Scaled BFGS preconditioned conjugate gradient algorithm for unconstrained optimization

Neculai Andrei[1]
Research Institute for Informatics, Center for Advanced Modeling and Optimization, 8-10, Averescu Avenue, Bucharest 1, Romania.

Abstract. This letter presents a scaled memoryless BFGS preconditioned conjugate gradient algorithm for solving unconstrained optimization problems. The basic idea is to combine the scaled memoryless BFGS method and the preconditioning technique in the frame of the conjugate gradient method. The preconditioner, which is also a scaled memoryless BFGS matrix, is reset when the Powell restart criterion holds. The parameter scaling the gradient is selected as spectral gradient. Computational results for a set consisting of 750 test unconstrained optimization problems, show that this new scaled conjugate gradient algorithm substantially outperforms known conjugate gradient methods as: the spectral conjugate gradient SCG by Birgin and Martínez [1] and the (classical) conjugate gradient by Polak and Ribière [9], but subject to CPU time metric it is outperformed by L-BFGS [6,7].

MSC: 49M07, 49M10, 90C06, 65K
Keywords: Unconstrained optimization, conjugate gradient method. BFGS preconditioning

1. Introduction

In this paper we consider the following unconstrained optimization problem:

$$\min f(x) \tag{1}$$

where $f:R^n \to R$ is continuously differentiable and its gradient is available. We are interested in elaborating an algorithm for solving large-scale cases for which the Hessian of f is either not available or requires a large amount of storage and computational costs.

The algorithm uses the conjugate gradient direction where the famous parameter β_k is obtained by equating the conjugate gradient direction with the direction corresponding to the Newton method. The method is an extension of the spectral conjugate gradient (SCG) by Birgin and Martínez [1] or of a variant of the conjugate gradient algorithm by Dai and Liao [3] (for $t=1$) to overcome the lack of positive definiteness of the matrix defining their search direction.

The paper is organized as follows: In section 2 we present the method. Section 3 is dedicated to the SCALCG algorithm. The algorithm performs two types of steps: a normal one in which a double quasi-Newton updating scheme is used and a restart one where the current information is used to define the search direction. In section 4 we present comparisons of SCALCG versus SCG conjugate gradient method by Birgin and Martínez [1], the Polak-Ribière conjugate gradient algorithm [9] and L-BFGS [6,7] on a set of 750 unconstrained optimization problems.

2. The method

The algorithm generates a sequence x_k of approximations to the minimum x^* of f, in which

$$x_{k+1} = x_k + \alpha_k d_k, \tag{2}$$

$$d_{k+1} = -\theta_{k+1} g_{k+1} + \beta_k s_k, \tag{3}$$

where $g_k = \nabla f(x_k)$, α_k is selected to minimize $f(x)$ along the search direction d_k, β_k is a scalar parameter and θ_{k+1} is a parameter to be determined. The iterative process is initialized with an initial point x_0 and $d_0 = -g_0$.

[1] *E-mail: nandrei@ici.ro*

Observe that if $\theta_{k+1}=1,$ then we get the classical conjugate gradient algorithms according to the value of the scalar parameter β_k. On the other hand, if $\beta_k=0,$ then we get another class of algorithms according to the selection of the parameter θ_{k+1}. There are two possibilities for θ_{k+1}: a positive scalar or a positive definite matrix. If $\theta_{k+1}=1$ we have the steepest descent algorithm. If $\theta_{k+1}=\nabla^2 f(x_{k+1})^{-1},$ or an approximation of it, then we get the Newton or the quasi-Newton algorithms, respectively. Therefore, we see that in the general case, when $\theta_{k+1}\neq 0$ is selected in a quasi-Newton manner, and $\beta_k\neq 0,$ (3) represents a combination between the quasi-Newton and the conjugate gradient methods.

To determine β_k consider the following procedure. As we know, the Newton direction for solving (1) is given by $d_{k+1}=-\nabla^2 f(x_{k+1})^{-1}g_{k+1}.$ Therefore, from the equality

$$-\nabla^2 f(x_{k+1})^{-1}g_{k+1}=-\theta_{k+1}g_{k+1}+\beta_k s_k,$$

we get:

$$\beta_k=\frac{s_k^T\nabla^2 f(x_{k+1})\theta_{k+1}g_{k+1}-s_k^T g_{k+1}}{s_k^T\nabla^2 f(x_{k+1})s_k}. \tag{4}$$

