Nonlinear Optimization Project Report

Yan Li, Wenzhong Han

School of Mathematics, Georgia Institute of Technology

1 Problem 1

In this problem, we implement Steepest descent, Conjugate gradient (both Fletcher-Reeves and Polak-Ribiere variants), BFGS and DFP quasi-Newton, limited memory BFGS, to minimize the following functions:

$$f(x) = \sum_{i=1}^{n} f_i(x)^2$$

where $x = (x_1, \dots, x_n)$. For each problem: (a) the dimension n; (b) the nonlinear functions $f_i(x)$; (c) the starting point x_0; (d) the root and minimizer $x_\star$. The minimum value of $f(x)$ is zero in all cases.

We also note that in our implementation the line search criterion is chosen to satisfied Strong Wolfe Condtion, the line search code was due to Jorge J. More' and David J. Thuente[2]. Their Matlab implementation of a MINPACK line search algorithm is contained in file named cvsrch.m and cstep.m. The limited memory BFGS algorithm file contains a subroutine used to update the approximate inverse Hessian matrix, the algorithm was due to original paper by Jorge Nocedal and the implementation was due to an open Github repositoray of Guipeng Li[1] at Tsinghua University.

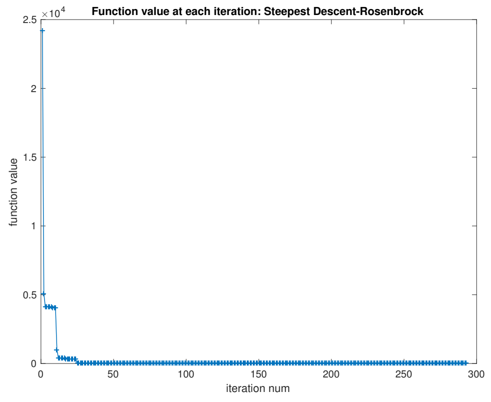
Extended Rosenbrock function

The Rosenbrock function is generated by, assuming n is multiple of 2, the following definition: For $i = 1, \dots, \frac{n}{2}$:

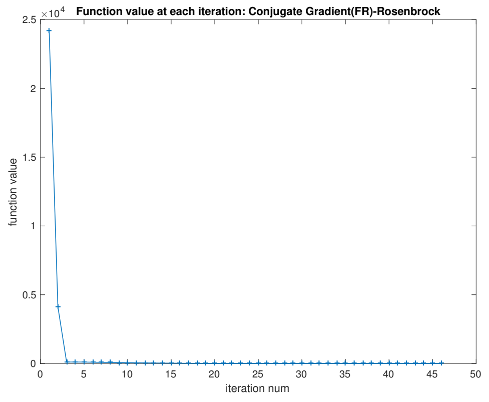
$$f_{2i-1}(x) = 10(x_{2i} - x_{2i-1}^2)$$
$$f_{2i}(x) = 1 - x_{2i-1}$$

We have $x_0 = (-1.2, 1, \dots, -1.2, 1)$ and $x_\star = (1, 1, \dots, 1, 1)$.
In our implementation of the algorithm, we always choose $n = 2000$ and consider termination criterion given by $\frac{f(x_k)-f_\star}{f(x_o)-f_\star} < \epsilon$ where $\epsilon = 10^{-6}$.

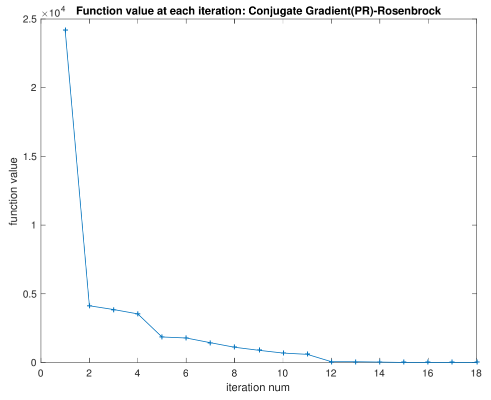
The optimization trajactery is given the profile plot figure 1:



