DISCUSSION PAPER NO. 228

A SELF CORRECTING CONJUGATE GRADIENT ALGORITHM

by

Avinoam Perry
Assistant Professor
Department of Decision Sciences

July 1976

Abstract

In this paper we develop a new procedure for constructing a conjugate gradient direction equation. The new equation is a linear combination of two orthogonal vectors one of which is the negative gradient. This procedure is reduced to the method of Perry [11] whenever line search is perfectly accurate. Otherwise, as reflected by our computational results, the method is more effective than any other conjugate gradient algorithm we have tested.

Introduction

Conjugate direction algorithms can be divided into two major classes. The first calss consists of algorithms with no memory such as Fletcher-Reeves (FRCG) method [6], Polak-Rebiére (PRCG) method [8,12], and the method of Perry (PMCG) [11]. The second class consists of quasi-Newton methods which apply a matrix update approximating the hessian inverse of f(x). Among the most sucessful quasi-Newton procedures we have the DFP update (Davidon [3], Fletcher Powell [5]), the BR1 update (Broyden's Rank One [1]), and the BFGS update (Broyden [2], Fletcher [4], Goldfarb [7], Shanno [15]). Computational experience with these methods support the assumptions that the BFGS method is the one with the best average performance (among all those mentioned above). Consequently, recent versions of the GRG method [9] employ a modification of this algorithm which accomodates upper and lower bounds on the variables. This approach may not be suitable for large scale nonlinear programs because of the need for storing and updating the large matrix approximating the hessian inverse of f(x). Due to this fact, conjugate gradient algorithms with no memory become more attractive for large scale optimization than quasi-Newton algorithms [14].

This line of thought has been the prime motivation for the development of a successful conjugate gradient code. In a recent publication [11] we have discussed one step forward in that direction. Our modified conjugate gradient algorithm performed significantly better than the Fletcher-Reeves method and was slightly better than the

Polak-Rebiére procedure, the performance of which tends to deteriorate rapidly as line search accuracy is reduced. In this communication we have been able to introduce an improvement over the last version of our modified conjugate gradient. The new version presented here applies a self correcting mechnaism to the conjugate gradient equation in [11]. This correction forces the orthogonality assumption $d_k' g_{k+1} = 0$ (where $d_k \in E^n$ is a direction vector which carries a point $x_k \in E^n$ to a point $x_{k+1} \in E^n$, and where $g_{k+1} \equiv \nabla f(x_{k+1})$) to hold even when line search accuracy is reduced. Our computational experience indicates that the overall performance of the self correcting conjugate gradient method is better than any other conjugate gradient algorithm despite the fact that the computational effort per iteration is increased as a result of applying the self correcting mechanism.

I. Derivation of the Self Correcting Conjugate Gradient Method

Let $p_k = x_{k+1} - x_k$ and $q_k = g_{k+1} - g_k$ where $g_k = \nabla f(x_k)$ and the unconstrained nonlinear program is: minimize $f(x)$, $x \in E^n$. Let $d_k = (1/\alpha_k)p_k$ where α_k is a scalar minimizing the one dimensional program: $\min_{\alpha \geq 0} f(x_k + \alpha d_k)$.

The modified conjugate gradient equation which we developed in [11] is:

$$d_{k+1} = -g_{k+1} + \frac{(q_k - \alpha_k d_k)' g_{k+1}}{d_k' q_k} \cdot d_k \tag{1}$$

The above equation, as well as Fletcher-Reeves' [6] and the Polak-Rebiere's [8] equations are constructed so as to minimize a quadratic function $f(x)$, $x \in E^n$ with perfect line search, in no more than n+1 steps.*

In the construction of Fletcher-Reeves and Polak-Rebiere equations the orthogonality assumption: $d_k' g_{k+1} = 0$ is made explicitly. This assumption always holds under the ideal** conditions, but does not hold under more general situations. One of the major properties of the modified conjugate gradient (1) is the relaxation of that assumption.

