorithm without line search for unconstrained minimization", Mathematics of Computation 27 (1973), 873-885.
[9] G.P. McCormick and K. Ritter, "Methods of conjugate directions versus quasi-Newton methods", Mathematical Programming 3 (1972) 101-116.
[10] 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.
[11] 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 Applications12 (1973) 337-338.
[12] H.R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems", Journal of Research of the National Bureau of Standards, 49 (1952), 409-436.
[13] 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.
[14] K.W. Brodlie, "Some topics in unconstrained minimization", Ph.D. thesis, University of Dundee, (1973).
[15] B. Bunday, "A Basic Optimization Methods", Edward Arnold, Bedford Square, London, (1984).

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 $-x^2z_i - e^{-zi} + e^{-10zi})\ ^2$,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 $x_l > 0$

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,
$f = (e^{x_1} - 1)^2 + \tan 4 (x_3 - x_4) + 100 (x_2 - x_3)^2 + 8x_1 + (x_4 - 1)^2$, $x_0 = (1, 2, 2, 2)^T$.
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,
$f = \sum_{i=1}^{n/4}[(x_{4i-3} + 10 x_{4i-2})^2 + 5 (x_{4i-1} - x_{4i})^2 + (x_{4i-2} - 2 x_{4i-1})^4 + 10 (x_{4i-3} - x^{4i})^4]$,
$x_0 = (3,-1,0,1;...)^T$.
17. Wood function(Generalized form), n = 60,100,
$\sum_{i=1}^{n/4} f = [100 (x_{4i-2} - x_{4i-3}^2)^2 + (1 - x_{4i-3})^2 + 90 (x_{4i} - x_{4i-1}^2)^2 + (1 - x_{4i-1})^2$
$+ 10.1 (x_{4i-2} - 1)^2 + (x_{4i} - 1)^2 + 19.8 (x_{4i-2} - 1)(x_{4i-l})$, $x_0 = (-3,-1;-3,-1,...)^T$.

Computer Science Division, Faculty of Science,
Beirut Arab University, Beirut, Lebanon
Email: imoghrabi@bau.edu.lb