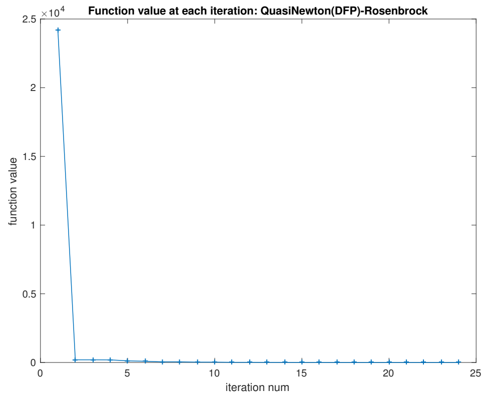
(a) Steepest Descent



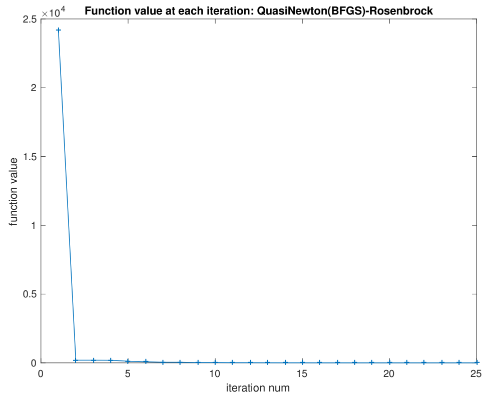
(b) Conjugate Gradient Fletecher Reeves



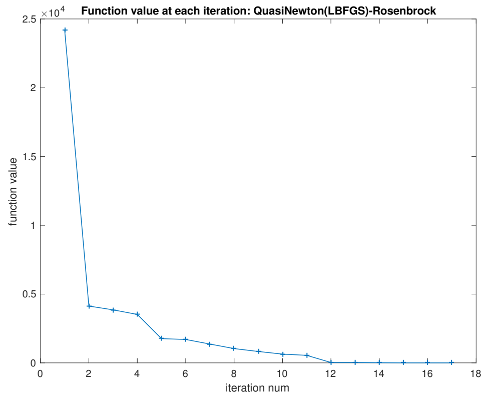
(c) Conjugate Gradient Polak-Ribiere



(d) DFP Quasi Newton



(e) BFGS Quasi Newton



(f) Limited Memory BFGS

Figure 1: Profile Plots of minimizing Rosenbrock function

Analysis

1. We first summarize the computation time of each algorithms:

Algorithm	Time(seconds)
Steepest Descent	0.345899
Conjugate Gradient FR	0.097148
Conjugate Gradient PR	0.080591
Quasi Newton DFP	7.221329
Quasi Newton BFGS	8.345914
Quasi Newton LBFGS	0.323257

2. The most obvious observation is that Steepest Descent, compare to other algorithm, takes siginificantly more number iterations to converge to given precision of error. From we plot we also see a several cases of slow decrease in function value during the procedure, overall the steepest descent algorithm is algorithmically simpliest one, but converging speed is relatively slow in the latter part of the algorithm.

3. In the class of Conjugate Gradient Methods, the Fletcher Reeves variant converge faster in the early phase, but shows slower converging speed in the latter phase. This also aligns with other numerical experiments showing that Polak-Ribiere method tends to be more efficient than the Fletcher-Reeves method in Nonquadratic setting. The Conjugate Gradient type methods enjoys much smallers number of iteration, while on the other hand do not have significant overhead compared to Quasi Newton method, in the experiments these two algorithms perform the best in terms of the actual computation time.

4. Overall the Quasi Newton type methods among others are the fastest ones in terms of **iteration numbers**. However per iteration in Quasi Newton type methods is much more time consuming than other methods. Especially the DFP and BFGS variants which took great amount of time in high dimension matrix multiplication and memory management per iteration. On the other hand, the limited memory BFGS reduce the computation for Approximate Inverse Hessian by converting matrix multiplication to matrix vector multiplication, and it do not explicitly form the approximate inversion in its computation, instead it directly computes the descent direction $-D_t g_t$. Its advantage over the previous two high memory methods was backed by our empirical results here.

