# Report on Numerical Optimization Coding Assignment

## Newton-type Methods

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

April 19, 2018

## 1 Overview

Given a smooth function $f : \mathbb{R}^n \to \mathbb{R}$, we use a series of Newton-type methods to solve the unconstrained optimization problem

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

including damped Newton's method, modified Newton's method and quasi-Newton methods (SR1, DFP and BFGS). Denote by $g$ and $G$ respectively the gradient and the Hessian of $f$, then the following is an overview of all implemented algorithms:

- Damped Newton: Newton's direction with line search;
- Modified Newton (mix): choose between Newton's direction and negative gradient;
- Modified Newton (LM): modify the Hessian $G$ by adding a multiple of identity when $G$ is not positive definite;
- SR1: rank-1 update of approximation of Hessian inverse $H_k$ by

$$H_{k+1}^{\mathrm{SR1}} = H_k + \frac{(s_k - H_k y_k)(s_k - H_k y_k)^T}{(s_k - H_k y_k)^T y_k};$$

- DFP: rank-2 update of approximation of Hessian inverse $H_k$ by

$$H_{k+1}^{\mathrm{DFP}} = H_k + \frac{s_k s_k^T}{s_k^T y_k} - \frac{H_k y_k y_k^T H_k}{y_k^T H_k y_k};$$

**Algorithm 1** Inexact line search (bracketing phase)

**Input:** Function $\phi(\alpha) = f(x_k + \alpha d_k)$, an interval $[\alpha_{\min}, \alpha_{\max}]$
**Output:** A step length $\alpha^*$ satisfying strong Wolfe condition

```
Initialize i = 1, α_0 = α_min, α_1 = 1
while true do
    if φ(α_i) > φ(0) + ρφ'(0)α_i then
        return zoom(α_{i-1}, α_i)
    if |φ'(α_i)| ≤ −σφ'(0) then
        return α_i
    if φ'(α_i) ≥ 0 then
        return zoom(α_i, α_{i-1})
    Choose α_{i+1} ∈ (α_i, α_max)
    i = i + 1
```

- BFGS: rank-2 update of approximation of Hessian inverse $H_k$ by

$$H_{k+1}^{\text{BFGS}} = H_k + \left(1 + \frac{y_k^T H_k y_k}{y_k^T s_k}\right) \frac{s_k s_k^T}{y_k^T s_k} - \frac{s_k y_k^T H_k + H_k y_k s_k^T}{y_k^T s_k}.$$

In general, a multiple of identity $\beta I_n$ ($\beta > 0$) is chosen as the initial approximate of Hessian inverse $H_0$, and we fix $\beta = 1$ for the moment. The iteration procedure stops when both $|f_k - f_{k-1}| < \epsilon$ and $\|g_k\| < \epsilon$ hold where $\epsilon = 10^{-8}$.

### 1.1 Line search algorithm

Given $x_k$ and a descent direction $d_k$, the line search algorithm finds an appropriate step length $\alpha$ for the value of $\phi(\alpha) = f(x_k + \alpha d_k)$ to have a sufficient decrease compared to $f(x_k)$. Both exact and inexact line search algorithms are implemented. Exact line search uses 0.618 method to search a step length that minimizes $\phi(\alpha)$ within a specified interval $[\alpha_{\min}, \alpha_{\max}]$, which we set to $[10^{-2}, 10^{1}]$ by default. Here we assume $\alpha_{\min} > 0$ to prevent unbearably slow progress caused by needlessly small step lengths.

In the meanwhile, inexact line search is also implemented to provide a step length which satisfies strong Wolfe conditions, i.e.

