ISE 417: Nonlinear Optimization
FINAL PROJECT REPORT

Xi He

A survey on some nonlinear programming algorithms

Department of Industrial and Systems Engineering
Lehigh University
Spring 2015

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This project concerns the practical implementation of several classical optimization algorithms in the context of unconstrained nonlinear problems. By choosing proper parameter set and delicate implementation, we intend to achieve economy of computation or the solution of such problems. With respect to several test problems, we compare the various performance of each algorithms with different parameters setup. It also might be a reasonable instruction on deciding which algorithm should be applied when faced a new problem.

At the first part of this project, we study several algorithms on solving unconstrained nonlinear optimization problems, such as steepest descent method, newton method and BFGS method with two kinds of linear search strategy: backtrack line search method and wolfe line search method. Trust region method is also argued with two similar approaches to solve trust region subproblem: conjugate gradient method and conjugate gradient method with SR1 Hessian matrix update method. Throughout the introduction of each algorithm, we also list convergence and complexity issues, however, one should note that we state some correct conclusions without rigorous proof.

In the latter part of this project, some detailed implementation concerns are discussed with respect to different problems and algorithms, including practical parameter set choosing, tricks on attaining economy computation and also some difficulties with respect to implementation. Results on numerical experiment are shown to clearly compare the performance of each algorithm with different parameters on each problem. Also, some analysis of the numerical result are presented.

The structure of this project report is as follows. Chapter 2 includes relevant background on unconstrained optimization problems which forms the basis of the discussions in later chapters. In Chapter 3, after showing the very basic two descent directions we used in implementation, we discuss two kinds of linear search strategy. And then we present two kinds of quasi-newton method and show the global convergence results when applying the two line search method mentioned before. We introduce in Chapter 4 about trust region algorithms and conjugate gradient method with is proposed by solving the trust region subproblems. SR1 update which is stated in Chapter 3 will be reconsidered as a approach to solving trust region subproblem when combining with conjugate gradient method. Finally, in Chapter 5, we present, analyze and provide numerical results with respect to the algorithms we discuss in above chapters on some test problems. Brief summary of this project and some general conclusion will be stated in the Chapter 6.

In this report, we use the following notation. Let $f(x)$ be original nonlinear, smooth and differentiable objective function we want to minimize and x be decision variables. We also denote gradient function of $f(x)$ by $g(x) = \nabla f(x)$ and denote $H(x) = \nabla^2 f(x)$ and $B(x)$ by the Hessian and approximation Hessian matrix of $f(x)$ at point x. With respect to algorithms, we use $d(x)$ to be an acceptable descent directions at point x and $\alpha(x)$ to be an acceptable step-size along $d(x)$.

At the last, $H \succ (\succeq) 0$ represents that the matrix is (semi) positive definite.

Chapter 2

Fundamentals of Unconstrained Optimization

We frame this report in the context of the unconstrained optimization problem

$$\min_{x} \quad f(x), \tag{2.1}$$

where $x \in \mathbb{R}^n$ is a real vector with $n \geq 1$ components $f : \mathbb{R}^n \to \mathbb{R}$ is a smooth function.

Generally, we intend to find out a global minimizer of f, a point where the function attains its least value in whole space. We state a formal definition as

Definition 2.1. *A point x^* is a global minimizer if $f(x^*) \leq f(x)$ for all x.*

Note that it can be very difficult to find the global minimizer, since our knowledge of f is usually only local. Actually, most algorithms are able to find only a local minimizer, a point that achieves the smallest value of f in its neighborhood.

Definition 2.2. *A point x^* is a local minimizer if there is a neighborhood $\mathcal{N}$ of x^* such that $f(x^*) \leq f(x)$ for $x \in \mathcal{N}$.*

As it is shown in Figure 2.1, in general case, a function f may have a lot of local minimizers but just one global minimizer. Most algorithms is sensitive on the initial point we chose, which means, by choosing different initial point to run those algorithms, we may get different local minimizer.

A special case we should concern is that of convex functions, every local minimizer is also a global minimizer. We state formal definition of convex as following

Definition 2.3. *A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if for all $\{x_1, x_2\} \in \mathbb{R}^n$ and $\alpha \in [0, 1]$ we have*

$$f(\alpha x_1 + (1 - \alpha)x_2) \leq \alpha f(x_1) + (1 - \alpha)f(x_2). \tag{2.2}$$

Therefore, to solve a unconstrained optimization with respect to a convex objective function, we have conclusion that

Theorem 2.4. *When f is convex, any local minimizer x^* is a global minimizer of f. If in addition f is differentiable, then any stationary point x^* is a global minimizer of f.*

When turns to general case, i.e., objective function is not assuming to be convex but still smooth, we have following efficient and practical ways to identify local minima. To do it, we need have knowledge of the gradient $g(x^*)$ and the Hessian $H(x^*)$ of function f at point x^*.