Using the Taylor development, after some algebra we obtain:

$$\beta_k=\frac{(\theta_{k+1}y_k-s_k)^T g_{k+1}}{y_k^T s_k}, \tag{5}$$

where $s_k=x_{k+1}-x_k$ and $y_k=g_{k+1}-g_k$. Birgin and Martínez [1] arived at the same formula for β_k, but using a geometric interpretatioin for quadratic function minimization. The direction corresponding to β_k given in (5) is as follows:

$$d_{k+1}=-\theta_{k+1}g_{k+1}+\frac{(\theta_{k+1}y_k-s_k)^T g_{k+1}}{y_k^T s_k}s_k. \tag{6}$$

The following particularizations are obvious. If $\theta_{k+1}=1,$ then (6) is the direction considered by Perry [8]. At the same time we see that (6) is the direction given by Dai and Liao [3] for $t=1,$ obtained this time by an interpretation of the conjugacy condition. Additionally, if $s_j^T g_{j+1}=0,\ \ j=0,1,\ldots,k,$ then from (6) we get:

$$d_{k+1}=-\theta_{k+1}g_{k+1}+\frac{\theta_{k+1}y_k^T g_{k+1}}{\alpha_k\theta_k g_k^T g_k}s_k, \tag{7}$$

which is the direction corresponding to a *generalization of the Polak and Ribière* formula. Of course, if $\theta_{k+1}=\theta_k=1$ in (7), we get the *classical Polak and Ribière* formula [9]. If $s_j^T g_{j+1}=0,\ \ j=0,1,\ldots,k,$ and additionally the successive gradients are orthogonal, then:

$$d_{k+1}=-\theta_{k+1}g_{k+1}+\frac{\theta_{k+1}g_{k+1}^T g_{k+1}}{\alpha_k\theta_k g_k^T g_k}s_k, \tag{8}$$

which is the direction corresponding to a *generalization of the Fletcher and Reeves* formula [4].

Now, using the same methodology as considered by Shanno [11] we get the following direction d_{k+1}:

$$d_{k+1}=-\theta_{k+1}g_{k+1}+\theta_{k+1}\left(\frac{g_{k+1}^T s_k}{y_k^T s_k}\right)y_k-\left[\left(1+\theta_{k+1}\frac{y_k^T y_k}{y_k^T s_k}\right)\frac{g_{k+1}^T s_k}{y_k^T s_k}-\theta_{k+1}\frac{g_{k+1}^T y_k}{y_k^T s_k}\right]s_k, \tag{9}$$

involving only 4 scalar products. Again observe that if $g_{k+1}^T s_k=0,$ then (9) reduces to:

$$d_{k+1}=-\theta_{k+1}g_{k+1}+\theta_{k+1}\frac{g_{k+1}^T y_k}{y_k^T s_k}s_k. \tag{10}$$

Thus, in this case, the effect is simply one of multiplying the Hestenes and Stiefel [5] search direction by a positive scalar.

In order to ensure the convergence of the algorithm (2), with d_{k+1} given by (9), we need to constrain the choice of α_k. We consider line searches that satisfy the Wolfe conditions [13,14]:

$$f(x_k+\alpha_k d_k)-f(x_k)\le \sigma_1\alpha_k g_k^T d_k, \tag{11}$$

$$\nabla f(x_k+\alpha_k d_k)^T d_k \ge \sigma_2 g_k^T d_k, \tag{12}$$

where $0<\sigma_1\le\sigma_2<1.$

Theorem 1. *Suppose that α_k in (2) satisfies the Wolfe conditions (11) and (12), then the direction d_{k+1} given by (9) is a descent direction.*

Proof: Since $d_0=-g_0,$ we have $g_0^T d_0=-\|g_0\|^2\le 0.$ Multiplying (9) by $g_{k+1}^T,$ we have

$$g_{k+1}^T d_{k+1}=\frac{1}{(y_k^T s_k)^2}\Big[-\theta_{k+1}\|g_{k+1}\|^2(y_k^T s_k)^2+2\theta_{k+1}(g_{k+1}^T y_k)(g_{k+1}^T s_k)(y_k^T s_k)$$
$$-(g_{k+1}^T s_k)^2(y_k^T s_k)-\theta_{k+1}(y_k^T y_k)(g_{k+1}^T s_k)^2\Big].$$

Applying the inequality $u^T v\le\frac{1}{2}(\|u\|^2+\|v\|^2)$ to the second term of the right hand side of the above equality, with $u=(s_k^T y_k)g_{k+1}$ and $v=(g_{k+1}^T s_k)y_k$ we get:

$$g_{k+1}^T d_{k+1}\le -\frac{(g_{k+1}^T s_k)^2}{y_k^T s_k}. \tag{13}$$

But, by Wolfe condition (12), $y_k^T s_k>0.$ Therefore, $g_{k+1}^T d_{k+1}<0$ for every $k=0,1,\ldots$ ■

