

# Solution to Assignment 3

*I will appreciate it if you could give me some advice on my assignment!*

December 26, 2018

1. Show that if  $x^*$  is a local minimum of the twice continuously differentiable function  $f : \mathbb{R}^n \mapsto \mathbb{R}$  over the convex set  $X$ , then

$$(x - x^*)' \nabla^2 f(x^*) (x - x^*) \geq 0,$$

for all  $x \in X$  such that  $\langle \nabla f(x^*), x - x^* \rangle = 0$ .

*Proof.* Assume there exists a  $x \in X$  such that  $\langle \nabla f(x^*), x - x^* \rangle = 0$  but

$$(x - x^*)' \nabla^2 f(x^*) (x - x^*) < 0.$$

The idea for contradiction is to apply the Taylor expansion at the point on the secant line between  $x$  and  $x^*$ , and show the value at that point may necessarily be less than the optimal value:

$$\begin{aligned} f(x^* + \varepsilon(x - x^*)) &= f(x^*) + \langle \nabla f(x^*), \varepsilon(x - x^*) \rangle + \frac{1}{2}(\varepsilon(x - x^*))' \nabla^2 f(x^* + s\varepsilon(x - x^*))(\varepsilon(x - x^*)) \\ &= f(x^*) + \varepsilon \langle \nabla f(x^*), x - x^* \rangle + \frac{\varepsilon^2}{2}(x - x^*)' \nabla^2 f(x^* + s\varepsilon(x - x^*))(x - x^*) \end{aligned} \tag{1}$$

for any  $\varepsilon \in (0, 1)$  and some  $s \in [0, 1]$ . Note that  $x^* + \varepsilon(x - x^*)$  is a feasible solution as it is a convex combination of  $x^*$  and  $x$ .

Since  $\nabla f(x^*) = 0$ , equivalently we write (9) as:

$$f(x^* + \varepsilon(x - x^*)) - f(x^*) = \frac{\varepsilon^2}{2}(x - x^*)' \nabla^2 f(x^* + s\varepsilon(x - x^*))(x - x^*) \tag{2}$$

For sufficiently small  $\varepsilon$ , the LHS is nonnegative, since  $x^*$  is local minimum; thus the RHS is non-negative, and thus  $(x - x^*)' \nabla^2 f(x^* + s\varepsilon(x - x^*))(x - x^*)$  is non-negative. By taking the limit  $\varepsilon \rightarrow 0$ , we obtain

$$(x - x^*)' \nabla^2 f(x^*) (x - x^*) \geq 0$$

□

2. Let  $f : \mathbb{R}^n \mapsto \mathbb{R}$  be a twice continuously differentiable function that satisfies

$$m\|y\|^2 \leq y' \nabla^2 f(x) y \leq M\|y\|^2, \quad \forall x, y \in \mathbb{R}^n$$

where  $m$  and  $M$  are some positive scalars. Let also  $X$  be a closed convex set. Show that  $f$  has a unique global minimum  $x^*$  over  $X$ , which satisfies

$$\theta_M(x) \leq f(x) - f(x^*) \leq \theta_m(x), \quad \forall x \in \mathbb{R}^n,$$

where for all  $\delta > 0$ , we denote

$$\theta_\delta(x) = - \inf_{y \in X} \left\{ \langle \nabla f(x), y - x \rangle + \frac{\delta}{2} \|y - x\|^2 \right\}.$$

*Proof.* (i) Firstly we show that  $f$  is strictly convex and *coercive*.  $f$  is strictly convex since  $\nabla^2 f$  is PD. To find the limit  $\lim_{\|x\| \rightarrow \infty} f(x)$ , we make a change of variable first:

$$M = \|x\|, \quad \mathbf{d} = \frac{\|x\|}{x}, \quad g(M) = f(M\mathbf{d})$$

Thus it suffices to compute the limit  $\lim_{M \rightarrow \infty} g(M)$  for fixed  $\mathbf{d}$ . The derivatives of  $g$  are:

$$\begin{aligned} g'(M) &= M \langle \nabla f(M\mathbf{d}), \mathbf{d} \rangle \\ g''(M) &= \langle \nabla f(M\mathbf{d}), \mathbf{d} \rangle + M \mathbf{d}' \nabla^2 f(M\mathbf{d}) \mathbf{d} \\ &\geq M m \|\mathbf{d}\|^2 + \langle \nabla f(M\mathbf{d}), \mathbf{d} \rangle \\ &\geq M m \|\mathbf{d}\|^2 - \|\nabla f(M\mathbf{d})\| \|\mathbf{d}\| \end{aligned}$$

As  $M$  goes sufficiently large,  $g''(M)$  is strictly larger than zero, say  $g''(M) \geq C > 0$ . It follows that

$$g(M) = \iint g''(t) > \iint C = \frac{1}{2} C M^2 + C M + o(1)$$

for sufficiently large  $M$ . Hence,  $g(M) \rightarrow \infty$  as  $M \rightarrow \infty$ , i.e.,  $f(x) \rightarrow \infty$  as  $\|x\| \rightarrow \infty$ .

(ii) Since  $X$  is closed and convex,  $f$  is strictly convex and coercive, by *Weierstrass theorem*,  $f$  has a unique global minimum over any closed convex set, say  $x^*$ .

(iii) For  $\forall x, y \in X \subseteq \mathbb{R}^n$ ,

$$f(y) - f(x) = \langle \nabla f(x), y - x \rangle + \frac{1}{2} (y - x)' \nabla^2 f(z) (y - x),$$

for some  $z$  on secant line between  $x$  and  $y$ . It follows that

$$\langle \nabla f(x), y - x \rangle + \frac{1}{2} m \|y - x\|^2 \leq f(y) - f(x) \leq \langle \nabla f(x), y - x \rangle + \frac{1}{2} M \|y - x\|^2 \quad (3)$$

Taking the infimum of  $y$  over Eq.(3), we derive:

$$\begin{aligned} \inf_{y \in X} \left\{ \langle \nabla f(x), y - x \rangle + \frac{1}{2} m \|y - x\|^2 \right\} &= -\theta_m(x) \\ &\leq \inf_{y \in X} f(y) - f(x) = f(x^*) - f(x) \\ &\leq \inf_{y \in X} \left\{ \langle \nabla f(x), y - x \rangle + \frac{1}{2} M \|y - x\|^2 \right\} \\ &= -\theta_M(x) \end{aligned}$$

Or equivalently,  $\theta_M(x) \leq f(x) - f(x^*) \leq \theta_m(x)$ .

□

3. Let  $\mathbf{A}$  be a  $m \times n$  matrix, let  $\mathbf{c}$  and  $\mathbf{a}_1, \dots, \mathbf{a}_r$  be vectors in  $\mathbb{R}^n$ , let  $b_1, \dots, b_r$  be scalars, and let  $f$  be a convex function satisfying  $\liminf_{\|\mathbf{y}\| \rightarrow \infty} \{ \frac{f(\mathbf{y})}{\|\mathbf{y}\|} \} = \infty$ , Assume that the optimal value of the problem

$$\begin{aligned} \min \quad & f(\mathbf{A}\mathbf{x}) + \mathbf{c}^T \mathbf{x} \\ \text{such that} \quad & \mathbf{a}_j^T \mathbf{x} + b_j \leq 0, \quad j = 1, 2, \dots, r, \end{aligned} \quad (4)$$

is finite. Then the problem has *at least one optimal* solution.