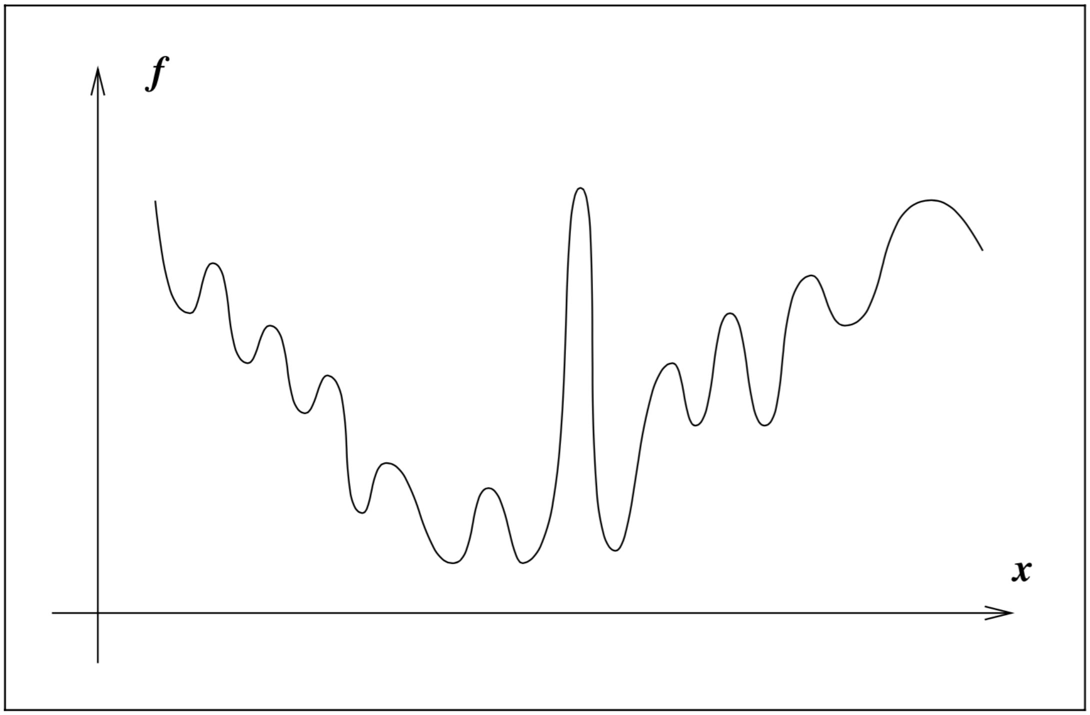

Figure 2.1: Global minimum and Local minimum

Theorem 2.5. *(First-Order Necessary Conditions) If x^* is a local minimizer and f is continuously differentiable in an open neighborhood of x^*, then $g(x^*) = 0$.*

We call x^* a stationary point if $g(x^*) = 0$. According to Theorem 2.5, any local minimizer must be a stationary point. When consider Hessian matrix, we have the following conditions

Theorem 2.6. *(Second-Order Necessary Conditions) If x^* is a local minimizer of f and H is continuous in an open neighborhood of x^*, then $g(x^*) = 0$ and $H(x^*)$ is positive semidefinite.*

Now, we state sufficient conditions, which are conditions on the derivatives of f at the point x^* that guarantee that x^* is a local minimizer.

Theorem 2.7. *(Second-Order Sufficient Conditions) Suppose that H is continuous in an open neighborhood of x^* and that $g(x^*) = 0$ and $H(x^*)$ is positive definite. Then x^* is a strict local minimizer of f.*

Chapter 3

Algorithm Descriptions I: Flexible Step Method

3.1 Steepest Descent and Newton method

3.2 Line Search Methods

3.3 Quasi-newton method

Chapter 4

Algorithm Descriptions II: Restricted Step Methods.

4.1 Trust Region Method

4.2 Conjugate Gradient Method

4.3 Trust Region Subproblem

Chapter 5

Numerical Results

This chapter contains numerical results with respect to those algorithms we discussed in Chapter 3 and 4.

Note that all algorithms are implemented by Matlab under Intel Core $i5$ 2.6 GHz processor and 8 GB memory. Throughout this chapter, with respect to each algorithm, we use *Iter.* to represent the total number of Iter.ations it takes to meet termination conditions. If the actual *Iter.* is greater than defined *maxiter*, we mark *Iter.* by a slash. *Cputime* stands for total time of a algorithms spend to find the minimizer and when the corresponding *Iter.* is marked as a slash, *Cputime* means time cost up to *maxiter* iteration. We denote $xNorm$ by the norm of difference between ending point of an algorithm and the accurate minimizer. $gNorm$ represents for the norm of gradient at the ending point.

We begin by providing some general comments that may be useful for achieve a economy computation.

We set the default parameter value as following

Table 5.1: Default Parameter Set

i	**maxiter**	**opttol**	**(c1ls,c2ls)**	**(c1tr,c2tr)**	
value	1000	10^{-6}	$(10^{-4}, 0.9)$	$(0.3, 0.9)$	
i	**cgopttol**	**cgmaxiter**	**sr1updatetol**	**bfgsupdatetol**	**radius**
value	10^{-6}	50	10^{-6}	0.2	0.25

By this default set, we state numerical results for all four test problems, respectively.

