Report on Numerical Optimization Coding Assignment
Nonlinear Conjugate Gradient Methods

Hongyi Zhang
Yuanpei College, Peking University
hongyizhang@pku.edu.cn

May 23, 2018

1 Overview

Given a smooth function $f : \mathbb{R}^n \to \mathbb{R}$, we use a series of nonlinear conjugate gradient (CG) methods to solve the unconstrained optimization problem

$$\min_{x \in \mathbb{R}^n} f(x),$$

including FR (Fletcher-Reeves), PRP (Polak-Ribire-Polyak), PRP+ (a variant of PRP), HS (Hestenes-Stiefel), CD (Conjugate Descent) and DY (Dai-Yuan). For comparison, BB (Barzilai-Borwein) gradient method is implemented as well, although it has a totally different nature from CG methods.

1.1 Nonlinear conjugate gradient methods

Nonlinear conjugate gradient methods fall into the category of line search methods. All of these methods share an iterative scheme

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

where the step length $\alpha_k > 0$ is obtained by a line search, and the directions d_k are generated through

$$d_{k+1} = -g_{k+1} + \beta_k d_k, \quad d_0 = -g_0,$$

with g standing for the gradient of the objective. Different CG methods provide different choices of β_k, and the following is an overview of all CG methods implemented [2].

- FR: $\beta_k^{FR} = \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k}$
- PRP: $\beta_k^{PRP} = \frac{g_{k+1}^T y_k}{g_k^T g_k}$
- PRP+: $\beta_k^{PRP+} = \max\{\frac{g_{k+1}^T y_k}{g_k^T g_k}, 0\}$
- HS: $\beta_k^{HS} = \frac{g_{k+1}^T y_k}{d_k^T y_k}$
- CD: $\beta_k^{CD} = \frac{g_{k+1}^T g_{k+1}}{-g_k^T d_k}$
- DY: $\beta_k^{DY} = \frac{g_{k+1}^T g_{k+1}}{d_k^T y_k}$

We employ an inexact line search routine, and impose strong Wolfe conditions on the step length α. To be specific, for the current iteration x_k and descent direction d_k, we find an α that satisfies

$$\begin{cases} \phi(\alpha) \leq \phi(0) + \rho\phi'(0)\alpha, \\ |\phi'(\alpha)| \leq -\sigma\phi'(0), \end{cases}$$

where $\phi(\alpha) = f(x_k + \alpha d_k)$. Parameter ρ for sufficient decrease is set to 10^{-4}, and the parameter σ for curvature condition is set to 0.1 for a tight search. The iteration procedure stops when both $|f_k - f_{k-1}| < \epsilon$ and $\|g_k\| < \epsilon$ are met, and we pick $\epsilon = 10^{-8}$.

1.2 BB (Barzilai-Borwein) Method

BB (Barzilai-Borwein) gradient method, in a form similar to steepest descent, is defined as $x_{k+1} = x_k - \alpha_k g_k$, but employs a particular step length

$$\alpha_k = \frac{s_{k-1}^T s_{k-1}}{s_{k-1}^T y_{k-1}}.$$

This method is preferred in large scale unconstrained optimization settings, since it requires no line search, and is provably R-superlinearly convergent for the quadratic case. For general objective functions, a globalization strategy based on non-monotonic line search is combined with BB method. Non-monotonic line search works by enforcing

$$f(x_{k+1}) \leq \max_{0 \leq j \leq M} f(x_{k-j}) + \rho g_k^T (x_{k+1} - x_k),$$

which, contrary to typical Armijo conditions, does not insist on descent at every iteration. The whole procedure [3] is shown in Algorithm 1. Clipping of λ_k is necessary to ensure the global convergence of the algorithm, and we perform clipping by

$$\lambda_k \leftarrow \max\left\{1, \min\left\{10^5, \|g_k\|^{-1}\right\}\right\}.$$

Parameters are fixed as $\lambda_0 = 1, \delta = 10^{-10}, \rho = 10^{-4}, \mu = 0.5$ and $M = 10$.