This relaxation becomes redundant in the self correcting conjugate gradient algorithm due to the fact that the new algorithm has the orthogonality property: $d_k' g_{k+1} = 0$ built into it. The cost associated with the enforcement of this property is an extra gradient evaluation per iteration, but as suggested by our computational experience, the overall performance of the self correcting algorithm seems to be better than any of the conjugate gradient algorithms mentioned above.

---

*A step is defined as a movement in E^n from x_k to x_{k+1}.

**$f(x)$ is quadratic and the line search is perfect.

The self correcting mechanism is applied whenever $|p_k' g_{k+1}| \geq \varepsilon$. (where $\varepsilon > 0$ is a small scalar).

If $|p_k' g_{k+1}| \geq \varepsilon$ one can define a vector $\bar{p}_k$ such that $\bar{p}_k' g_{k+1} = 0$. Given x_{k+1} and $\bar{p}_k$, there exists a point $\bar{x}_k = x_{k+1} - \bar{p}_k$ such that the vector $\bar{g}_k \equiv \nabla f(\bar{x}_k)$.

If $\bar{p}_k' \bar{g}_k < 0$ then, by restarting at the point $\bar{x}_k$ and moving along the descent direction $\bar{p}_k$ the problem: minimize $f(\bar{x}_k + \alpha \bar{p}_k)$ $\alpha > 0$ yields the exact solution $\alpha^* = 1$ which implies $\bar{x}_k + \bar{p}_k = x_{k+1}$.

The algorithm

Step 0: Let $d_o = -g_o$

Step 1: Minimize $h(\alpha) = f(x_k + \alpha d_k)$

Step 2: set $x_{k+1} = x_k + \alpha_k d_k$

Step 3: Let $g_{k+1} \equiv \nabla f(x_{k+1})$, $p_k \equiv x_{k+1} - x_k$, $q_k \equiv g_{k+1} - g_n$

Step 4: If $|p_k' g_{k+1}| < \varepsilon$ go to 8, otherwise go to 5

Step 5: Let $$\bar{p}_k = p_k - \frac{p_k' g_{k+1}}{g_{k+1}' g_{k+1}} \cdot g_{k+1}$$

Step 6: Let $\bar{x}_k = x_{k+1} - \bar{p}_k$, $\bar{g}_k \equiv \nabla f(\bar{x}_k)$, $\bar{q}_k \equiv g_{k+1} - \bar{g}_k$

Step 7: Set $p_k = \bar{p}_k$, $q_k = \bar{q}_k$, and $g_k = \bar{g}_k$

Step 8: Compute: $$d_{k+1} = -g_{k+1} + \frac{q_k' g_{k+1}}{p_k' q_k} \cdot p_k$$

This procedure insures that if $\bar{x}_k$ is a starting point and $\bar{p}_k$ is a descent direction such that $\bar{x}_k + \bar{p}_k = x_{k+1}$, then p_{k+1} is constructed by taking a linear combination of two orthogonal vectors, one of which

is the negative gradient $\nabla f(x_{k+1})$, and the other is conjugate to p_{k+1}. These two desired properties are essential in proving quadratic termination of conjugate gradient algorithms and their enforcement proves to be effective as indicated by the following computational results.

Computational Experience

Experiments with the self correcting conjugate gradient algorithm (SCCG) as well as, Fletcher-Reeves (FRCG) [6] , Polak-Rebière (PRCG) [8], Perry [11], BR1 [1], DFP [3,5], and BFGS [2,4,7,15] methods involved seven well known test problems which are denoted here as functions I through VII respectively (see appendix for a detailed description of each function and its source). All problems were solved by each one of the algorithms above under two different line search accuracy measures. The line search technique applied is the well known quadratic interpolation method. In order to insure successful implementation of the line search procedure an effort was made to secure a unimodel region before the first interpolation was performed. We also define "number of iterations per line search" as the number of times the quadratic interpolation was performed along a given direction.