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

By default, we set $\rho = 10^{-4}$ and $\sigma = 0.9$ for a general loose search. We adopt a searching

**Algorithm 2** Inexact line search (selection phase)

**Input:** Function $\phi(\alpha) = f(x_k + \alpha d_k)$, an interval $[\alpha_{\text{lo}}, \alpha_{\text{hi}}]$
**Output:** A step length $\alpha^*$ satisfying strong Wolfe condition

```
while true do
    Choose α_i ∈ (α_lo, α_hi)
    if φ(α_i) > φ(0) + ρφ'(0)α_i or φ(α_i) ≥ φ(α_lo) then
        α_hi = α_i
    else
        if |φ'(α_i)| ≤ −σφ'(0) then
            return α_i
        if φ'(α_i)(α_hi − α_lo) ≥ 0 then
            α_hi = α_lo
        α_lo = α_i
```

routine introduced in [1], which first enters a bracketing phase, shown in Algorithm 1, to find an interval $[\alpha_{\text{lo}}, \alpha_{\text{hi}}]$ which must contain an $\alpha$ satisfying (*), and then enters a selection phase called zoom, shown in Algorithm 2, with $\alpha_{\text{lo}}$ being the one that gives the smallest $\phi(\alpha)$ among all $\alpha_i$'s, and $\phi'(\alpha_{\text{lo}})(\alpha_{\text{hi}} - \alpha_{\text{lo}}) < 0$ always true. In our implementation, the choose step uses interpolation in the first phase, and bisection in the second phase.

## 2 Problem 1: Box three-dimensional function

The first objective function we are going to tackle is the box three-dimensional function, defined as

$$f(x) = \sum_{k=1}^{m} \left[ e^{-0.1kx_1} - e^{-0.1kx_2} - x_3(e^{-0.1k} - e^{-k}) \right]^2, \quad x \in \mathbb{R}^3,$$

where $m \geq 3$. The initial point $x_0 = (0, 10, 20)^T$, $H_0 = I_n$, and the minimum $f^* = 0$.

The results are given in Table 1, in which we try five different methods on parameters $m = 3, 5, 10, 15, 20$. 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 1000, and $N_{iter} = \max$ if the method does not converge after 1000 iterations. $N_{eval}$ counts evaluations of $f, g$ and $G$. Damped Newton's method is not listed since the initial Hessian $G_0$ is not positive definite, but it actually acts the same as modified Newton (mix) if we ignore this issue and proceed the iteration.

Table 1: Result comparison for the box three-dimensional function

(a) $m = 3$, $m = 5$

| Method | $m = 3$ | | | | $m = 5$ | | | |
|---|---|---|---|---|---|---|---|---|
| | $f(x^*)$ | $-\lg\|g^*\|$ | $N_{iter}$ | $N_{eval}$ | $f(x^*)$ | $-\lg\|g^*\|$ | $N_{iter}$ | $N_{eval}$ |
| Modified (mix) | 1.536e-15 | 8.0 | 6 | 26 | 9.995e-18 | 9.2 | 8 | 34 |
| Modified (LM) | 7.185e-21 | 10.2 | 16 | 78 | 6.773e-24 | 12.5 | 18 | 84 |
| SR1 | 4.116e-16 | 8.2 | 12 | 48 | 3.916e-17 | 8.9 | 24 | 86 |
| DFP | 6.047e-08 | 3.6 | max | 3034 | 3.909e-21 | 9.8 | 517 | 3512 |
| BFGS | 1.397e-17 | 8.4 | 17 | 53 | 7.561e-17 | 8.2 | 33 | 104 |

(b) $m = 10$, $m = 15$

| Method | $m = 10$ | | | | $m = 15$ | | | |
|---|---|---|---|---|---|---|---|---|
| | $f(x^*)$ | $-\lg\|g^*\|$ | $N_{iter}$ | $N_{eval}$ | $f(x^*)$ | $-\lg\|g^*\|$ | $N_{iter}$ | $N_{eval}$ |
| Modified (mix) | 1.337e-24 | 12.2 | 9 | 38 | 6.044e-17 | 8.2 | 9 | 38 |
| Modified (LM) | 3.485e-21 | 10.5 | 14 | 68 | 8.709e-02 | 8.0 | 124 | 542 |
| SR1 | 2.267e-23 | 10.8 | 14 | 62 | 1.737e-19 | 8.9 | 17 | 81 |
| DFP | 5.159e-06 | 2.2 | max | 3032 | 1.928e-07 | 4.0 | max | 3026 |
| BFGS | 5.937e-21 | 9.6 | 18 | 88 | 1.372e-18 | 8.4 | 25 | 90 |