*Proof.* Suppose  $f^*$  is the optimal solution. The feasible region is denoted by:

$$F = \{\mathbf{x} \mid \mathbf{a}_j^T \mathbf{x} + b_j \leq 0, j = 1, \dots, r\}.$$

Set a decreasing sequence  $\{\gamma^k\}$  with limit  $f^*$ , and set

$$S^k := \{\mathbf{x} \in F \mid f(\mathbf{Ax}) + \mathbf{c}^T \mathbf{x} \leq \gamma^k\}$$

Thus the set of optimal solutions is  $\bigcap_{k=0}^{\infty} S^k$ . It suffices to show that all asymptotic sequences corresponding to asymptotic directions are *retractive*.

**Asymptotic Directions are essentially Boundary Directions** For fixed asymptotic pair  $(\{\mathbf{x}^k\}, \mathbf{d})$ , we claim that

$$\mathbf{Ad} = \mathbf{0}, \langle \mathbf{c}, \mathbf{d} \rangle \leq 0 \quad (5)$$

$$\langle \mathbf{a}_j, \mathbf{d} \rangle \leq 0, j = 1, 2, \dots, r \quad (6)$$

- Assume the first equality is not satisfied, then we define  $\mathbf{d}^k = \frac{\mathbf{x}^k}{\|\mathbf{x}^k\|}$ , and decompose  $\mathbf{d}^k = \mathbf{s}^k + \mathbf{t}^k$  with  $\mathbf{s}^k \in \mathcal{N}(\mathbf{A})$  and  $\mathbf{t}^k \in \mathcal{N}(\mathbf{A})^\perp$ . Since  $\mathbf{x}^k = \|\mathbf{x}^k\| \mathbf{d}^k = \|\mathbf{x}^k\| (\mathbf{s}^k + \mathbf{t}^k) \in S^k$ , we have

$$f \left[ \|\mathbf{x}^k\| \mathbf{A}(\mathbf{s}^k + \mathbf{t}^k) \right] + \|\mathbf{x}^k\| \langle \mathbf{c}, \mathbf{s}^k + \mathbf{t}^k \rangle \leq \gamma^k$$

Since  $\mathbf{As}^k = \mathbf{0}$ , equivalently we have

$$f \left[ \|\mathbf{x}^k\| \mathbf{At}^k \right] + \|\mathbf{x}^k\| \langle \mathbf{c}, \mathbf{s}^k + \mathbf{t}^k \rangle \leq \gamma^k \quad (7)$$

Define  $\mathbf{d}, \mathbf{s}, \mathbf{t}$  as the limit of  $\mathbf{d}^k, \mathbf{s}^k, \mathbf{t}^k$  respectively. Since  $\mathbf{At} \neq \mathbf{0}$ , there exists a number  $\delta > 0$  such that  $\|\mathbf{At}\| = \delta > 0$ . As  $\{\mathbf{t}^k\}$  is convergent,  $\{\|\mathbf{At}^k\|\}$  is also convergent, i.e., for sufficiently large  $k$ ,  $\|\mathbf{At}^k\| > 0$ . Hence, dividing  $\|\mathbf{x}^k\| \|\mathbf{At}^k\|$  both sides for (7), we derive

$$\frac{f \left[ \|\mathbf{x}^k\| \mathbf{A}(\mathbf{s}^k + \mathbf{t}^k) \right]}{\|\mathbf{x}^k\| \|\mathbf{At}^k\|} + \frac{\langle \mathbf{c}, \mathbf{s}^k + \mathbf{t}^k \rangle}{\|\mathbf{At}^k\|} \leq \frac{\gamma^k}{\|\mathbf{x}^k\| \|\mathbf{At}^k\|} \quad (8)$$

Taking limit  $k \rightarrow \infty$  both sides, we derive:

$$\lim_{k \rightarrow \infty} \frac{f \left[ \|\mathbf{x}^k\| \mathbf{A}(\mathbf{s}^k + \mathbf{t}^k) \right]}{\|\mathbf{x}^k\| \|\mathbf{At}^k\|} = +\infty, \quad \lim_{k \rightarrow \infty} \frac{\langle \mathbf{c}, \mathbf{s}^k + \mathbf{t}^k \rangle}{\|\mathbf{At}^k\|} = \frac{\langle \mathbf{c}, \mathbf{s} + \mathbf{t} \rangle}{\delta}, \quad \frac{\gamma^k}{\|\mathbf{x}^k\| \|\mathbf{At}^k\|} = 0,$$

the limit for the LHS of (8) is unbounded, while the limit for the RHS of (8) is bounded, which is a contradiction. Hence, we derive  $\mathbf{Ad} = \mathbf{0}$ .

- Due to (8), and  $\frac{f \left[ \|\mathbf{x}^k\| \mathbf{A}(\mathbf{s}^k + \mathbf{t}^k) \right]}{\|\mathbf{x}^k\| \|\mathbf{At}^k\|} \geq 0$  for sufficiently large  $k$ , we obtain:

$$\langle \mathbf{c}, \mathbf{s}^k + \mathbf{t}^k \rangle \leq \|\mathbf{At}^k\| \cdot \frac{f \left[ \|\mathbf{x}^k\| \mathbf{A}(\mathbf{s}^k + \mathbf{t}^k) \right]}{\|\mathbf{x}^k\| \|\mathbf{At}^k\|} + \langle \mathbf{c}, \mathbf{s}^k + \mathbf{t}^k \rangle \leq \frac{\gamma^k}{\|\mathbf{x}^k\|}$$

Taking  $k \rightarrow \infty$  both sides, we imply  $\langle \mathbf{c}, \mathbf{d} \rangle \leq 0$ .

- Similarly,  $\langle \mathbf{a}_j, \mathbf{d}^k \rangle \leq -\frac{b_j}{\|\mathbf{x}^k\|}$  implies  $\langle \mathbf{a}_j, \mathbf{d} \rangle \leq 0, j = 1, 2, \dots, r$

$\mathbf{x}^k - \alpha \mathbf{d}$  satisfies the condition in  $S^k$  Next we show  $\langle \mathbf{c}, \mathbf{d} \rangle = 0$ . For a feasible vector  $\bar{\mathbf{x}}$ , consider  $\tilde{\mathbf{x}} := \bar{\mathbf{x}} + m\mathbf{d}$  for any positive  $m$ , which is also feasible as  $\langle \mathbf{a}_j, \mathbf{d} \rangle \leq 0$ . Then checking the function evaluated at  $\tilde{\mathbf{x}}$ :

$$\begin{aligned} f^* &\leq f(\mathbf{A}\tilde{\mathbf{x}}) + \langle \mathbf{c}, \tilde{\mathbf{x}} \rangle = f(\mathbf{A}\bar{\mathbf{x}} + m\mathbf{A}\mathbf{d}) + \langle \mathbf{c}, \bar{\mathbf{x}} \rangle + m\langle \mathbf{c}, \mathbf{d} \rangle \\ &= \underbrace{f(\mathbf{A}\bar{\mathbf{x}}) + \langle \mathbf{c}, \bar{\mathbf{x}} \rangle}_{\text{bounded}} + m\langle \mathbf{c}, \mathbf{d} \rangle \end{aligned}$$

As  $f^*$  is finite and  $m > 0$ ,  $\langle \mathbf{c}, \mathbf{d} \rangle \leq 0$ , we imply  $\langle \mathbf{c}, \mathbf{d} \rangle = 0$ .