Observe that the second Wolfe condition (12) is crucial for the descent character of direction (9). Besides, we see that the estimation (13) is independent of the parameter $\theta_{k+1}.$

Usually, all conjugate gradient algorithms are periodically restarted. The Powell restarting procedure [10], used in this algorithm, is to test if there is very little orthogonality left between the current gradient and the previous one. At step r when:

$$\left|g_{r+1}^T g_r\right|\ge 0.2\|g_{r+1}\|^2, \tag{14}$$

we restart the algorithm using the direction given by (9).

At step r when (14) is satisfied the direction is computed as in (9). For $k\ge r+1,$ we consider the same philosophy used by Shanno [11,12], i.e. that of modifying the gradient g_{k+1} with a positive definite matrix which best estimates the inverse Hessian without any additional storage requirements. Therefore, the direction $d_{k+1},$ for $k\ge r+1,$ is computed using a double update scheme as:

$$v=\theta_{r+1}g_{k+1}-\theta_{r+1}\left(\frac{g_{k+1}^T s_r}{y_r^T s_r}\right)y_r$$
$$+\left[\left(1+\theta_{r+1}\frac{y_r^T y_r}{y_r^T s_r}\right)\frac{g_{k+1}^T s_r}{y_r^T s_r}-\theta_{r+1}\frac{g_{k+1}^T y_r}{y_r^T s_r}\right]s_r, \tag{15}$$

and

$$w=\theta_{r+1}y_k-\theta_{r+1}\left(\frac{y_k^T s_r}{y_r^T s_r}\right)y_r$$
$$+\left[\left(1+\theta_{r+1}\frac{y_r^T y_r}{y_r^T s_r}\right)\frac{y_k^T s_r}{y_r^T s_r}-\theta_{r+1}\frac{y_k^T y_r}{y_r^T s_r}\right]s_r, \tag{16}$$

involving 6 scalar products. With these the direction d_{k+1}, at any nonrestart step, can be computed as:

$$d_{k+1}=-v+\frac{(g_{k+1}^T s_k)w+(g_{k+1}^T w)s_k}{y_k^T s_k}-\left(1+\frac{y_k^T w}{y_k^T s_k}\right)\frac{g_{k+1}^T s_k}{y_k^T s_k}s_k, \tag{17}$$

involving only 4 scalar products. It is useful to note that $y_k^T s_k > 0$ is sufficient to ensure that the direction d_{k+1} given by (17) is well defined and it is always a descent direction.

In this paper, motivated by the success of the spectral gradient used by Birgin and Martínez [1] in their SCG algorithm, we consider the spectral gradient choice of θ_{k+1} as:

$$\theta_{k+1} = \frac{s_k^T s_k}{y_k^T s_k}. \tag{18}$$

The parameter θ_{k+1} given in (18) is the inverse of the Rayleigh quotient, and is always well defined and positive, since (12) implies that $y_k^T s_k > 0$.

3. SCALCG algorithm

Step 1. Initialization. Select $x_0 \in R^n$, and the parameters $0 < \sigma_1 \le \sigma_2 < 1$. Compute $f(x_0)$ and $g_0 = \nabla f(x_0)$. Set $d_0 = -g_0$ and $\alpha_0 = 1/\|g_0\|$. Set $k = 0$.

Step 2. Line search. Compute α_k satisfying the Wolfe conditions (11) and (12). Update the variables $x_{k+1} = x_k + \alpha_k d_k$. Compute $f(x_{k+1}), g_{k+1}$ and $s_k = x_{k+1} - x_k$, $y_k = g_{k+1} - g_k$.

Step 3. Test for continuation of iterations. If this test is satisfied the iterations are stopped, else set $k = k + 1$.

Step 4. Scaling factor computation. Compute θ_k using (18).

Step 5. Restart direction. Compute the (restart) direction d_k as in (9).

Step 6. Line search. Compute the initial guess: $\alpha_k = \alpha_{k-1}\|d_{k-1}\|_2 / \|d_k\|_2$. Using this initialization compute α_k satisfying the Wolfe conditions. Update the variables $x_{k+1} = x_k + \alpha_k d_k$. Compute $f(x_{k+1})$, g_{k+1} and $s_k = x_{k+1} - x_k$, $y_k = g_{k+1} - g_k$.

Step 7. Store: $\theta = \theta_k$, $s = s_k$ and $y = y_k$.