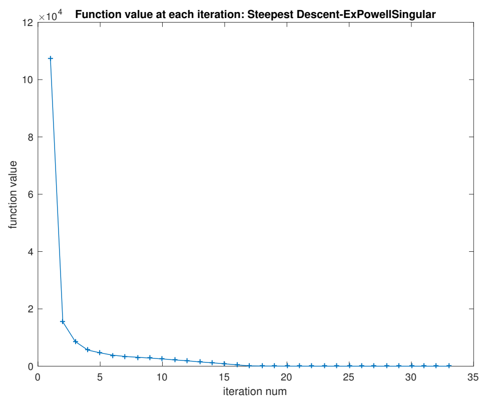
Extended Powell Singular Function

The Extended Powell Singular Function is defined, given n being multiple of 4, as the following:
For $i = 1, \ldots, \frac{n}{4}$:

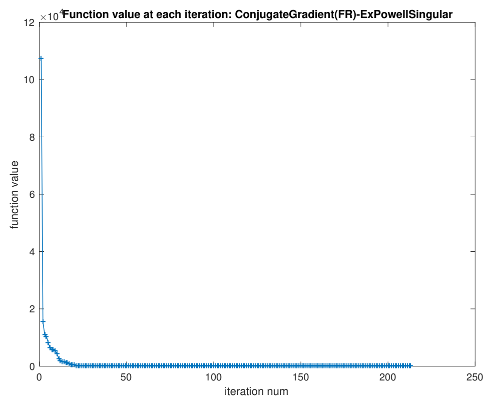
$$\begin{aligned} f_{4i-3}(x) &= x_{4i-3} + 10x_{4i-2} \\ f_{4i-2}(x) &= \sqrt{5}(x_{4i-1} - x_{4i}) \\ f_{4i-1}(x) &= (x_{4i-2} - 2x_{4i-1})^2 \\ f_{4i}(x) &= \sqrt{10}(x_{4i-3} - x_{4i})^2 \end{aligned}$$

We have $x_0 = (3, -1, 0, 1, \ldots, 3, -1, 0, 1)$ and $x_\star = (0, 0, 0, 0, \ldots, 0, 0, 0, 0)$.
In our implementation of the algorithm, we always choose $n = 2000$ and consider termination criterion given by $\frac{f(x_k) - f_\star}{f(x_o) - f_\star} < \epsilon$ where $\epsilon = 10^{-6}$.

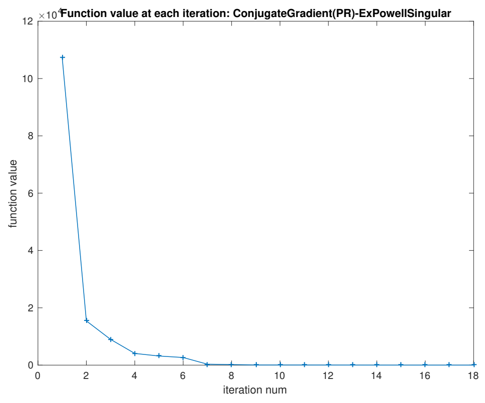
The optimization trajactery is given the profile plot figure 2.