Algorithm 1 Barzilai-Borwein method

```
Input: Objective function $f(x)$, initial point $x_0, \lambda_0$, parameters $\delta, \rho, \mu, M$
Output: Optimal solution $x^* = \arg\min f(x)$
  Initialize $k = 0$
  while stopping criterion is not satisfied do
      if $\lambda_k < \delta$ or $\lambda_k > 1/\delta$ then
          Clip $\lambda_k$ to keep $\{1/\lambda_k\}$ bounded
      $\alpha_k = 1/\lambda_k$
      while $f(x_k - \alpha_k g_k) > \max_{0 \le j \le \min\{k,M\}}\{f_{k-j}\} - \rho g_k^T g_k \alpha_k$ do
          $\alpha_k = \mu\alpha_k$
      $x_{k+1} = x_k - \alpha_k g_k$
      $\lambda_{k+1} = -(g_k^T y_k)/(\alpha_k g_k^T g_k)$
      $k = k + 1$
```

2 Problem 1: Generalized Powell singular function

The first objective function we are going to tackle is the generalized Powell singular function (GENPOWSG), defined as

$$f(x) = \sum_{i=1}^{n/2-1} [(x_{2i-1} + 10x_{2i})^2 + 5(x_{2i+1} - x_{2i+2})^2 + (x_{2i} - 2x_{2i+1})^4 + 10(x_{2i-1} - x_{2i+2})^4],$$

where $n \geq 4$ in an even integer. When $n = 4$, it reduces to the original Powell singular function. This type of generalization increases the non-separability of the variables, and therefore adds to the difficulty of the problem. The initial point $x_0 = (3, -1, \cdots, 3, -1)^T$, and the global optimum $f(x^*) = 0$ is obtained at $x^* = (0, 0, \cdots, 0)^T$.

Table 1 gives the results on how the methods perform on this function. We vary the problem scale among $n = 1000, 2000, 5000$ and 10000. Notations in the table are: the objective value $f(x^*)$, the magnitude of gradient $-\lg \|g^*\|$, the number of iterations N_{iter} and the number of function evaluations N_{eval}. We set the maximum number of iterations to be 10000, and $N_{iter} = \max$ if the method does not meet the stopping criterion after 10000 iterations. N_{eval} counts evaluations of both f and g.

The first six rows of Table 1 offers a brief view over the performances of nonlinear conjugate gradient methods. The problem is relatively easy to solve, and it is indeed hard to tell which methods are superior to others. HS and DY methods enjoy fast convergence in all four cases. FR and CD methods can show similar performance from time to time. PRP and PRP+ methods are almost identical, implying that most of the β_k are non-negative,

Table 1: Result comparison for the generalized Powell singular function

(a) $n = 1000, 2000$

Method	$n = 1000$				$n = 2000$			
	$f(x^*)$	$-\lg \|g^*\|$	N_{iter}	N_{eval}	$f(x^*)$	$-\lg \|g^*\|$	N_{iter}	N_{eval}
FR	6.546e-13	8.3	355	2704	8.714e-13	8.0	145	1072
PRP	2.951e-13	8.4	422	3275	1.677e-13	8.5	474	3538
PRP+	2.951e-13	8.4	422	3275	1.677e-13	8.5	474	3538
HS	4.268e-14	8.0	184	1440	6.859e-13	8.2	131	944
CD	1.031e-12	8.2	362	2825	7.554e-15	8.4	163	1198
DY	5.957e-13	8.1	188	1440	8.664e-13	8.1	160	1165
BB	6.669e-12	6.1	max	28102	5.097e-12	6.3	max	28295
SD	4.117e-07	3.9	max	96470	4.126e-07	3.8	max	96500

(b) $n = 5000, 10000$

Method	$n = 5000$				$n = 10000$			
	$f(x^*)$	$-\lg \|g^*\|$	N_{iter}	N_{eval}	$f(x^*)$	$-\lg \|g^*\|$	N_{iter}	N_{eval}
FR	7.965e-13	8.1	152	1077	6.682e-13	8.3	398	2983
PRP	1.293e-13	8.4	536	4087	5.231e-13	8.1	437	3300
PRP+	1.293e-13	8.4	536	4087	5.231e-13	8.1	437	3300
HS	8.338e-13	8.1	161	1259	3.809e-14	8.1	167	1252
CD	3.175e-13	8.3	632	4914	2.257e-13	8.4	469	3564
DY	1.920e-13	8.4	191	1435	2.055e-13	8.4	193	1493
BB	1.641e-12	8.0	6131	17246	1.988e-12	8.0	7591	21487
SD	4.027e-07	3.9	max	96553	4.047e-07	3.9	max	96539

and the correction $\beta_k^{PRP+} = \max\{\beta_k^{PRP}, 0\}$ is not active for most of the time.

The last two rows of Table 1 gives how BB method works for this problem. The results of Steepest Descent (SD) are also presented for comparison. As one can see, BB method actually requires far more iterations to converge compared to CG methods, but is definitely a remarkable improvement over SD method — a lot fewer iterations and function evaluations are needed for BB method than SD method.