Another measure of accuracy used in our study is δ where:

$$(2) \quad \delta > \frac{|p(\alpha^*) - f(\alpha^*)|}{p(\alpha^*)}$$

and where:

$p(\alpha^*)$ is the value of the interpolating polynomial at the point $x_k + \alpha^* d_k$, and $f(\alpha^*)$ is the value of the function at the same point.

The term "stage" is defined as the step carrying a point x_k along a direction d_k to a new point $x_{k+1} = x_k + \alpha_k d_k$. It follows that the total number of stages per algorithm is equal to the total number of gradient evaluations (twice that number in the case of SCCG).

The statistics "total number of function evaluations" includes the number of gradient evaluations multiplied by n (if one is interested

in number of function evaluations not including gradient evaluations, the total number of stages multiplied by n (or by 2n for SCCG) should be subtracted from the total number of function evaluations). Each time a direction vector pointed upwards rather than downwards it was replaced by the direction of steepest descent. Although we do not provide statistics regarding the number of times per experiment this phenomenon took place, we wish to note that this procedure was a significant factor in determining the algorithmic mapping of BR1, as well as, FRCG and PRCG whenever line search accuracy measures were light.

The stopping rule applied throughout was

$$(3)\quad |\nabla f(x^*)| \leq 0.0001$$

If an experimental run exceeded a given number of stages before reaching the point x^* in (3) the run was terminated by the operator.

All computer programs were coded in APL using interactive mode [10] and were run on the CDC6400 computer at Northwestern University.

In the following tables we present our computational results under two measures of line search accuracy denoted as mode 1 (N=5, δ=.01) and mode 2 (N=1) where the variable name N stands for the maximum number of iterations per one dimensional search, and δ is defined as in (3). The one dimensional search was terminated whenever one of these constraints became active.

Function 1
$x_0 = -1.2, 1$

Algorithm	# stages		# function evaluations		Reported best value	
	Mode 1	Mode 2	Mode 1	Mode 2	Mode 1	Mode 2
SCCG	13	17	182	188	$9.92 \cdot 10^{-13}$	$2.10 \cdot 10^{-14}$
FRCG	65	206	581	1764	$2.67 \cdot 10^{-10}$	$7.46 \cdot 10^{-10}$
PRCG	23	31	253	325	$8.61 \cdot 10^{-14}$	$1.84 \cdot 10^{-13}$
PMCG	23	25	234	228	$1.87 \cdot 10^{-10}$	$1.93 \cdot 10^{-15}$
BR1	26	24	284	223	$5.34 \cdot 10^{-16}$	$1.31 \cdot 10^{-13}$
DFP	23	25	224	207	$1.91 \cdot 10^{-11}$	$6.05 \cdot 10^{-12}$
BFGS	20	25	209	216	$2.85 \cdot 10^{-13}$	$8.23 \cdot 10^{-11}$

Function 2
$x_0 = -1.2, 1$

Algorithm	# stages		# function evaluations		Reported best value	
	Mode 1	Mode 2	Mode 1	Mode 2	Mode 1	Mode 2
SCCG	5	6	66	64	$1.99 \cdot 10^{-11}$	$9.84 \cdot 10^{-14}$
FRCG	8	7	85	69	$8.16 \cdot 10^{-9}$	$8.27 \cdot 10^{-9}$
PRCG	6	9	64	79	$3.53 \cdot 10^{-10}$	$7.48 \cdot 10^{-14}$
PMCG	5	6	60	62	$3.03 \cdot 10^{-12}$	$1.51 \cdot 10^{-12}$
BR1	5	6	58	55	$1.51 \cdot 10^{-10}$	$9.69 \cdot 10^{-10}$
DFP	5	7	56	68	$7.89 \cdot 10^{-11}$	$2.75 \cdot 10^{-15}$
BFGS	5	7	57	67	$1.76 \cdot 10^{-11}$	$9.84 \cdot 10^{-14}$