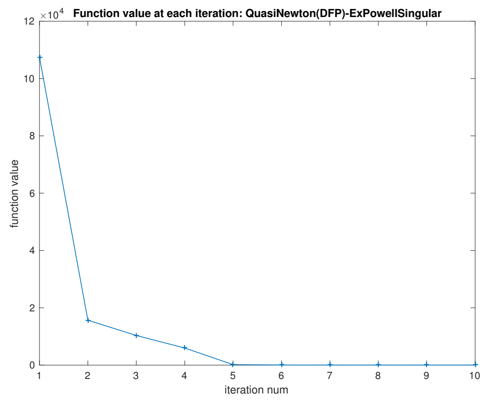
(a) Steepest Descent



(b) Conjugate Gradient Fletecher Reeves



(c) Conjugate Gradient Polak-Ribiere



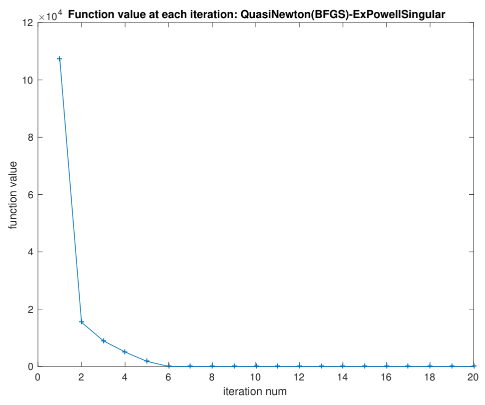
(d) DFP Quasi Newton

Figure 2: Profile Plots of minimizing Extended Powell Singular function

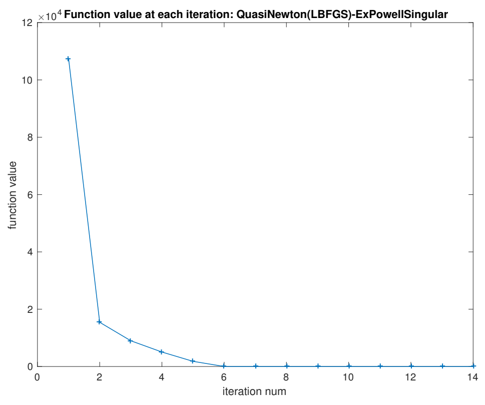
Analysis

1. We first summarize the actual computation time of different algorithms:

Algorithm	Time(seconds)
Steepest Descent	0.308159
Conjugate Gradient FR	0.571543
Conjugate Gradient PR	0.100476
Quasi Newton DFP	2.928752
Quasi Newton BFGS	5.021567
Quasi Newton LBFGS	0.234726



(e) BFGS Quasi Newton



(f) Limited Memory BFGS

Figure 2: Profile Plots of minimizing Extended Powell Singular function

2. The Steepest Descent methods takes fewer iteration compared to the Rosenbrock function.

3. In the Conjugate Gradient Type methods, the performance gap between the Fletecher Reeves variant and the Polak-Ribiere variant has increased siginificantly, leading it to have even worse converging speed than Steepest Descent, this might due to the increase of non-quadratic component in the function. The CG Polak-Ribiere variant remains to be the most computation efficient algorithm (in terms of computation time) among others.

4. The Quasi Newton type methods' performance remains steady, the DFP and BFGS variants still need great amout of time per iteration, while limited memory BFGS siginificantly reduce computation per iteration as before.

2 Problem 2

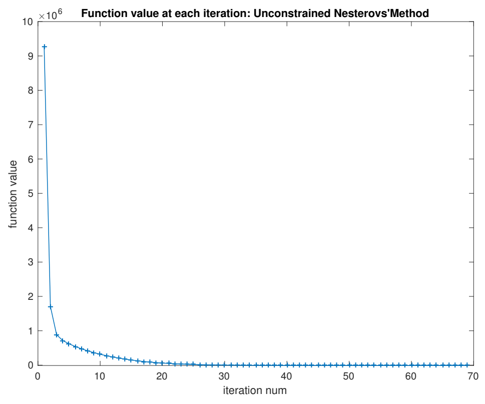
In this problem we condider minimizing the following function:

$$f(x) = \frac{x^T Q x}{2} - b^T x$$

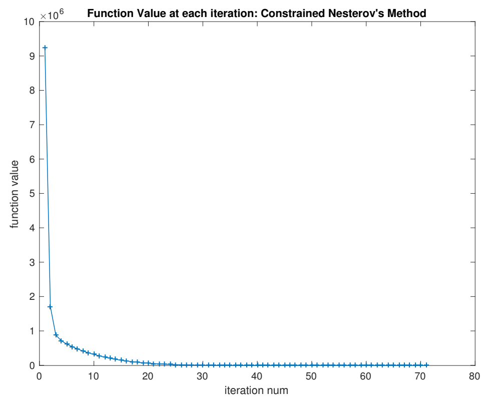
$$Q = \begin{pmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{pmatrix} + \delta \mathbf{I} \in \mathbb{R}^{n\times n},\ b = randn(n,1) \in \mathbb{R}^n$$

.