(c) $m = 20$

| Method | $f(x^*)$ | $-\lg\|g^*\|$ | $N_{iter}$ | $N_{eval}$ |
|---|---|---|---|---|
| Modified (mix) | 9.985e-20 | 9.4 | 10 | 42 |
| Modified (LM) | 9.439e-02 | 8.1 | 155 | 696 |
| SR1 | 4.060e-19 | 9.0 | 28 | 121 |
| DFP | 3.615e-21 | 9.6 | 272 | 2087 |
| BFGS | 6.646e-21 | 9.5 | 42 | 157 |

Table 2: DFP method with different line search routines

| | | $f(x^*)$ | $-\lg\|g^*\|$ | $N_{iter}$ | $N_{eval}$ |
|---|---|---|---|---|---|
| $m = 10$ | Exact | 1.860e-17 | 8.2 | 44 | 662 |
| | Inexact | 4.714e-20 | 9.1 | 425 | 5963 |
| $m = 15$ | Exact | 1.807e-19 | 9.2 | 25 | 377 |
| | Inexact | 7.650e-17 | 8.3 | 609 | 7610 |
| $m = 20$ | Exact | 1.106e-22 | 10.4 | 35 | 527 |
| | Inexact | 3.124e-07 | 2.7 | max | 3013 |

As can be seen from Table 1, modified Newton's method with mixed directions converge at the fastest rate with no surprise. Levenberg-Marquardt (LM) method, however, does not seem attractive — the modification on Hessian $G + \nu_k I$ drags the direction towards negative gradient $-\nu_k^{-1} g_k$, but the upper limit set for the step length makes a feasible line search much tougher if $\nu_k$ is too large. An interesting discovery about $\nu$: if we initialize $\nu = \gamma \|G_k\|_F$, it is usually more effective to set $\gamma = 1/2$.

For quasi-Newton methods, SR! and BFGS works fairly well. They take more iterations than Newton's method to converge due to their superlinear convergence. SR1 tends to provide a better approximation of Hessian for this problem than BFGS does, as is indicated by its fewer iterations to reach convergence.

DFP, however, performs a lot worse than SR1 and BFGS. We find in our experiments that SR1 and DFP are more sensitive to the accuracy of line search. The SR1 results in Table 1 are obtained with $\alpha_{\min} = 0.008$. Nonetheless, two out of these five settings fail to converge if we set $\alpha_{\min} = 0.01$. Furthermore, DFP fails in a lot of cases we tried, but this misery can be relieved if we improve the line search precision by using exact search routines. Table 2 shows the comparison of DFP performance using exact and inexact searches with $\alpha \in [0.005, 2]$. DFP with exact search can function nearly as well as BFGS, at a cost of more number of function evaluations during line search.

## 3 Problem 2: Penalty II function

The second objective function we are coping with is penalty II function, given by

$$f(x) = (x_1 - 0.2)^2 + a \sum_{i=2}^{n} \left[ (e^{\frac{x_i}{10}} + e^{\frac{x_{i-1}}{10}} - y_i)^2 + (e^{\frac{x_i}{10}} - e^{\frac{-1}{10}})^2 \right] + \left[ \sum_{j=1}^{n} (n - j + 1) x_j^2 - 1 \right]^2$$

where $a = 10^{-5}$ and $y_i = e^{\frac{i}{10}} - e^{\frac{i-1}{10}}$. The initial value $x_0 = (1/2, 1/2, \cdots, 1/2)^T \in \mathbb{R}^n$. The function minimum is a bit complicated to conclude as $n$ varies.

Table 3 gives the results on how these methods perform on this more sophisticated problem. Modified Newton (mix) still shows incredibly fast convergence, and LM method, contrary to how it behaves in the first problem, is as efficient as mixed direction method. BFGS method apparently outperforms SR1 method this time, which is another result different from our previous observations. DFP again ranks the last — four out of five settings fail to converge.