Function 3
$x_0 = -1.2, 1$

Algorithm	# stages		# function evaluations		Reported best value	
	Mode 1	Mode 2	Mode 1	Mode 2	Mode 1	Mode 2
SCCG	3	4	55	59	$6.82 \cdot 10^{-19}$	$3.63 \cdot 10^{-18}$
FRCG	5	5	69	65	$4.70 \cdot 10^{-12}$	$3.69 \cdot 10^{-15}$
PRCG	5	5	60	62	$3.33 \cdot 10^{-17}$	$3.77 \cdot 10^{-14}$
PMCG	5	5	65	56	$5.86 \cdot 10^{-18}$	$2.02 \cdot 10^{-13}$
BR1	3	5	50	56	$4.93 \cdot 10^{-17}$	$3.17 \cdot 10^{-14}$
DFP	4	4	51	45	$8.59 \cdot 10^{-17}$	$5.35 \cdot 10^{-12}$
BFGS	3	4	50	45	$6.35 \cdot 10^{-18}$	$3.78 \cdot 10^{-11}$

Function 4
$x_0 = -1.2, 1$

Algorithm	# stages		# function evaluations		Reported best value	
	Mode 1	Mode 2	Mode 1	Mode 2	Mode 1	Mode 2
SCCG	12	20	162	222	$2.43 \cdot 10^{-12}$	$3.51 \cdot 10^{-11}$
FRCG	13	27	161	264	$6.91 \cdot 10^{-11}$	$3.82 \cdot 10^{-10}$
PRCG	13	27	152	284	$6.30 \cdot 10^{-12}$	$3.05 \cdot 10^{-10}$
PMCG	13	23	138	206	$1.99 \cdot 10^{-13}$	$3.15 \cdot 10^{-15}$
BR1	16	24	220	254	$1.88 \cdot 10^{-9}$	$5.55 \cdot 10^{-12}$
DFP	15	24	168	226	$1.07 \cdot 10^{-12}$	$7.11 \cdot 10^{-16}$
BFGS	14	22	161	215	$1.66 \cdot 10^{-12}$	$1.66 \cdot 10^{-16}$

Function 5
$x_0 = -3,-1,-3,-1$

Algorithm	# stages		# function evaluations		Reported best value	
	Mode 1	Mode 2	Mode 1	Mode 2	Mode 1	Mode 2
SCCG	59	68	855	1065	$5.39 \cdot 10^{-11}$	$2.85 \cdot 10^{-11}$
FRCG	1500*	1500*	14031*	13542*	$2.84 \cdot 10^{-3}$	$3.27 \cdot 10^{-1}$
PRCG	85	115*	870	1193*	$4.37 \cdot 10^{-10}$	$1.56 \cdot 10^{-4}$
PMCG	75	115	839	1263	$1.66 \cdot 10^{-10}$	$2.26 \cdot 10^{-9}$
BR1	74	207	765	2362	$8.12 \cdot 10^{-11}$	$1.08 \cdot 10^{-9}$
DFP	71	128	780	1439	$7.93 \cdot 10^{-12}$	$2.66 \cdot 10^{-16}$
BFGS	38	44	436	478	$4.54 \cdot 10^{-14}$	$1.11 \cdot 10^{-14}$