As a result, for fixed  $\mathbf{x}^k$ , the function evaluated at  $\mathbf{x}^k - \alpha \mathbf{d}$  satisfies the condition

$$\begin{aligned} f(\mathbf{A}(\mathbf{x}^k - \alpha \mathbf{d})) + \langle \mathbf{c}, \mathbf{x}^k - \alpha \mathbf{d} \rangle &= f(\mathbf{A}\mathbf{x}^k) + \langle \mathbf{c}, \mathbf{x}^k \rangle - \alpha \langle \mathbf{c}, \mathbf{d} \rangle \\ &= f(\mathbf{A}\mathbf{x}^k) + \langle \mathbf{c}, \mathbf{x}^k \rangle \leq \gamma^k, \forall \alpha, k. \end{aligned}$$

**Feasiblensness of  $\mathbf{x}^k - \alpha \mathbf{d}$**  To show the retractsiveness ( $\mathbf{x}^k - \alpha \mathbf{d} \in S^k$ ), it suffices to choose  $\alpha > 0$  to let  $\mathbf{x}^k - \alpha \mathbf{d} \in F$  for sufficiently large  $k$ , i.e.,

$$\langle \mathbf{a}_j, \mathbf{x}^k - \alpha \mathbf{d} \rangle + b_j \leq 0, \quad j = 1, \dots, r$$

- This is true for  $\forall \alpha > 0$  if  $\langle \mathbf{a}_j, \mathbf{d} \rangle = 0$
- Otherwise, suppose  $\langle \mathbf{a}_j, \mathbf{d} \rangle < -\varepsilon$ . Thus  $\langle \mathbf{a}_j, \mathbf{d}^k \rangle < -\varepsilon$  for sufficiently large  $k$ , i.e.,  $\langle \mathbf{a}_j, \mathbf{x}^k \rangle \leq -\varepsilon \|\mathbf{x}^k\|$ . Combining the unboundness of  $\{\|\mathbf{x}^k\|\}$ , we imply that for sufficiently large  $k$ ,

$$\langle \mathbf{a}_j, \mathbf{x}^k - \alpha \mathbf{d} \rangle + b_j \leq -\varepsilon \|\mathbf{x}^k\| - \alpha \langle \mathbf{a}_j, \mathbf{d} \rangle + b_j < 0$$

The proof is complete. □

# 1 MATLAB Project: Comparison of three 1st-order methods

## 1.1 A copy of code myL1reg2c.m

---

```

function [ x,iter ] = myL1reg2c( A,b,D )
% Input:
%   A: a m*n matrix
%   b: a m*1 vector
%   D: a k*n matrix
% Usage:
%   solve the unconstrained minimization model
%   min phi_sigma(D * x) + mu/2 * ||A*x - b||_2^2
% with
% sigma = 0.05 around
% mu = 0.1 around
% phi_sigma(y) = \sum_{i=1}^k (y_i^2 + sigma)^(1/2)
%% parameters setting
[~,n] = size(D);
global sigma mu alpha
tol_2 = 1e-8;
maxiter = 50000;
%% initial setting
x = sparse(n,1);
x_p = sparse(n,1);
Dx = D*x;
[f,g] = my_ob_nabla(x,Dx,D,A,b,sigma,mu);
gnorm = norm(g);
tol_1 = gnorm * 1e-2;
a_p = 0; a = 1;
%% Iteration Running
for iter = 1:maxiter
    m = 0.5 * (1+sqrt(1+4*a^2)); a_p = a; a = m;
    t = (a_p - 1) / a;
    y = (1+t) * x - t * x_p; Dy = D*y;
    % update function
    x_p = x; [f_try,nabla] = my_ob_nabla(y,Dy,D,A,b,sigma,mu);
    x = y - alpha * nabla;
    f_diff = abs(1 - f_try / f); f = f_try;
    gnorm = norm(nabla);
    if f_diff <= tol_2 && gnorm <= tol_1, break; end
end
end

%% Computing object and nabla
function [object,nabla] = my_ob_nabla(x,Dx,D,A,b,sigma,mu)

K = sqrt((Dx).^2+sigma);
Y = A*x - b;
object = sum(K) + mu/2 * norm(Y)^2;
if nargin == 1, return;end
nabla = ((Dx./K)' * D)' + (mu*Y' * A)';

end

```

---

## 1.2 Matlab screen printout for the run results of test2 L1.m

```

Command Window
>>
>>
>>
>> test2_L1
runcase (0 to 4) = 0

[m, n, k] = [500, 1000, 100]
[sigma mu alpha] = [1.00e-02 1.00e-01 3.43e-03]

===== yzL1reg2a =====
p = 1: rel_err: 6.832738e-03 time: 0.3139 iter: 1012
p = 2: rel_err: 6.414191e-03 time: 4.0465 iter: 12473

===== yzL1reg2b =====
p = 1: rel_err: 6.890278e-03 time: 2.1551 iter: 14117
p = 2: rel_err: 6.415389e-03 time: 26.6454 iter: 175101

===== yzL1reg2c =====
p = 1: rel_err: 6.758495e-03 time: 0.1601 iter: 1040
p = 2: rel_err: 6.370492e-03 time: 0.7393 iter: 4868

**** Error ****

    0.0068    0.0064
    0.0069    0.0064
    0.0068    0.0064

**** Time ****

    0.3139    4.0465
    2.1551   26.6454
    0.1601    0.7393
  