In our implementation we choose $n = 10^4$,$\delta = 0.05$. For part (a), we implemented unconstrained Nesterov's Acclerated Gradient Descent, for part (b), we implemented constrained

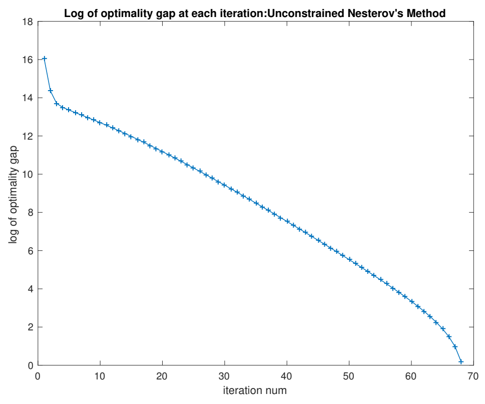


(a) Unconstrained Nesterov's Method

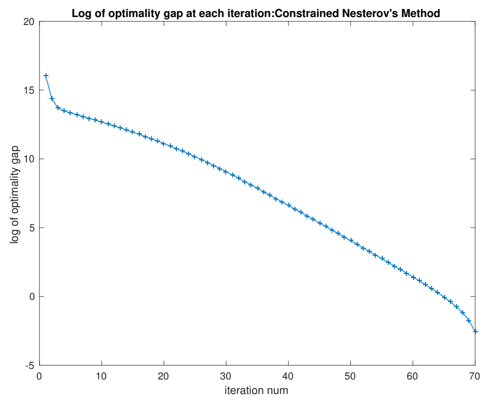


(b) Constrained Nesterov's Method

Figure 3: Profile Plots of minimizing f(x) using Nesterov's Method



(a) Unconstrained Nesterov's Method



(b) Constrained Nesterov's Method

Figure 4: Profile Plots of Log of Optimality Gap

Nesterov's Accelerated Gradient Descent. Our implementation closely follows the algorithm given in Nesterov's book[4], specifically on page 75 and 83 where constant stepsize scheme is adopted. Note that this algorithm needs to estimate condition number, in our example given structure of Q we could estimate $u = 0.05, L = 4.05$. The main difference in part (a) and part (b) is an additional projection step of calculating x_{k+1} from y_k.
The termination criterion is a bit different, for part (a) we used criterion $\|\nabla f(x_k)\|^2 \leqslant \epsilon(1 + \|\nabla f(x_0)\|^2)$. Since the problem is strongly convex, the left hand side of the equation is an upper bound of the optimality gap in terms of $f(x_k) - f_\star \leqslant \frac{1}{2u}\|\nabla f(x_k)\|^2$. Hence when $\|\nabla f(x_k)\|^2$ is small enough, then so is $f(x_k) - f_\star$. For part (b) we do not know anything about the optimal value, hence we choose to monitor the iterates x_t and terminate the algorithm whenever the distance between three consecutive iterates are sufficiently close.

The optimization trajactery is given figure 3

Analysis

1. We first summarize the computation time of two algorithms:

Algorithm	Time(seconds)
Unconstrained Nesterov's method	8.786521
Constrained Nesterov's method	5.746452

2. Since the objective function is strongly convex, we should in fact expect linear rate of convergence of algorithm, if we approximate optimal value using the last function value produced by the algorithm and consequently define optimality gap $f(x_k) - \hat{f}_\star$, and plot the log optimality gap against iteration number as in figure 4. It is quite straightforward to see the log of optimality gap is well approximated by a straight line, indicating linear convergence.

3 Conclusion

In this project we implement several classical gradient type optimization algorithm to convex and nonconvex function minimization. We saw that in Problem 1 the conjugate gradient methods enjoys both few iteration number and low computation overhead. The Quasi Newton method, with expection of LBFGS, have low iteration number but expensive overhead. The Steepest Descent memthod is the easiest to implement, converge with moderate speed. The Nesterov's Method enjoys linear convergence rate for our strongly convex objective function. During the implementation we also found that the line search criterion also plays significant part in the performance of each algorithm. Thus how to implement a line search method that is more robust to line search's parameter specification is still an open problem to us. From a theoretical point of view, the strong wolfe condition's role in gauranteeing convergence is also of interest to us.

References

[1] Guipeng li. `"https://github.com/GuipengLi/optLBFGS"`.

[2] Jorge j. more', david j. thuente. `"https://www.cs.umd.edu/users/oleary/software/"`.

[3] Dimitri P. Bertsekas. *Nonlinear Programming.* Athena Scientific, 1995.

[4] Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course.* Springer, 2004.