Function 6
$x_0 = 1,1,1,1$

Algorithm	# stages		# function evaluations		Reported best value	
	Mode 1	Mode 2	Mode 1	Mode 2	Mode 1	Mode 2
SCCG	35	40	545	594	$3.62 \cdot 10^{-9}$	$1.75 \cdot 10^{-9}$
FRCG	100*	100*	975*	1047*	$4.23 \cdot 10^{-6}$	$3.29 \cdot 10^{-4}$
PRCG	97	100*	1060	1030*	$1.44 \cdot 10^{-9}$	$4.37 \cdot 10^{-6}$
PMCG	99	89	1085	959	$3.30 \cdot 10^{-9}$	$4.20 \cdot 10^{-9}$
BR1	100*	100	1002*	1031	$3.65 \cdot 10^{-4}$	$2.01 \cdot 10^{-4}$
DFP	16	24	201	231	$3.33 \cdot 10^{-9}$	$1.61 \cdot 10^{-10}$
BFGS	15	19	196	197	$1.63 \cdot 10^{-9}$	$5.79 \cdot 10^{-10}$

*Terminated by operator

Function 6
$x_0 = 3,-1,0,1$

Algorithm	# stages		# function evaluations		Reported best value	
	Mode 1	Mode 2	Mode 1	Mode 2	Mode 1	Mode 2
SCCG	35	45	536	635	$1.12 \cdot 10^{-9}$	$2.07 \cdot 10^{-9}$
FRCG	100*	100*	1026*	1062*	$3.18 \cdot 10^{-5}$	$2.47 \cdot 10^{-6}$
PRCG	74	64	822	656	$7.10 \cdot 10^{-9}$	$7.22 \cdot 10^{-9}$
PMCG	78	61	891	664	$6.33 \cdot 10^{-9}$	$6.65 \cdot 10^{-9}$
BR1	100*	100*	1140*	1059*	$9.44 \cdot 10^{-5}$	$1.05 \cdot 10^{-5}$
DFP	20	19	253	214	$9.93 \cdot 10^{-10}$	$5.54 \cdot 10^{-9}$
BFGS	18	19	233	212	$3.02 \cdot 10^{-11}$	$3.07 \cdot 10^{-11}$

Function 7
$x_0 = 1, 1$

Algorithm	# stages		# function evaluations		Reported best value	
	Mode 1	Mode 2	Mode 1	Mode 2	Mode 1	Mode 2
SCCG	5	5	72	60	$1.47 \cdot 10^{-11}$	$2.41 \cdot 10^{-12}$
FRCG	10	15	112	140	$2.44 \cdot 10^{-11}$	$1.00 \cdot 10^{-11}$
PRCG	6	6	73	62	$1.39 \cdot 10^{-12}$	$1.31 \cdot 10^{-12}$
PMCG	6	6	72	59	$1.22 \cdot 10^{-12}$	$3.26 \cdot 10^{-12}$
BR1	6	7	72	72	$1.19 \cdot 10^{-12}$	$1.93 \cdot 10^{-13}$
DFP	6	6	73	66	$2.76 \cdot 10^{-15}$	$6.68 \cdot 10^{-18}$
BFGS	6	6	73	65	$2.74 \cdot 10^{-15}$	$1.24 \cdot 10^{-17}$

*Terminated by operator

Concluding remarks

As reflected by our computational results, the self correcting conjugate gradient algorithm performed better than all other conjugate gradient methods tested here. This conclusion is especially evident in the results corresponding to functions 1, 5 and 6 which are the most difficult problems tested in this study. The outcomes corresponding to functions 2, 3, 4, and 7 were not as significant as the ones above, but, nevertheless, a careful evaluation of the results in these cases does not contradict the conclusions reached with regard to the more difficult problems. When evaluating the performance of SCCG one should note that the statistics "# function evaluations" contains all gradient evaluations weighted by n. Although this way of combining function and gradient evaluations into one index is somewhat arbitrary, the reader should be reminded that this method of presenting the amount of effort invested in solving a problem tends to penalize SCCG more than any other method whenever n is relatively large. The largest n (n=4) in our sample of test problems happened to be associated with functions 5 and 6 which, despite this heavier penalty, exhibited better performance for SCCG. As a matter of fact, n function evaluations require more computing effort than one gradient evaluation, and this is another reason why we consider the performance of SCCG as promising.