```

Figure 1: Printout from Run Case 1

```
Command Window
>>
>>
>>
>>
>>
>>
>>
>>
>> test2_L1
runcase (0 to 4) = 1

[m, n, k] = [500, 1000, 100]
[sigma mu alpha] = [1.00e-02 1.00e-01 3.43e-03]

===== myL1reg2b =====
p = 1: rel_err: 6.889918e-03 time: 2.0794 iter: 14136
p = 2: rel_err: 6.415404e-03 time: 29.7674 iter: 174910

===== myL1reg2c =====
p = 1: rel_err: 6.955403e-03 time: 0.2321 iter: 1256
p = 2: rel_err: 6.357816e-03 time: 0.6178 iter: 3722

**** Error ****

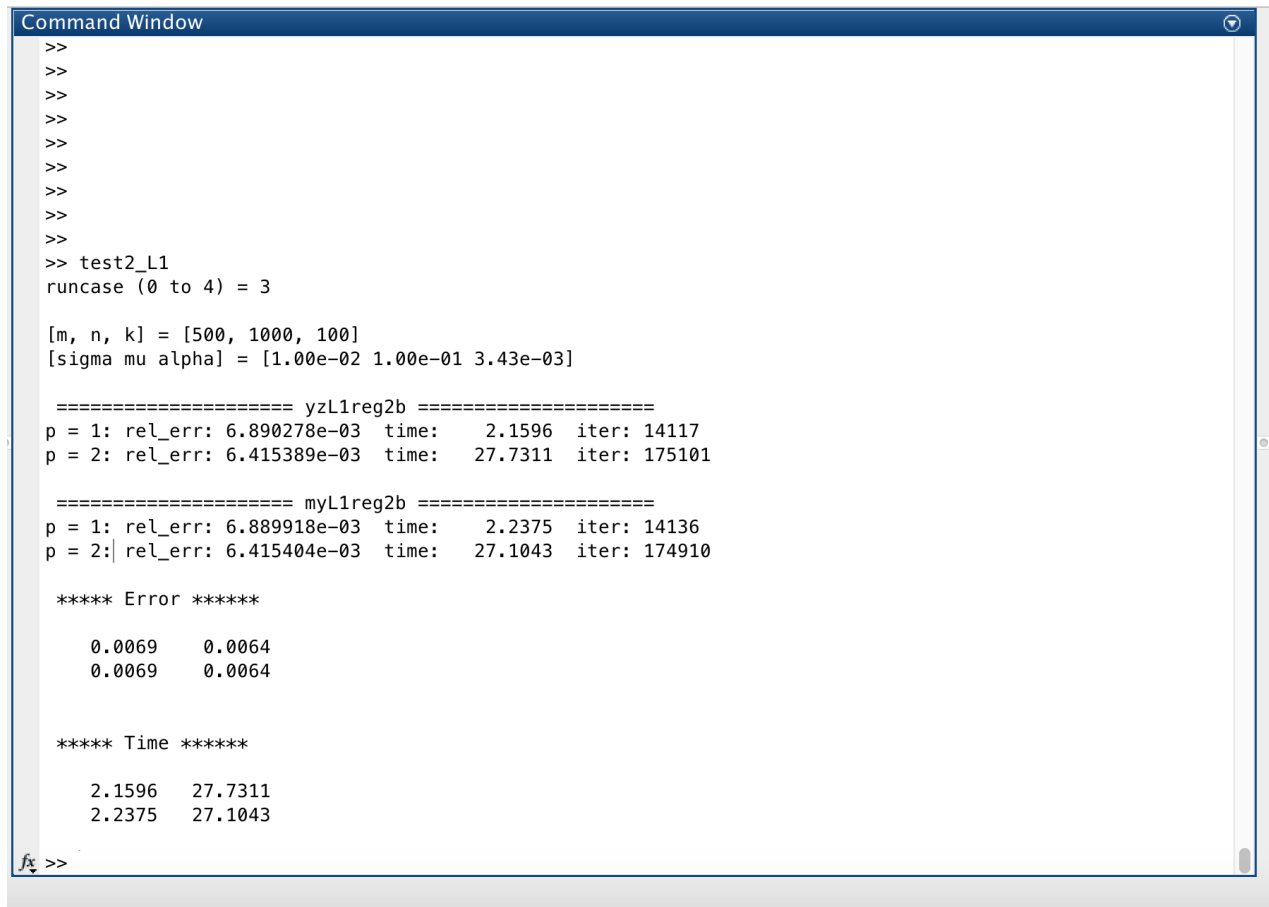
0.0069 0.0064
0.0070 0.0064

**** Time ****

2.0794 29.7674
0.2321 0.6178

fx >>
```

Figure 2: Printout from Run Case 2



```
Command Window
>>
>>
>>
>>
>>
>>
>>
>>
>> test2_L1
runcase (0 to 4) = 3

[m, n, k] = [500, 1000, 100]
[sigma mu alpha] = [1.00e-02 1.00e-01 3.43e-03]

===== yzL1reg2b =====
p = 1: rel_err: 6.890278e-03 time: 2.1596 iter: 14117
p = 2: rel_err: 6.415389e-03 time: 27.7311 iter: 175101

===== myL1reg2b =====
p = 1: rel_err: 6.889918e-03 time: 2.2375 iter: 14136
p = 2: rel_err: 6.415404e-03 time: 27.1043 iter: 174910

**** Error ****

    0.0069    0.0064
    0.0069    0.0064

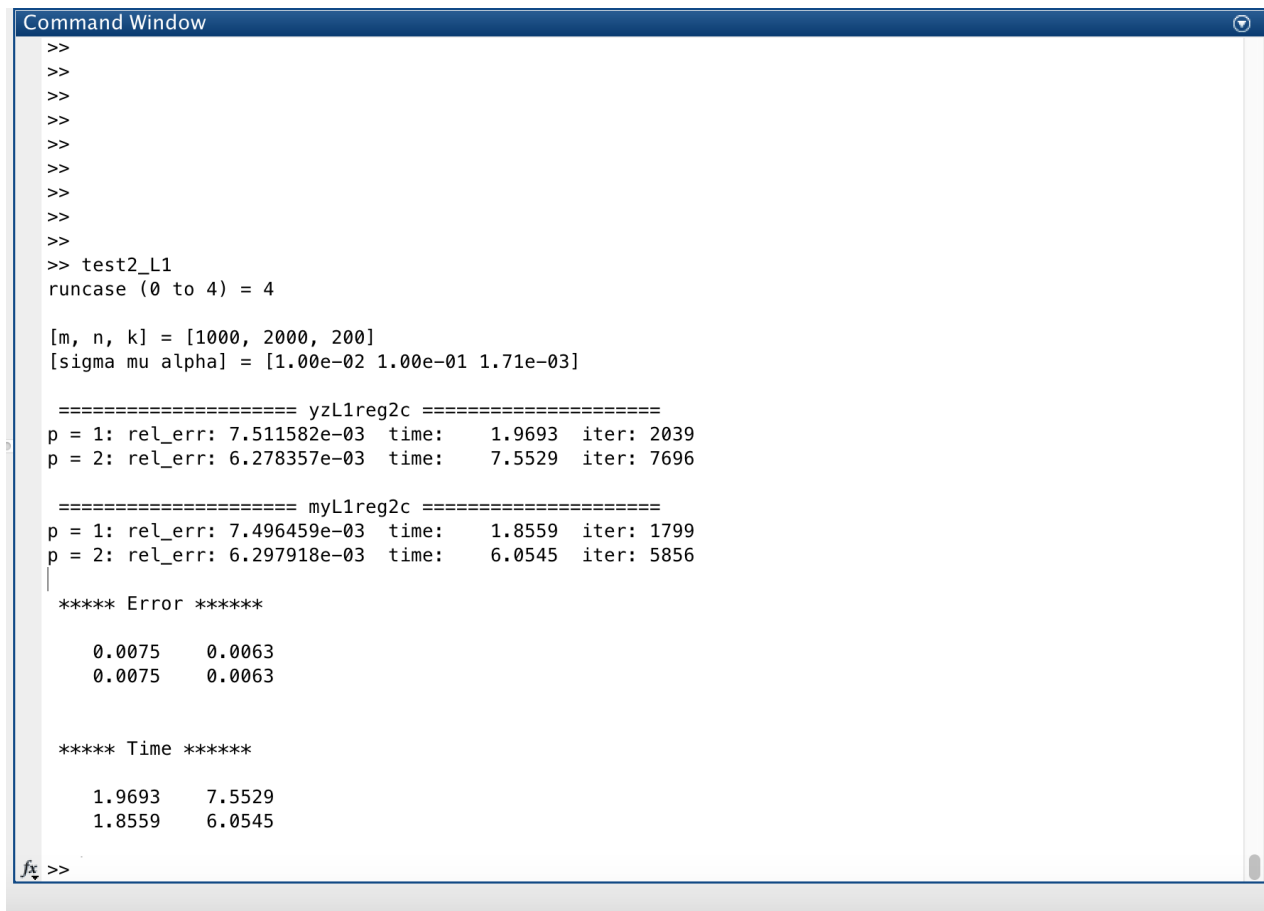
**** Time ****

    2.1596    27.7311
    2.2375    27.1043

fx >>
```