One interesting phenomenon is, the difficulty of the problem does not seem to change monotonically as the dimension n grows, and BB method might perform better for larger scale problems. Further investigations show that BB method and CG methods converge to the same point for $n = 5000$, but BB method converges to a local minimum different from CG methods for $n = 1000$. This might be an indicator that BB method is sensitive to some numerical issues.

3 Problem 2: Generalized SINEVAL function

The second objective function we are coping with is the generalized SINEVAL function (GENSIN), given by

$$f(x) = \sum_{i=1}^{n-1} \left[c_1 (x_{i+1} - \sin x_i)^2 + c_2 x_i^2 \right], \quad x \in \mathbb{R}^n,$$

where $c_1 = 10^4$ and $c_2 = 1/4$. The initial point $x_0 = (4.712389, -1, -1, \cdots, -1)^T$, and the optimal point is apparently $x^* = (0, 0, \cdots, 0)^T$, corresponding to the minimum $f(x^*) = 0$.

Table 2 compares the performances of different optimization methods. To avoid unpleasant numerical headaches, we rescale the objective function by a factor of $\gamma = 10^{-5}$, and optimize $\tilde{f}(x) = \gamma \cdot f(x)$ instead of working on the original problem directly. Consequently, we stop the iteration when $|\tilde{f}_k - \tilde{f}_{k-1}| < \gamma\epsilon$. The gradient condition $\|\tilde{g}_k\| < \gamma\epsilon$ is extremely hard to satisfy, and we discard this criterion for this problem. Note that the objective value in Table 2 is replaced by $f(x^*) - 2.455$ — we find that all CG methods produce an optimal value around 2.455, and thus this amount is subtracted from $f(x^*)$ to present the more subtle differences in these function values.

All conjugate gradient methods show no significant differences with respect to this problem, and the numerical behavior does not change much when the dimension n grows large. In contrast, BB and SD methods only reach solutions of an awfully low precision. And their performances drop when more variables are involved. But still, BB method outperforms SD method over the accuracy by almost three orders of magnitude.

It is noteworthy that $f(x^*)$ output by the algorithms are all far from the true optimum 0. The iteration procedure stops at

$$x_N = (3.12573, 0.01577, 0.01570, 0.01562, \cdots)^T,$$

which we believe is an approximation of

$$\bar{x}_N = \left(\frac{199}{200}\pi, \frac{\pi}{200}, \sin\frac{\pi}{200}, \sin\sin\frac{\pi}{200}, \cdots \right)^T.$$

We plot the objective function $\hat{f}(x)$ for two variables in Figure 1. Marked red, green and blue are the initial point x_0, the optimal point x^* and the algorithm output x_N respectively.

Table 2: Result comparison for the generalized SINEVAL function

(a) $n = 1000, 2000$

Method	$n = 1000$				$n = 2000$			
	$f(x^*) - 2.455$	$-\lg \|g^*\|$	N_{iter}	N_{eval}	$f(x^*) - 2.455$	$-\lg \|g^*\|$	N_{iter}	N_{eval}
FR	4.371e-05	1.7	1754	8846	4.467e-05	1.7	1907	9603
PRP	4.334e-05	1.7	1832	9246	4.472e-05	1.7	2021	10169
PRP+	4.334e-05	1.7	1832	9246	4.472e-05	1.7	2021	10169
HS	4.326e-05	1.7	1849	9327	4.470e-05	1.7	1987	9995
CD	4.335e-05	1.8	1520	7737	4.463e-05	1.7	1901	9573
DY	4.320e-05	1.7	1828	9220	4.447e-05	1.7	1798	9058
BB	5.644e-02	0.6	max	26823	3.424e-02	0.2	max	26727
SD	1.106e+01	-0.9	max	60038	1.897e+01	-1.0	max	60038

(b) $n = 5000, 10000$

Method	$n = 5000$				$n = 10000$			
	$f(x^*) - 2.455$	$-\lg \|g^*\|$	N_{iter}	N_{eval}	$f(x^*) - 2.455$	$-\lg \|g^*\|$	N_{iter}	N_{eval}
FR	4.459e-05	1.7	1942	9752	4.457e-05	1.7	1980	9966
PRP	4.466e-05	1.7	2048	10280	4.460e-05	1.7	2001	10065
PRP+	4.466e-05	1.7	2048	10280	4.460e-05	1.7	2001	10065
HS	4.465e-05	1.7	2019	10137	4.465e-05	1.7	2115	10637
CD	4.461e-05	1.7	1943	9755	4.454e-05	1.7	1982	9976
DY	4.461e-05	1.7	1934	9710	4.461e-05	1.7	2003	10073
BB	6.246e-02	-0.4	max	26845	4.773e-01	-0.8	max	26815
SD	4.282e+01	-1.2	max	60039	8.294e+01	-1.4	max	60057