Table 3: Result comparison for penalty II function

(a) $n = 2$, $n = 4$

| Method | $n = 2$ | | | | $n = 4$ | | | |
|---|---|---|---|---|---|---|---|---|
| | $f(x^*)$ | $-\lg\|g^*\|$ | $N_{iter}$ | $N_{eval}$ | $f(x^*)$ | $-\lg\|g^*\|$ | $N_{iter}$ | $N_{eval}$ |
| Modified (mix) | 8.066e-07 | 10.8 | 4 | 23 | 9.376e-06 | 8.2 | 108 | 441 |
| Modified (LM) | 8.066e-07 | 9.8 | 7 | 32 | 9.376e-06 | 8.1 | 110 | 455 |
| SR1 | 8.066e-07 | 10.0 | 94 | 1084 | 9.376e-06 | 9.2 | 484 | 3043 |
| DFP | 8.066e-07 | 9.7 | 28 | 90 | 1.022e-05 | 3.6 | max | 3925 |
| BFGS | 8.066e-07 | 8.3 | 10 | 35 | 9.376e-06 | 8.3 | 461 | 1478 |

(b) $n = 6$, $n = 8$

| Method | $n = 6$ | | | | $n = 8$ | | | |
|---|---|---|---|---|---|---|---|---|
| | $f(x^*)$ | $-\lg\|g^*\|$ | $N_{iter}$ | $N_{eval}$ | $f(x^*)$ | $-\lg\|g^*\|$ | $N_{iter}$ | $N_{eval}$ |
| Modified (mix) | 4.193e-05 | 8.3 | 111 | 456 | 1.233e-04 | 9.3 | 103 | 423 |
| Modified (LM) | 4.193e-05 | 9.0 | 112 | 469 | 1.233e-04 | 9.2 | 103 | 416 |
| SR1 | 4.193e-05 | 8.4 | 518 | 3423 | 1.233e-04 | 8.6 | 729 | 5380 |
| DFP | 4.293e-05 | 3.0 | max | 10634 | 1.236e-04 | 3.2 | max | 3042 |
| BFGS | 4.193e-05 | 9.8 | 408 | 1328 | 1.233e-04 | 8.0 | 275 | 874 |

(c) $n = 10$

| Method | $f(x^*)$ | $-\lg\|g^*\|$ | $N_{iter}$ | $N_{eval}$ |
|---|---|---|---|---|
| Modified (mix) | 2.937e-04 | 10.3 | 92 | 376 |
| Modified (LM) | 2.937e-04 | 9.3 | 93 | 381 |
| SR1 | 2.937e-04 | 8.2 | 724 | 4621 |
| DFP | 8.821e-04 | 1.2 | max | 4971 |
| BFGS | 2.937e-04 | 8.0 | 312 | 976 |

Table 4: Numerical performance after/before adding initial scaling step

| | | $n = 6$ | | | $n = 8$ | | | $n = 10$ | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | $-\lg\|g^*\|$ | $N_{iter}$ | $N_{eval}$ | $-\lg\|g^*\|$ | $N_{iter}$ | $N_{eval}$ | $-\lg\|g^*\|$ | $N_{iter}$ | $N_{eval}$ |
| SR1 | After | 8.4 | 518 | 3423 | 8.6 | 729 | 5380 | 8.2 | 724 | 4621 |
| | Before | 3.8 | max | 6786 | 3.4 | max | 6925 | 8.7 | 862 | 5768 |
| DFP | After | 3.0 | max | 10634 | 3.2 | max | 3042 | 1.2 | max | 4971 |
| | Before | 3.2 | max | 3107 | 2.7 | max | 9966 | 2.4 | max | 3128 |
| BFGS | After | 9.8 | 408 | 1328 | 8.0 | 275 | 874 | 8.0 | 312 | 976 |
| | Before | 10.3 | 718 | 2351 | 8.9 | 614 | 2024 | NaN | / | / |