Figure 3: Printout from Run Case 3





```
Command Window
>>
>>
>>
>>
>>
>>
>>
>>
>> test2_L1
runcase (0 to 4) = 4

[m, n, k] = [1000, 2000, 200]
[sigma mu alpha] = [1.00e-02 1.00e-01 1.71e-03]

===== yzL1reg2c =====
p = 1: rel_err: 7.511582e-03 time: 1.9693 iter: 2039
p = 2: rel_err: 6.278357e-03 time: 7.5529 iter: 7696

===== myL1reg2c =====
p = 1: rel_err: 7.496459e-03 time: 1.8559 iter: 1799
p = 2: rel_err: 6.297918e-03 time: 6.0545 iter: 5856
|
**** Error ****
    0.0075    0.0063
    0.0075    0.0063

**** Time ****
    1.9693    7.5529
    1.8559    6.0545
fx >>
```

Figure 4: Printout from Run Case 4

### 1.3 Output figure on run case number 4

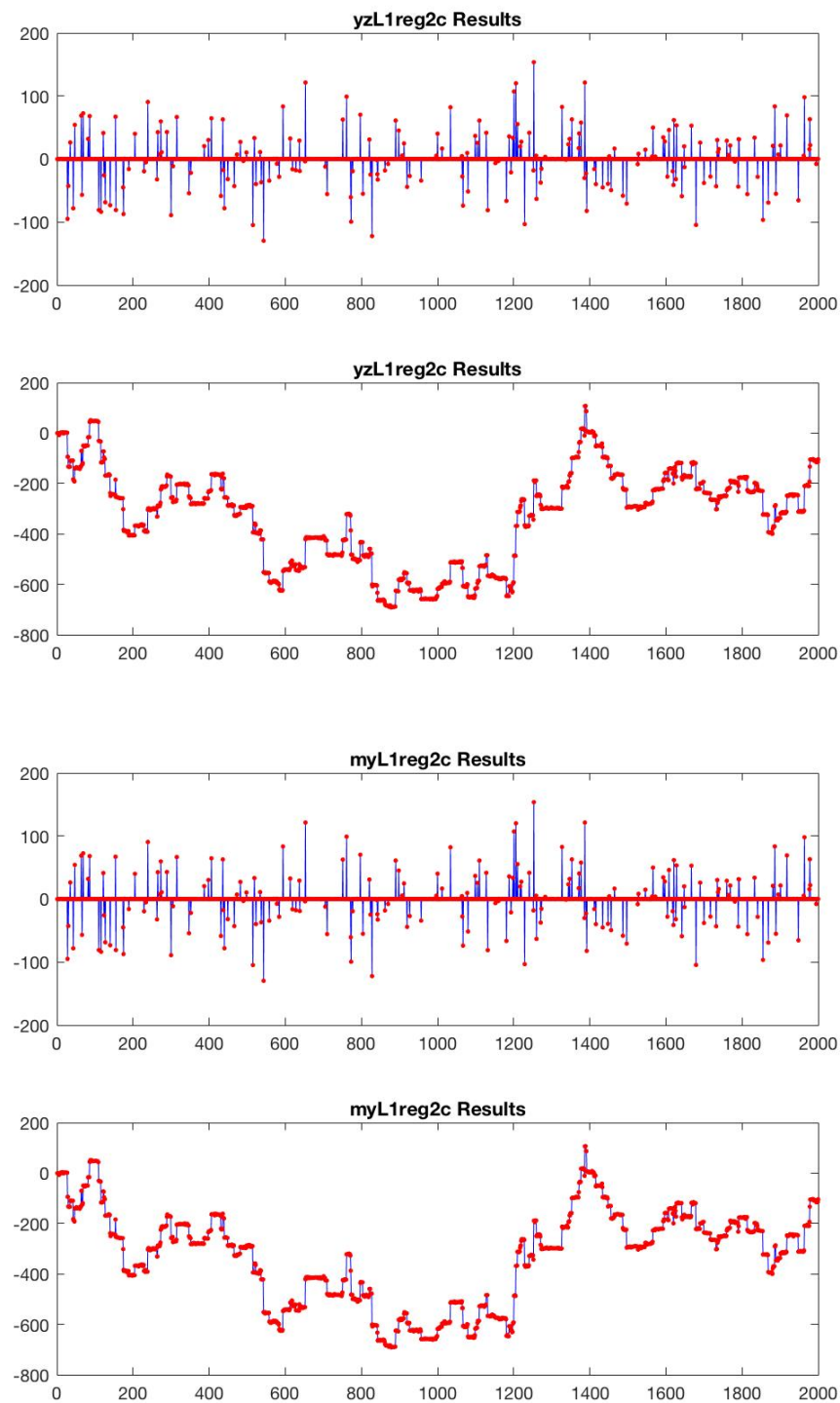


Figure 5: Output figure for Run Case 4

## 1.4 Short summary about my observations and experiments

**Introduction** This project is about the comparison of three first order methods to solve an unconstrained problem. The first and the second method has different step-size, i.e., the first one is step-size variant while the second is invariant. The third method is optimal among all first order methods, i.e., it will obtain fastest speed in worse case.

**Comments about step-size-invariant method** Although the step-size-invariant method also guarantees the convergence and the error using this method is also very small in this project, it has too small step-size ( $\frac{1}{L}$ ) such that the convergence rate is very small. In large scale optimization, every iteration for computing the gradient is expansive, thus the step-size-invariant method is not recommended in this case as it requires many iterations, i.e., too much computation and time resources.

**Comments about step-size variant method** If applying step-size variant method, there may also rise some problems. First, the choice of step-size may spend lots of time. Second, the step-size variant method only choose  $\{\nabla g(x_1), \dots, \nabla g(x_k), \dots\}$  sequentially as the descent direction. In practice, such method may not receive satisfying results as well.

**Comments about accelerated method** The Nesterovs optimal method is an alternative way to rescue the step-size invariant method, i.e., it has fixed step-size, but for each iteration it also use part of the direction from last iteration. With combination of two directions (last and this iteration), this method has faster convergence rate than that with only one direction.