Table 3: Impact of rescaling factor ($n = 1000, \epsilon = 10^{-8}$)

		$f(x^*) - 2.455$	$-\lg \|g^*\|$	N_{iter}	N_{eval}
FR	$\gamma = 10^{-3}$	5.641e-05	1.8	2522	23927
	$\gamma = 10^{-2}$	2.658e-04	1.6	3069	39020
	$\gamma = 10^{-1}$	3.055e-03	0.1	8489	110008
	$\gamma = 10^{0}$	2.018e-01	-0.3	1997	25980
PRP	$\gamma = 10^{-3}$	1.308e-04	1.7	2776	26370
	$\gamma = 10^{-2}$	3.458e-04	1.5	3401	43443
	$\gamma = 10^{-1}$	1.746e+00	-1.3	max	129968
	$\gamma = 10^{0}$	4.593e+02	-2.8	71	925

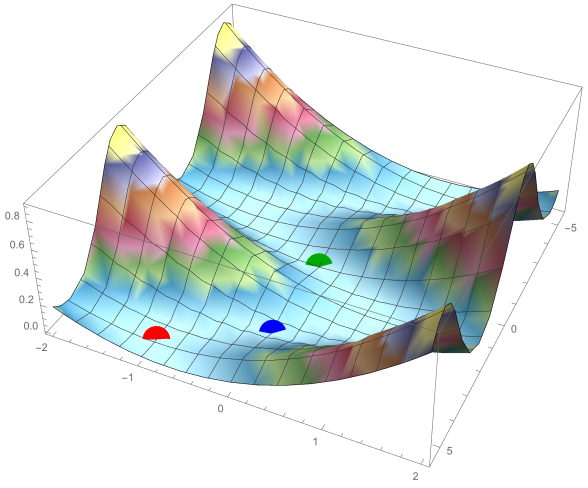

Figure 1: Plot of the generalized SINEVAL function ($n = 2$)

The iteration starts at the red point, winds along the valley towards the green point, but terminates halfway at the blue point. We estimate

$$|\hat{f}_{k+1} - \hat{f}_k| \approx \alpha_k \|\hat{g}_k^T \hat{d}_k\| = \gamma^2 \alpha_k \|g_k^T d_k\| \approx \gamma^2 |f_{k+1} - f_k|,$$

which implies that the rescaling process makes it easier for the criterion $|f_{k+1} - f_k| < \epsilon$ to be satisfied, and therefore the iteration stops before reaching the optimum. To solve this problem, it is possible to change either the rescaling factor γ or the required precision ϵ. Experiments show that it is not easy to escape from x_N even if we set ϵ to be unreasonably small. And the impact of rescaling factor γ is displayed in Table 3.

From Table 3, we observe that smaller factor helps achieve faster convergence and higher accuracy. When $\gamma = 1$, namely for the original problem with no rescaling, line search usually fails due to large gradients, step lengths have to be unbearably small, and as a consequence, strong Wolfe conditions are less likely to be satisfied. The iteration thereupon ends up with rather unsatisfactory results.

4 Problem 3: FLETCHCR function

FLETCHCR function, adopted from the CUTEr test problem set [4], is defined as

$$f(x) = \sum_{i=1}^{n-1} c(x_{i+1} - x_i + 1 - x_i^2)^2,$$

Table 4: Result comparison for the FLETCHCR function

(a) $n = 1000, 2000$

Method	$n = 1000$				$n = 2000$			
	$f(x^*)$	$-\lg \|g^*\|$	N_{iter}	N_{eval}	$f(x^*)$	$-\lg \|g^*\|$	N_{iter}	N_{eval}
FR	7.083e-14	7.0	6381	46725	4.421e-11	7.0	4618	32316
PRP	1.071e-11	7.1	6129	43271	4.217e-11	7.0	14105	98915
PRP+	1.071e-11	7.1	6129	43271	4.217e-11	7.0	14105	98915
HS	3.831e-12	7.0	3849	31584	1.727e-11	7.0	6583	52168
CD	2.745e-12	7.0	5442	39452	5.110e-11	7.0	8353	59614
DY	1.553e-12	7.0	4869	37472	2.250e-11	7.0	7421	56520
BB	4.964e-01	7.0	22127	59267	2.979e+00	5.8	max	133990
SD	1.894e-09	5.9	max	301878	1.060e-07	4.9	max	396849