Appendix

1: $100(x_2 - x_1^2)^2 + (1-x_1)^2$ [1]

2: $(x_2-x_1^2)^2 + (1-x_1)^2$ [2]

3: $(x_2 - x_1^2)^2 + 100(1-x_1)^2$ [2]

4: $100(x_2 - x_1^3)^2 + (1 - x_1)^2$ [2]

5: $100(x_2 - x_1^2)^2 + (1 - x_1)^2 + 90(x_4-x_3^2)^2$ [3]
$+ (1-x_3)^2 + 10.1 [(x_2-1)^2 + (x_4 - 1)^2]$
$+ 19.8 (x_2-1)(x_4-1)$

6: $(x_1 + 10x_2)^2 + 5(x_3-x_4)^2 + (x_2-2x_3)^4 + 10(x_1-x_4)^4$ [4]

7: $(x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$ [5]

---

[1] H.H. Rosenbrock, Computer J., 3: 175 (1960)

[2] B.F. Whitte and W.R. Holst, paper submitted at the 1964 Spring Joint Computer Conference, Washington D.C., (1964).

[3] C.F. Wood, Westinghouse Research Laboratories, (1968).

[4] M.J.D. Powell, Computer J., 5: 147 (1962).

[5] D.M. Himmelblau, Applied Nonlinear Programming, pp. 428. McGraw-Hill (1972).

References

[1] Broyden, C.G., "Quasi-Newton Methods and Their Application to Function Minimization", Math. Comp., 21, 1967 pp. 368-381.

[2] Broyden, C.G., "The Convergence of a Class of Double Rank Algorithms, Parts I and II," J. Inst. Math. Appl., 7, 1971, pp. 76-90, 222-236.

[3] Davidon, W.C., "Variable Metric Method for Minimization", Research and Development Report ANL-5990, U.S. Atomic Energy Commission, Argonne National Laboratories, 1959.

[4] Fletcher, R. "A New Approach to Variable Metric Algorithms", Comp. J., 13, 1970, pp. 317-322.

[5] Fletcher, R. and M.J.D. Powell, "A Rapidly Convergent Descent Method for Minimization", Comp.J., 6, 1963, pp. 163-168.

[6] Fletcher, R. and C.M. Reeves, "Function Minimization by Conjugate Gradients", Comp. J., 7, 1964, pp. 149-154.

[7] Goldfarb, D., "A Family of Variable Metric Methods Derived by Variational Means", Math. Comp. 24, 1970, pp. 23-26.

[8] Klessig, R. and E. Polak, "Efficient Implementation of the Polak-Rebiere Conjugate Gradient Algorithm," Siam J. Control 10, 1972, pp. 524-549.

[9] Lasdon, L.S., A.D. Waren, A. Jain and M. Ratner, "Design and Testing of a Generalized Reduced Gradient Code For Nonlinear Programming", Technical Report SoL 76-3, Department of Operations Research, Stanford University, Feb. 1976.

[10] Perry, Avinoam, "UCNLP-An Interactive Package of Programs for Unconstrained Nonlinear Optimization Purposes", A Working Paper, Nov. 1975, Revised May 1976.

[11] Perry, Avinoam, "A Modified Conjugate Gradient Algorithm" Forthcoming in Operations Research.

[12] Polak, E. and G. Rebière, "Note Sur La Convergence de Methodes de Directions Conjugees", Revue Francaise Inf. Rech, Oper., 16 RI 1969, pp. 35-43.

[13] Powell, M.J.D. "Recent Advances in Unconstrained Optimization", Math. Prog. 1, 1971, pp. 26-57.

[14] Saundres M.A. and B.A. Murtagh, "Nonlinear Programming for Large, Sparse Systems" Technical Report, Department of Operations Research, Stanford University, June 1976.

[15] Shanno, D.F., "Conditioning of Quasi-Newton Methods for Function Minimization", Math. Comp. 24, 1970, pp. 647-656.