## 2 MATLAB Project: Newtons Method

### 2.1 Three Copies of my Code

---

```
function [x, iter ] = myNewton(func, x0, tol, maxit, varargin )
% Implement Newton's method using analytic Jacobian
% Input:
%   func: function handle
%   x0: initial guess
%   tol: tolerance
%   maxit: maximum iteration
%   varargin - parameters
% Output:
%   x: optimal solution
%   iter: total number of iterations
%% First Iteration
x = x0;
[g,H] = feval(func, x, varargin{:});
nrmg0 = norm(g);
x = x - H \ g;
fprintf('iter: %3d nrmg/nrmg0 = %6.2e\n',1,1)
%% Remaining iterations
for iter = 2:maxit
    [g, H] = feval(func, x, varargin{:}); % get gradient g and Hessian H
    fraction = norm(g)/nrmg0;
    fprintf('iter: %3d nrmg/nrmg0 = %6.2e\n',iter,fraction)
    if fraction <= tol,break;end
    x = x - H \ g;
end

end
```

---

```
function [ x,iter ] = myFdNewton( func,x0,tol,maxit,varargin )
% Implement Newton's method using finite difference approximation
% Input:
%   func: function handle
%   x0: initial guess
%   tol: tolerance
%   maxit: maximum iteration
%   varargin - parameters
% Output:
%   x: optimal solution
%   iter: total number of iterations
%% First Iteration
x = x0;
g = feval(func,x,varargin{:});nrmg0 = norm(g);
fprintf('iter: %3d nrmg/nrmg0 = %6.2e\n',1,1)
H = feval('myHessian',func,x,g,varargin{:});
x = x - H \ g;
%% Remaining iterations
for iter = 2:maxit
    g = feval(func, x, varargin{:}); % get gradient g and Hessian H
```

---

```

    fraction = norm(g)/nrmg0;
    fprintf('iter: %3d nrmg/nrmg0 = %.2e\n',iter,fraction)
    if fraction <= tol,break;end
    H = feval('myHessian',func,x,g,varargin{:});
    x = x - H \ g;
end

end

function H = myHessian(func,x,gx,varargin)
%delta = 1e-9;
m = length(gx); n = length(x);
H = zeros(m,n);
for j = 1:n
    epsilon = 1e-9 * max(1,max(1,abs(x(j))) * sign(x(j)));
    x(j) = x(j) + epsilon;
    g = feval(func, x, varargin{:});
    x(j) = x(j) - epsilon;
    H(:,j) = (g - gx) / epsilon;
end

end

function [ g,H,f ] = myQuartic( x,A,u,mu )
% Output:
%     g: gradient, g = (x'*A*x - 1/4*u'*A*u) * A*x + mu * (x - u)
%     H: Hessian, H = (x'*A*x - 1/4*u'*A*u) * A + 2*A*x*x'*A + mu * eye(n)
%     f: function itself,
%     f = 1/4*(x'*A*x - 1/4*u'*A*u) + mu/2 * ||x - u||_2^2

n = length(x);

Ax = A*x;
quartic = x'*Ax - u'*A*u/4;
y = x - u;

%% Evaluate g
g = quartic * Ax + mu * y;
if nargin < 2;return;end

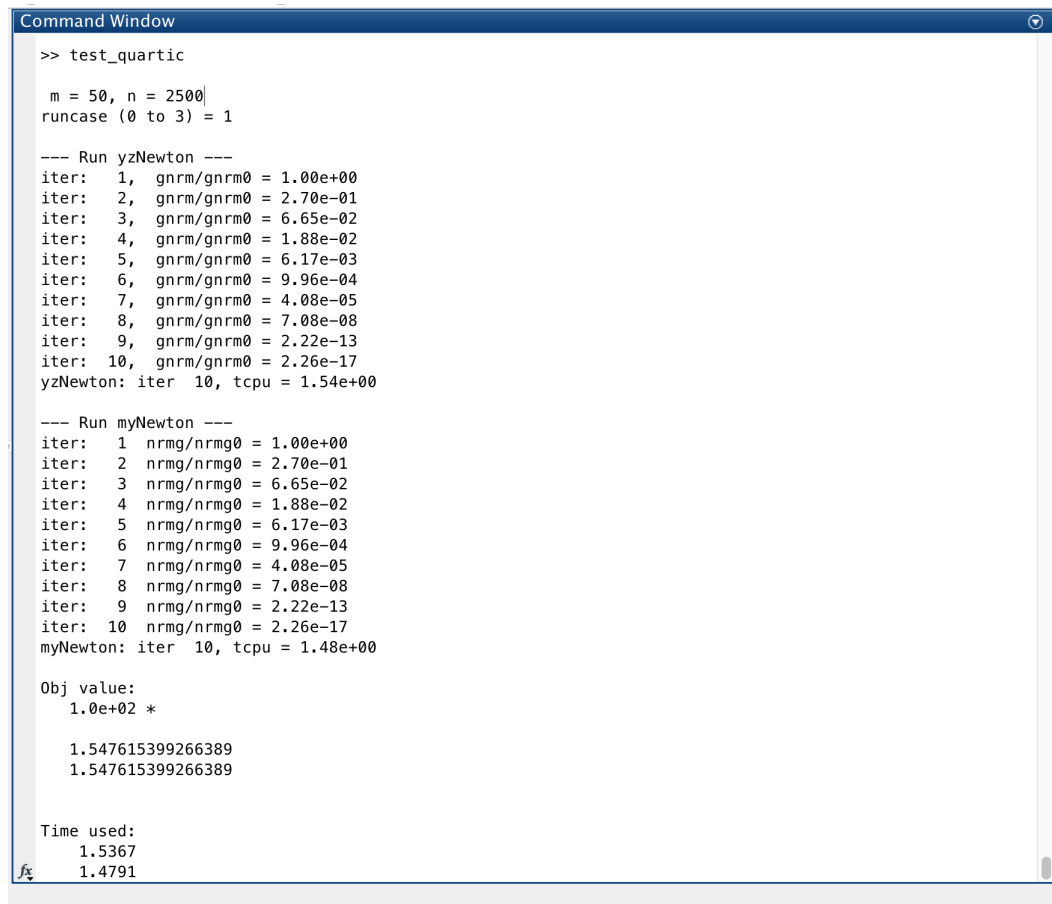
%% Evaluate H
H = quartic * A + 2 * (Ax * Ax') + mu * speye(n);
if nargin < 3;return;end

%% Evaluate f
f = quartic/4 + mu/2 * (y'*y);

end

```

## 2.2 MATLAB Screen Printout for the run results of test quartic.m



```
Command Window

>> test_quartic

m = 50, n = 2500
runcase (0 to 3) = 1

--- Run yzNewton ---
iter: 1, gnrm/gnrm0 = 1.00e+00
iter: 2, gnrm/gnrm0 = 2.70e-01
iter: 3, gnrm/gnrm0 = 6.65e-02
iter: 4, gnrm/gnrm0 = 1.88e-02
iter: 5, gnrm/gnrm0 = 6.17e-03
iter: 6, gnrm/gnrm0 = 9.96e-04
iter: 7, gnrm/gnrm0 = 4.08e-05
iter: 8, gnrm/gnrm0 = 7.08e-08
iter: 9, gnrm/gnrm0 = 2.22e-13
iter: 10, gnrm/gnrm0 = 2.26e-17
yzNewton: iter 10, tcpu = 1.54e+00

--- Run myNewton ---
iter: 1 nrmg/nrmg0 = 1.00e+00
iter: 2 nrmg/nrmg0 = 2.70e-01
iter: 3 nrmg/nrmg0 = 6.65e-02
iter: 4 nrmg/nrmg0 = 1.88e-02
iter: 5 nrmg/nrmg0 = 6.17e-03
iter: 6 nrmg/nrmg0 = 9.96e-04
iter: 7 nrmg/nrmg0 = 4.08e-05
iter: 8 nrmg/nrmg0 = 7.08e-08
iter: 9 nrmg/nrmg0 = 2.22e-13
iter: 10 nrmg/nrmg0 = 2.26e-17
myNewton: iter 10, tcpu = 1.48e+00

Obj value:
1.0e+02 *

1.547615399266389
1.547615399266389

Time used:
1.5367
1.4791
```