(b) $n = 5000, 10000$

Method	$n = 5000$				$n = 10000$			
	$f(x^*)$	$-\lg \|g^*\|$	N_{iter}	N_{eval}	$f(x^*)$	$-\lg \|g^*\|$	N_{iter}	N_{eval}
FR	2.899e-11	7.1	10355	72147	5.231e-10	7.0	20437	142255
PRP	6.253e-10	7.0	12219	83521	1.320e-09	7.0	26018	178604
PRP+	6.253e-10	7.0	12219	83521	1.320e-09	7.0	26018	178604
HS	1.591e-10	7.0	16287	129566	1.751e-10	7.1	32610	260119
CD	3.129e-10	7.0	16731	118241	1.969e-10	7.0	17007	115346
DY	2.336e-10	7.0	17652	132862	5.732e-10	7.0	27371	203782
BB	4.964e+00	4.7	max	132263	1.340e+01	5.1	max	129482
SD	6.306e-07	4.7	max	304766	7.872e-09	5.7	max	375721

where $c = 100$. Initial point is chosen as $x_0 = (0, \cdots, 0)^T$. This function owns a global optimum $f(x^*) = 0$, but has uncountable optimal solutions x^*, each characterized by a sequence $\{x_i\}_{i=1}^n$ following the recurrence relation

$$x_{i+1} = x_i^2 + x_i - 1, \quad 1 \le i \le n - 1. \tag{*}$$

Table 4 displays the results produced by different optimization algorithms. We relax the criterion to $\epsilon = 10^{-7}$ for this problem, and the maximum number of iterations is set to 50000. We rescale the problem by a factor of $\gamma = 10^{-3}$. From Table 4, we can conclude that different CG methods are all comparably effective, while negative gradient methods (BB and SD) are not as compelling. Note that this is the first problem we encounter where

Table 5: Impact of different restart schemes ($n = 1000, \epsilon = 10^{-7}$)

		$f(x^*)$	$-\lg \Vert g^* \Vert$	N_{iter}	$f(x^*)$	$-\lg \Vert g^* \Vert$	N_{iter}	
FR	No restart	7.523e+02	-1.9	max	2.842e-12	7.1	7758	PRP
	Periodic	3.227e+00	7.1	23108	7.984e-12	7.0	8003	
	Orthogonal	7.083e-14	7.0	6381	1.071e-11	7.1	6129	
CD	No restart	9.868e+02	-1.2	max	1.965e-13	7.0	5129	HS
	Periodic	2.922e-14	7.0	24059	2.879e-12	7.0	5006	
	Orthogonal	2.745e-12	7.0	5442	3.831e-12	7.0	3849	

SD method shows better performance than BB method. A typical optimal solution output by the algorithm takes the form of

$$x_N = (1, 1, \cdots, -1, -1)^T,$$

with components in between shifting from 1 to -1 in a steep logistic-like manner. It could be that 1 and -1 are the only two fixed points of the recurrence relation (*) that pushes the solution x_N towards ±1.

It is worth mentioning that the restart scheme helps a great deal with convergence of conjugate gradient methods. A simple periodic restart scheme sets $d_k = -g_k$, the negative gradient direction, for every n iterations. Powell [5] observed that, if g_k and d_k are almost orthogonal and the step length α_k is small for some iteration k, then a long sequence of unproductive iterations may follow for the FR method (but not for PRP method). Based on this, he suggested restart when two consecutive gradients are far from orthogonal, i.e.

$$|g_k^T g_{k-1}| \geq \nu \|g_k\|^2,$$

and we choose $\nu = 0.2$. Table 5 shows how different restart schemes can influence the effectiveness of CG methods. It can be concluded that FR, CD and DY methods (with $g_{k+1}^T g_{k+1}$ in the numerator of β_k) can suffer from the problem that Powell described, and it can be remedied greatly by restart with orthogonality tests. Periodic restart can also help, but only to a limited extent. On the contrary, PRP and HS methods (with $g_{k+1}^T y_k$ in the numerator) do not have this problem and can perform self-correction to restart in an adaptive manner.

References

[1] Wright, S. and Nocedal, J. (1999). Numerical optimization. Springer, 35(67-68), 7.
[2] Hager, W. and Zhang, H. (2006). A survey of nonlinear conjugate gradient methods. Pacific journal of Optimization, 2(1), 35-58.
[3] Raydan, M. (1997). The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem. SIAM Journal on Optimization, 7(1), 26-33.
[4] Andrei, N. (2008). An unconstrained optimization test functions collection. Advanced Modeling and Optimization, 10(1), 147-161.
[5] Powell, M. J. D. (1977). Restart procedures for the conjugate gradient method. Mathematical programming, 12(1), 241-254.