Step 8. Test for continuation of iterations. If this test is satisfied the iterations are stopped, else set $k = k + 1$.

Step 9. Restart. If the Powell restart criterion (14) is satisfied, then go to step 4 (a restart step); otherwise continue with step 10 (a normal step).

Step 10. Normal direction. Compute the direction d_k as in (17), where v and w are computed as in (15) and (16) with saved values θ, s and y.

Step 11. Line search. Compute the initial guess: $\alpha_k = \alpha_{k-1}\|d_{k-1}\|_2 / \|d_k\|_2$. Using this initialization compute α_k satisfying the Wolfe conditions. Update the variables $x_{k+1} = x_k + \alpha_k d_k$. Compute $f(x_{k+1})$, g_{k+1} and $s_k = x_{k+1} - x_k$, $y_k = g_{k+1} - g_k$.

Step 12. Test for continuation of iterations. If this test is satisfied the iterations are stopped, else set $k = k + 1$ and go to step 9. ■

Assuming that f is strongly convex and Lipschitz continuous on the level set $L_0 = \{x \in R^n : f(x) \le f(x_0)\}$, then it is easy to prove that *if at every step of the conjugate gradient (2) with d_{k+1} given by (9) and the step length α_k selected to satisfy the Wolfe conditions (11) and (12), then either $g_k = 0$ for some k, or $\lim_{k\to\infty} g_k = 0$.*

For general functions the convergence of the algorithm is coming from theorem 1 and the restart procedure. In fact SCALCG is a scaled memoryless BFGS preconditioned algorithm where the scaling factor is the inverse of a scalar approximation of the Hessian. Although a global convergence result has not been established for SCALCG, recall that for the Perry/Shanno scheme, the iterates either converge to a stationary point or the iterates cycle.

4. Computational results and comparisons

In this section we present the performance of a Fortran implementation of the *SCALCG* on a set of 750 test unconstrained optimization problems. We compare the performance of SCALCG (Powell restart) with *the best spectral conjugate gradient algorithm, SCG* (betatype=1,Perry-M1, spectral gradient and Powell restart), by Birgin and Martínez [1], the *classical Polak-Ribière* (PRP) conjugate gradient algorithm with Powell restart, and L-BFGS by Liu and Nocedal [6]. The SCALCG and Polak-Ribière codes are authored by Andrei, while the SCG is co-authored by Birgin and Martínez. L-BFGS is authored by Nocedal [7]. In order to compare SCALCG to SCG we manufactured a new SCG code by introducing a sequence of code implementing the Powell restart and the same stopping criterion used in all these algorithms. All codes are written in double precision Fortran and compiled with f77 (default compiler settings) on Intel Pentium 4, 1.8GHz workstation. The test problems are the unconstrained problems in the CUTE [2] library, along with other large-scale optimization test problems. We selected 75 large-scale unconstrained optimization test problems in extended or generalized form. For each test function we have considered 10 numerical experiments with number of variables $n = 1000,2000,\ldots,10000$. All algorithms use exactly the same implementation of the Wolfe line search conditions with $\sigma_1 = 0.0001$ and $\sigma_2 = 0.9$. Concerning the stopping criterion to be used in steps 3, 8 and 12 of SCALCG, in our numerical experiments, in all algorithms, we have considered the following criterion:

$$\left\|\nabla f(x_k)\right\|_\infty \le \varepsilon_g, \text{ where } \varepsilon_g = 10^{-6}. \tag{19}$$

Let f_i^{SCALCG}, f_i^{SCG} be the optimal functional value found by SCALCG and SCG algorithms, for test problem $i = 1,\ldots,750$, respectively. We say that, in the particular problem i, the performamce of SCALCG was better than the performance of SCG if: $\left|f_i^{SCALCG} - f_i^{SCG}\right| < 10^{-3}$ and the number of iterations, or the number of function-gradient evaluations, or the CPU time of SCALCG was less than the number of iterations, or the number of function-gradient evaluations, or the CPU time corresponding to SCG respectively. The same criterion is used when SCALCG is compared to PRP, or L-BFGS. Tables 1 and 2 present the comparisons between SCALCG and SCG, and SCALCG and PRP, respectively.

Table 1. Comparison: SCALCG versus SCG.

	SCALCG	SCG	=
#iter	486	98	89
#fg	425	128	120
CPU	592	43	38

Table 2. Comparison: SCALCG versus PRP.