Figure 6: Printout from Run Case 1

```
Command Window

>> test_quartic

m = 50, n = 2500
runcase (0 to 3) = 2

--- Run yzFdNewton ---
iter: 1  nrmg/nrmg0 = 1.00e+00
iter: 2  nrmg/nrmg0 = 2.70e-01
iter: 3  nrmg/nrmg0 = 6.65e-02
iter: 4  nrmg/nrmg0 = 1.88e-02
iter: 5  nrmg/nrmg0 = 6.17e-03
iter: 6  nrmg/nrmg0 = 9.96e-04
iter: 7  nrmg/nrmg0 = 4.08e-05
iter: 8  nrmg/nrmg0 = 7.08e-08
iter: 9  nrmg/nrmg0 = 2.26e-13
iter: 10 nrmg/nrmg0 = 2.19e-17
yzFdNewton: iter 10, tcpu = 2.82e+00

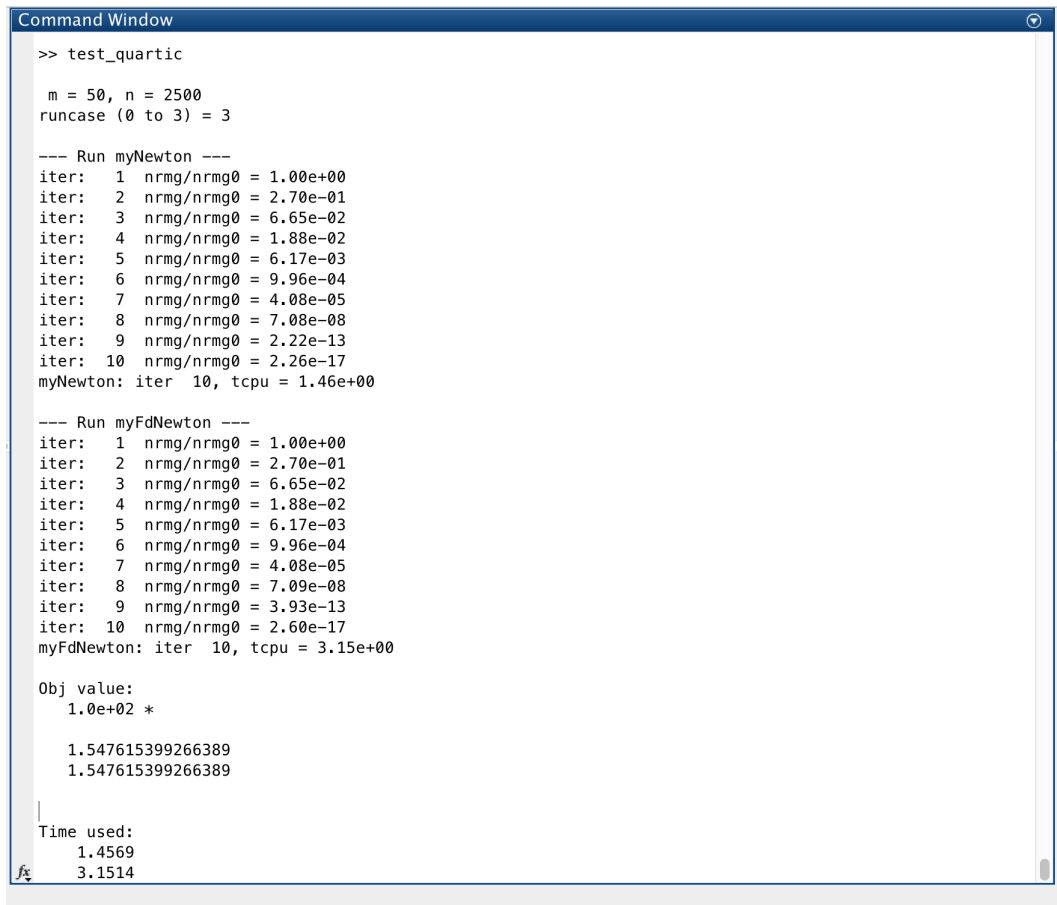
--- Run myFdNewton ---
iter: 1  nrmg/nrmg0 = 1.00e+00
iter: 2  nrmg/nrmg0 = 2.70e-01
iter: 3  nrmg/nrmg0 = 6.65e-02
iter: 4  nrmg/nrmg0 = 1.88e-02
iter: 5  nrmg/nrmg0 = 6.17e-03
iter: 6  nrmg/nrmg0 = 9.96e-04
iter: 7  nrmg/nrmg0 = 4.08e-05
iter: 8  nrmg/nrmg0 = 7.09e-08
iter: 9  nrmg/nrmg0 = 3.93e-13
iter: 10 nrmg/nrmg0 = 2.60e-17
myFdNewton: iter 10, tcpu = 2.81e+00

Obj value:
1.0e+02 *

1.547615399266389
1.547615399266389

Time used:
2.8175
2.8125
```

Figure 7: Printout from Run Case 2



```
Command Window

>> test_quartic

m = 50, n = 2500
runcase (0 to 3) = 3

--- Run myNewton ---
iter: 1  nrmg/nrmg0 = 1.00e+00
iter: 2  nrmg/nrmg0 = 2.70e-01
iter: 3  nrmg/nrmg0 = 6.65e-02
iter: 4  nrmg/nrmg0 = 1.88e-02
iter: 5  nrmg/nrmg0 = 6.17e-03
iter: 6  nrmg/nrmg0 = 9.96e-04
iter: 7  nrmg/nrmg0 = 4.08e-05
iter: 8  nrmg/nrmg0 = 7.08e-08
iter: 9  nrmg/nrmg0 = 2.22e-13
iter: 10 nrmg/nrmg0 = 2.26e-17
myNewton: iter 10, tcpu = 1.46e+00

--- Run myFdNewton ---
iter: 1  nrmg/nrmg0 = 1.00e+00
iter: 2  nrmg/nrmg0 = 2.70e-01
iter: 3  nrmg/nrmg0 = 6.65e-02
iter: 4  nrmg/nrmg0 = 1.88e-02
iter: 5  nrmg/nrmg0 = 6.17e-03
iter: 6  nrmg/nrmg0 = 9.96e-04
iter: 7  nrmg/nrmg0 = 4.08e-05
iter: 8  nrmg/nrmg0 = 7.09e-08
iter: 9  nrmg/nrmg0 = 3.93e-13
iter: 10 nrmg/nrmg0 = 2.60e-17
myFdNewton: iter 10, tcpu = 3.15e+00

Obj value:
1.0e+02 *

1.547615399266389
1.547615399266389

Time used:
1.4569
3.1514
```