When solving this problem, overflow is a great headache we often encounter during the iteration process. We notice that the initial gradient $g_0$ has some very large elements, leading to overflow in the exponential part of $f$. Therefore, instead of repeatedly cutting down the step length even when it goes below $\alpha_{\min}$, we choose to scale $H_0$ right before the first update of $H_k$ by

$$H_0 \leftarrow \frac{y_k^T s_k}{y_k^T y_k} I_n,$$

which tries to approximate $G_k^{-1}$. To see this, define

$$\bar{G}_k = \int_0^1 G(x_k + \tau \alpha_k d_k) \mathrm{d}\tau,$$

then the Taylor's theorem gives that $y_k = \bar{G}_k s_k$. Suppose $\bar{G}_k$ is positive definite, then it has a square root $\sqrt{\bar{G}_k}$. Let $z_k = \sqrt{\bar{G}_k} s_k$, then

$$\frac{y_k^T s_k}{y_k^T y_k} = \frac{(\sqrt{\bar{G}_k} s_k)^T (\sqrt{\bar{G}_k} s_k)}{(\sqrt{\bar{G}_k} s_k)^T \bar{G}_k (\sqrt{\bar{G}_k} s_k)} = \frac{z_k^T z_k}{z_k \bar{G}_k z_k}$$

is the reciprocal of a Rayleigh quotient, which can approximate an eigenvalue of $G_k^{-1}$ (assuming $\bar{G}_k$ is close to $G_k$ in some sense). Table 4 gives some results on how this scaling on initial $H_0$ helps improve convergence, in which "After" and "Before" means whether or not to add this scaling step. It greatly cuts down the number of iterations needed for convergence in terms of SR1 and BFGS, and also overcomes an NaN caused by overflow. DFP does not benefit much from this step, encumbered primarily by the loose line search.

# 4 Problem 3: Minimal surface problem

Given a convex domain $\Omega \subseteq \mathbb{R}^2$ and its boundary values $q : \partial\Omega \to \mathbb{R}$, the minimal surface problem aims to solve

$$\min \{E[u] : \ u \in K\},$$

where the functional $E : K \to \mathbb{R}$ is the area of a surface $u(x, y)$ defined on $\Omega$, given by

$$E[u] = \int_\Omega \sqrt{1 + \|\nabla u(x)\|^2} \, \mathrm{d}x\mathrm{d}y,$$

and all the candidates form the set

$$K = \left\{ u \in H^1(\Omega) : \ u|_{\partial\Omega} = q \right\}.$$

Table 5: Results for minimal surface problem ($N = 20$)

| Method | $f(x^*)$ | $-\lg\|g^*\|$ | $N_{iter}$ | $N_{eval}$ |
|---|---|---|---|---|
| SR1 | 3.306e+00 | 8.2 | 542 | 4620 |
| DFP | 3.306e+00 | 8.1 | 403 | 2636 |
| BFGS | 3.306e+00 | 8.1 | 539 | 5526 |

Given a grid on a rectangular region $\Omega$ with stride $h_x$ and $h_y$ in $x$ and $y$ directions respectively, a finite element approximation to this problem can be obtained by minimizing $E[u]$ over the space of piecewise constant functions, whose element $u$ remains constant $u_{ij}$ on its control volume $V_{ij} = (x_i - \frac{h_x}{2}, x_i + \frac{h_x}{2}) \times (y_j - \frac{h_y}{2}, y_j + \frac{h_y}{2})$. With the central difference scheme for approximating derivatives, the original problem is reduced to

$$\min_U \sum_{i,j=1}^{N-1} h_x h_y \sqrt{1 + \left(\frac{u_{i-1.j} + u_{i+1,j} - 2u_{ij}}{h_x^2}\right)^2 + \left(\frac{u_{i.j-1} + u_{i,j+1} - 2u_{ij}}{h_y^2}\right)^2},$$

where $U = [u_{ij}] \in \mathbb{R}^{N-1 \times N-1}$ and $u_{ij} = q(x_i, y_j)$ if $\{i, j\} \cap \{0, N\} \neq \emptyset$. Assume $h_x = h_y = 1/N$, then the objective function above can be written as

$$f(U) = \sum_{i,j=1}^{N-1} \sqrt{N^{-4} + (u_{i-1.j} + u_{i+1,j} - 2u_{ij})^2 + (u_{i.j-1} + u_{i,j+1} - 2u_{ij})^2}.$$