(1) Rosenbrock

Objective function:

$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2. \tag{5.1}$$

Accurate minimizer:

$$(x_1, x_2) = (1, 1). \tag{5.2}$$

Numerical results with initial point $0\mathbf{e}$:

(2) Genhumps

Table 5.2: Rosenbrock, with initial $0\mathbf{e}$

Rosenbrock	**xNorm**	**gNorm**	**Iter.**	**Cputime**
steepestbacktrack	$1.253e+00$	$1.574e-01$	–	0.6407
steepestwolfe	$1.255e+00$	$1.010e-01$	–	0.5415
newtonbacktrack	$1.414e+00$	$9.801e-07$	13	0.0660
newtonwolfe	$1.414e+00$	$3.652e-10$	14	0.0810
trustregioncg	$1.414e+00$	$1.310e-07$	42	0.0855
sr1trustregioncg	$1.414e+00$	$3.032e-07$	77	0.0822
bfgsbacktrack	$1.414e+00$	$1.205e-08$	26	0.0771
bfgswolfe	$1.414e+00$	$1.170e-07$	21	0.0778

Objective function:

$$f(x) = \sum_{i=1}^{4} (\sin(2x_i)^2 \sin(2x_{i+1})^2 + 0.05(x_i^2 + x_{i+1}^2)). \tag{5.3}$$

Accurate minimizer:

$$x = 0\mathbf{e}. \tag{5.4}$$

Numerical results with initial point $2\mathbf{e}$:

Table 5.3: Genhumps, with initial $2\mathbf{e}$

Genhumps	**xNorm**	**gNorm**	**Iter.**	**Cputime**
steepestbacktrack	$4.552e-05$	$4.552e-06$	156	0.1750
steepestwolfe	$4.648e-05$	$4.648e-06$	155	0.1365
newtonbacktrack	$7.363e-08$	$1.291e-08$	18	0.0817
newtonwolfe	$1.476e-09$	$1.682e-10$	14	0.0665
trustregioncg	$1.037e-06$	$1.674e-07$	50	0.0833
sr1trustregioncg	$1.046e-05$	$1.551e-06$	461	0.2313
bfgsbacktrack	$9.840e-06$	$1.867e-06$	19	0.0754
bfgswolfe	$9.840e-06$	$1.867e-06$	19	0.0671

(3) Quadratic
Objective function:

$$f(x) = g^T x + \frac{1}{2} x^T H x. \tag{5.5}$$

where $g \in \mathbb{R}^{10}$ and $H \in \mathbb{R}^{10\times 10}$.
Accurate minimizer:

$$x = (H^T H)^\dagger H^T g. \tag{5.6}$$

Numerical results with initial point $0\mathbf{e}$:
(4) Leastsquares
Objective function:

$$f(x) = \frac{1}{2} \| x_1 \mathbf{e} + x_2 e^{-\frac{t + x_3 \mathbf{e}}{x_4}} - y \|^2. \tag{5.7}$$

Table 5.4: Quadratic, with initial $0\mathbf{e}$

Quadratic	**xNorm**	**gNorm**	**Iter.**	**Cputime**
steepestbacktrack	$5.612e-04$	$1.537e-06$	18	0.2044
steepestwolfe	$5.612e-04$	$1.537e-06$	18	0.1459
newtonbacktrack	$5.611e-04$	$4.591e-16$	1	0.0777
newtonwolfe	$5.611e-04$	$4.591e-16$	1	0.0823
trustregioncg	$5.605e-04$	$2.445e-06$	33	0.1816
sr1trustregioncg	$5.615e-04$	$2.596e-06$	28	0.1749
bfgsbacktrack	$5.624e-04$	$2.952e-06$	10	0.2255
bfgswolfe	$5.624e-04$	$2.952e-06$	10	0.1300

where $y = z_1\mathbf{e} + z_2 e^{-\frac{t+z_3\mathbf{e}}{z_4}} + \epsilon \in \mathbb{R}^{100}$, $z = (2, 1, -5, 4)^T$ and $\epsilon \in \mathbb{R}^{100}$ is a perturbation.
Accurate minimizer:

$$x = (189.9 - 187.2 - 4.948 - 4862)^T. \tag{5.8}$$

Numerical results with initial point $(0, 0, 0, 1)^T$:

Table 5.5: Leastsquares, with initial $(0, 0, 0, 1)^T$

Leastsquares	**xNorm**	**gNorm**	**Iter.**	**Cputime**
steepestbacktrack	$4.391e+03$	$1.189e-02$	–	1.6771
steepestwolfe	$4.391e+03$	$1.185e-02$	–	1.0915
newtonbacktrack	$4.392e+03$	$6.375e-07$	17	0.0973
newtonwolfe	$4.392e+03$	$6.375e-07$	17	0.0820
trustregioncg	$3.261e-01$	$9.553e-07$	177	0.1801
sr1trustregioncg	$4.392e+03$	$4.419e-07$	152	0.1783
bfgsbacktrack	$4.390e+03$	$1.132e-14$	5	0.0615
bfgswolfe	$4.390e+03$	$1.132e-14$	5	0.0626

Chapter 6

Conclusion

Bibliography

Appendix A

Mathematical Details