Figure 8: Printout from Run Case 3



## 2.3 MATLAB Plots generated by *my* codes

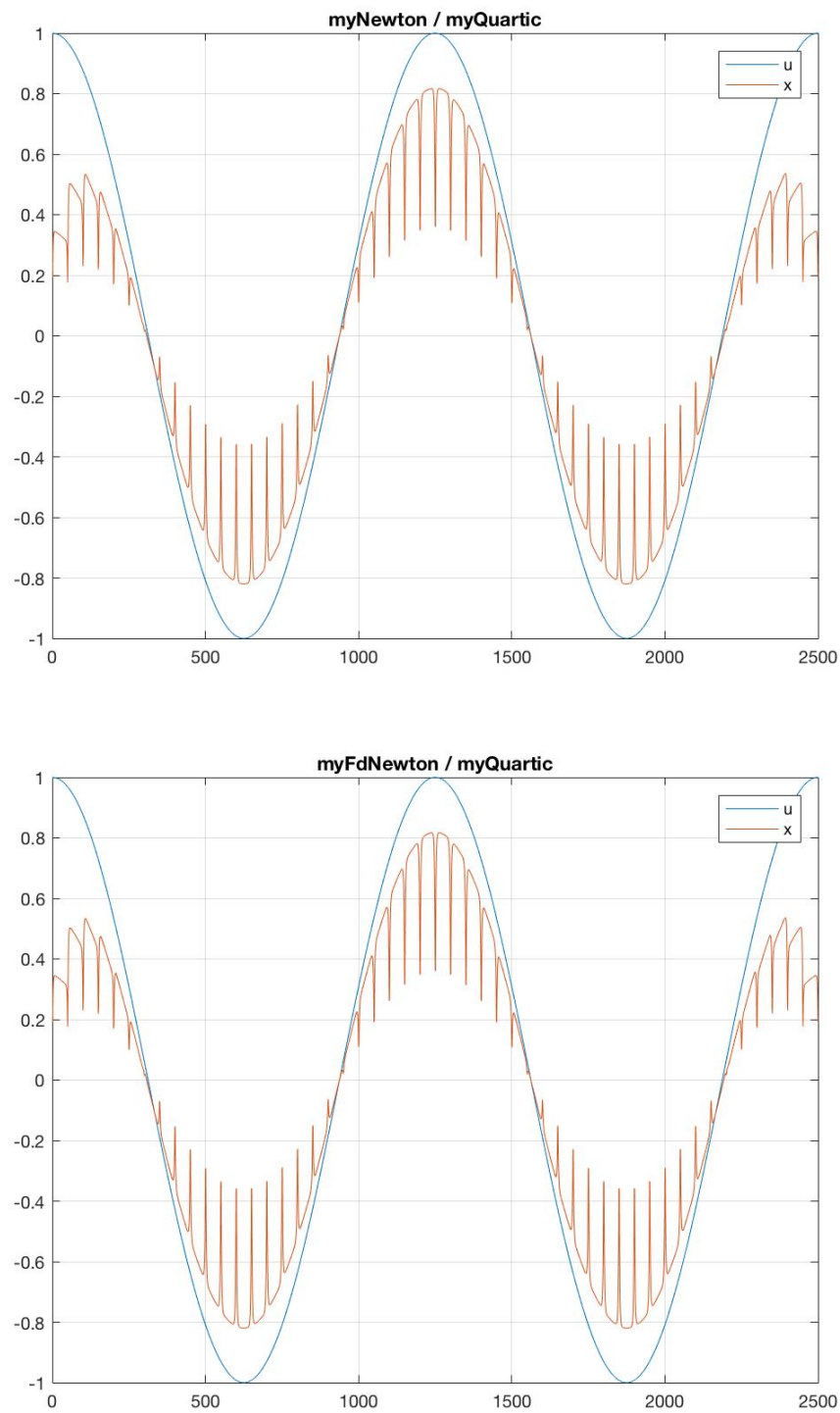


Figure 9: MATLAB plots generated by *my* codes (run case = 4)

## 2.4 Short summary about my experiments and observations

**Introduction** This project is to solve an unconstrained optimization problem using the pure Newton's methods. The first one computes the Hessian matrix using its analytic form, the second one computes it using finite difference method.

**Computation of Hessian matrix is expansive** We observe that the method using analytic form of Hessian matrix will obtain a faster speed. That's because the numerical approximation of Hessian matrix requires too many iteration, and thus the computation of Hessian matrix in each for loop is very expansive than that of analytic form.

**Comments about the implementation of the quartic function's evaluations** For the myQuartic.m file, the stop command nargout is necessary, since when calling this function, we may not necessarily require the computation of gradient or Hessian matrix, and the computation cost for which is expansive.

**Avoid unnecessary computation** During Newton's method implementation, we should try to avoid the update of function as much as possible, since every update requires the inverse computation of a large dimension matrix. One trick is to put the stop criteria in front of the update command, and therefore the update will stop timely once the stopping criteria is satisfied.

**Avoid loss of accuracy during finite difference method** Two comments about finite difference method to approximate derivative. First, the  $h$  for computing  $[f(x \pm h) - f(x)]/h$  should have reasonable scale relative to  $x$ , hence we set  $h = 10^{-9} \times \max\{1, |x|\}$ ; second, when  $x$  and  $h$  have different sign,  $x + h$  will loss accuracy due to rounding error in computer. Hence, the approximate of derivative in computer should be modified as  $[f(x + h * \text{sign}(x)) - f(x)]/h$ .

## 2.5 Derivation and proof for $\mu$ condition

Since  $f$  is continuously differentiable,  $f$  is convex iff  $\nabla^2 f(x) \succeq 0, \forall x$ . We find that:

$$\begin{aligned}\nabla f(x) &= (x^T A x - \frac{1}{4} u^T A u) * A x + \mu(x - u) \\ \nabla^2 f(x) &= (x^T A x - \frac{1}{4} u^T A u) * A + A x * (2x^T A^T) + \mu * I \\ &= (x^T A x - \frac{1}{4} u^T A u) * A + 2A x x^T A + \mu * I\end{aligned}$$

For any  $s \in \mathbb{R}^n$ , we want to choose  $\mu$  such that  $s^T \nabla^2 f(x) s \geq 0$ , or equivalently,

$$(x^T A x - \frac{1}{4} u^T A u) * (s^T A s) + s^T A x * (2x^T A^T) s + \mu * s^T s \geq 0, \forall x, \forall s \quad (9)$$

We find that  $(x^T A x) * (s^T A s) + s^T A x * (2x^T A^T) s \geq 0$  for  $\forall x$ , and choosing  $x = 0$ ,  $(x^T A x) * (s^T A s) + s^T A x * (2x^T A^T) s = 0$ , i.e., we obtain the minimum value of LHS of (9):

$$\min_x \{ (x^T A x - \frac{1}{4} u^T A u) * (s^T A s) + s^T A x * (2x^T A^T) s + \mu * s^T s \} = -\frac{1}{4} (u^T A u) * (s^T A s) + \mu * s^T s \geq 0, \forall s$$

Or equivalently,

$$\mu * s^T s \geq \frac{1}{4} (u^T A u) * (s^T A s), \forall s \implies \mu \geq \frac{1}{4} (u^T A u) * \left( \frac{s^T}{\|s\|} A \frac{s}{\|s\|} \right) \forall s$$

Note that  $\max_s \left( \frac{s^T}{\|s\|} A \frac{s}{\|s\|} \right) = \lambda_{\max}(A)$ , which follows that

$$\mu \geq \max_s \frac{1}{4} (u^T A u) * \left( \frac{s^T}{\|s\|} A \frac{s}{\|s\|} \right) = \frac{\lambda_{\max}(A)}{4} (u^T A u),$$

i.e.,  $\mu \geq \frac{\lambda_{\max}(A)}{4} (u^T A u)$  guarantees convexity of  $f(x)$ .