	SCALCG	PRP	=
#iter	497	93	81
#fg	415	134	122
CPU	580	54	37

In these Tables we find the number of problems, out 750, for which an algorithm achieved the minimum number of iterations (#iter), or the minimum number of function-gradient evaluations (#fg) or the minimum CPU time. For example when comparing SCALCG and SCG (Table 1), subject to the number of iterations, SCALCG was better in 486 problems (i.e. it achieved the minimum number of iterations in 486 problems), SCG was better in 98 problems, and they had the same number of iterations in 89 problems, etc. From these Tables we see that the top performer is SCALCG. Since these codes uses the same line search, they differ in their choice of the search direction. Hence, SCALCG appears to generate the best search direction, on average. Next, in Tables 3 and 4 we compared SCALCG with L-BFGS using $m = 3$ and $m = 5$ correction pairs, respectively.

Table 3. Comparison: SCALCG versus L-BFGS.

$m = 3$	SCALCG	L-BFGS	=
#iter	497	148	21
#fg	654	12	0
CPU	189	348	129

Table 4. Comparison: SCALCG versus L-BFGS.

$m = 5$	SCALCG	L-BFGS	=
#iter	433	222	11
#fg	650	16	0
CPU	201	350	115

We observe that subject to the minimum number of iterations, and to the minimum number of function-gradient evaluations, SCALCG is more economical, but subject to the CPU time metric L-BFGS is clearly better.

5. Conclusion

We have presented a scaled memoryless BFGS preconditioned conjugate gradient algorithm, *SCALCG,* for solving large-scale unconstrained optimization problems. SCALCG can be considered as a modification of the best algorithm by Birgin and Martínez [1], which is mainly a scaled variant of Perry's [8], and of the Dai and Liao [3] $(t = 1)$, in order to overcome the lack of positive definiteness of the matrix defining the search direction. This modification takes advantage of the quasi-Newton BFGS updating formula firstly considered by Perry [8] and Shanno [11,12].

The numerical experiments suggest that the SCALCG algorithm should be considered as a top performer in the class of conjugate gradient methods. In these tests, subject to the CPU time metric, the SCALCG did not outperform L-BFGS, showing that L-BFGS performs inexpensive iterations, with poorer curvature information - process that can become slow on some (ill-conditioned) problems like: CLIFF, WOODS, NONDIA, DQDRTIC, LIARWHD, DENSCHNB from CUTE library.

References

[1] E. Birgin and J.M. Martínez, "*A spectral conjugate gradient method for unconstrained optimization*", Applied Math. and Optimization, 43, pp.117-128, 2001

[2] I. Bongartz, A.R. Conn, N.I.M. Gould and P.L. Toint, "*CUTE: constrained and unconstrained testing environments*", ACM Trans. Math. Software, 21, pp.123-160, 1995.

[3] Y.H. Dai and L.Z. Liao, "*New conjugate conditions and related nonlinear conjugate gradient methods*", Appl. Math. Optim., vol. 43 pp.87-101, 2001.

[4] R. Fletcher and C.M. Reeves, "*Function minimization by conjugate gradients*", Comput. J. 7, pp. 149-154, 1964.

[5] M.R. Hestenes and E. Stiefel, "*Methods of conjugate gradients for solving linear systems*", J. Research Nat. Bur. Standards Sec. B. 48, pp. 409-436, 1952.

[6] D. Liu and J. Nocedal, "*On the limited memory BFGS method for large scale optimization*", Mathematical Programming B 45, 503-528, 1989.

[7] J. Nocedal, http://www.ece.northwestern.edu/~nocedal/lbfgs.html

[8] J.M. Perry, "*A class of conjugateradient algorithms with a two step variable metric memory*", Discussion paper 269, Center for Mathematical Studies in Economics and Management Science, Northwestern University, 1977.

[9] E. Polak and G. Ribière, "*Note sur la convergence de methods de directions conjugres*", Revue Francaise Informat. Reserche Opérationnelle 16, pp. 35-43, 1969.

[10] M.J.D. Powell, "*Restart procedures for the conjugate gradient method*", Math. Programming, 12, pp.241-254, 1977.

[11] D.F. Shanno, "*Conjugate gradient methods with inexact searches*", Mathematics of Operations Research, vol. 3, pp.244-256, 1978.

[12] D.F. Shanno, "*On the convergence of a new conjugate gradient algorithm*", SIAM J. Numer. Anal. vol. 15, pp.1247-1257, 1978.

[13] P. Wolfe, "*Convergence conditions for ascent methods*", SIAM Rev., 11, pp.226-235, 1969.

[14] P. Wolfe, "*Convergence conditions for ascent methods II: some corrections*", SIAM Rev. 13, pp.185-188, 1971.

June 2, 2006