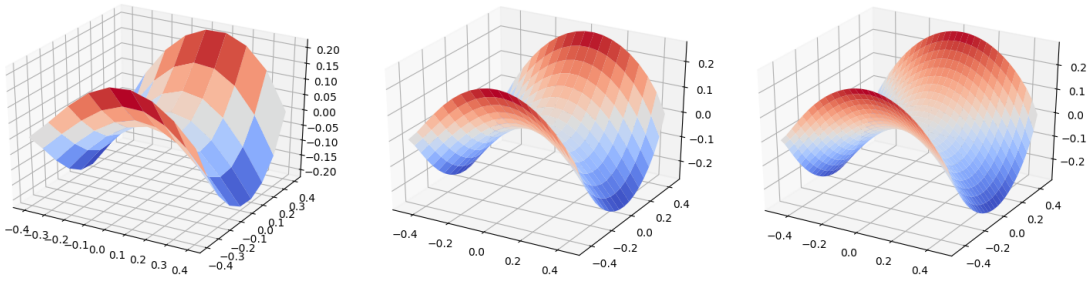
Note that transforms between matrices and vectors are needed to make it possible to optimize by Newton-type methods.

We pick the Enneper surface for numerical experiments. It specifies the domain $\Omega = (-\frac{1}{2}, \frac{1}{2}) \times (-\frac{1}{2}, \frac{1}{2})$, and the boundary values are given by $q(x, y) = s^2 - t^2$ where $s$ and $t$ are the solutions to the equations

$$x = s + st^2 - \frac{s^3}{3}, \quad y = -t - s^2 t + \frac{t^3}{3}. \tag{**}$$

We have tried $N = 10, 20, 30$ with increasing complexity as $N$ grows large. Table 5 shows how quasi-Newton methods perform on this particular problem when $N = 20$. Again different from previous conclusions, DFP converges with the fewest iterations among the three methods, while SR1 and BFGS produce similar results.

Figure 1 displays how our numerical Enneper surface looks like for $N = 10, 20, 30$. As $N$ increases, the surface looks smoother and the results are more accurate. But the problem becomes much harder to optimize: more computations in function evaluation



(a) $N = 10$ (b) $N = 20$ (c) $N = 30$

Figure 1: Numerical solution of Enneper surface

($O(N^2)$ per time), more storage requirement for matrices ($O(N^4)$), and higher susceptibility to parameters and accumulative errors. In fact, when $N = 40$, the result shows greater fluctuations and are not as smooth as those of smaller $N$'s, and thus efforts should be put into large scale optimization.

In the specific implementation, one has to solve nonlinear equations (**) to obtain boundary values. Note that (**) does not necessarily have unique solutions (usually up to 5 real solutions), and thus the initial value for the nonlinear solver is fairly important. If the solver converges to a relatively far position, the surface will appear to have spikes on its boundaries. We finally choose $(s_0, t_0) = (0, 0)$ as initial value for the Broyden algorithm to solve (**), and produces satisfying results as Figure 1.

Another approach to solve the original problem is to derive its corresponding Euler-Lagrange equation from the calculus of variations:

$$\nabla \cdot \left( \frac{\nabla u}{\sqrt{1 + \|\nabla u\|^2}} \right) = 0,$$

which expands to

$$(1 + u_x^2)u_{yy} - 2u_x u_y u_{xy} + (1 + u_y^2)u_{xx} = 0.$$

A finite element approximation to this equation yields $T(U) = 0$, which can be solved by minimizing $\|T(U)\|$. We just mention the feasibility of this approach, and will not discuss it any further here.

## References

[1] J. Nocedal and S. Wright. Numerical Optimization (Second Edition). Springer, 2006.