

Solution to Assignment 2

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

December 26, 2018

1. Let f be twice continuously differentiable. Suppose that x^* is a local minimum such that for all x in an open sphere S centered at x^* , we have, for some $m > 0$,

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

Show that for every $x \in S$, we have

$$\|x - x^*\| \leq \frac{\|\nabla f(x)\|}{m}, \quad \|f(x) - f(x^*)\| \leq \frac{\|\nabla f(x)\|^2}{2m}$$

Proof. The idea for the estimation of $\|x - x^*\|$ is to build the estimation of $\nabla f(x) - \nabla f(x^*)$:

$$\nabla f(x) = \nabla f(x^*) + \int_0^1 \nabla^2 f(x^* + t(x - x^*))(x - x^*) dt \quad (1a)$$

$$= \int_0^1 \nabla^2 f(x^* + t(x - x^*))(x - x^*) dt \quad (1b)$$

Leftmultiplying $(x - x^*)'$ both sides, we obtain:

$$\langle \nabla f(x), (x - x^*) \rangle = (x - x^*)' * \int_0^1 \nabla^2 f(x^* + t(x - x^*))(x - x^*) dt \quad (1c)$$

$$= \int_0^1 (x - x^*)' \nabla^2 f(x^* + t(x - x^*))(x - x^*) dt \quad (1d)$$

$$\geq \int_0^1 m\|x - x^*\|^2 dt = m\|x - x^*\|^2 \quad (1e)$$

By applying Cauchy-Schwarz inequality for LHS, we derive:

$$\|\nabla f(x)\| \cdot \|x - x^*\| \geq \langle \nabla f(x), (x - x^*) \rangle \geq m\|x - x^*\|^2,$$

by re-arranging terms we imply $\|x - x^*\| \leq \frac{\|\nabla f(x)\|}{m}$.

The estimation of $\|f(x) - f(x^*)\|$ relies on the Taylor expansion: Expanding $f(x^*)$ into 2nd order, we obtain:

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

for some z between x^* and x . Since z is still in the sphere S , we obtain:

$$f(x^*) - f(x) \geq \langle \nabla f(x), (x^* - x) \rangle + \frac{m}{2}\|x^* - x\|^2 \quad (2b)$$

Note that the RHS has a lower bound:

$$\langle \nabla f(x), (x^* - x) \rangle + \frac{m}{2} \|x^* - x\|^2 \geq \min_y \langle \nabla f(x), (y - x) \rangle + \frac{m}{2} \|y - x\|^2 \quad (2c)$$

Solving the quadratic minimization, i.e., taking the derivative w.r.t. y leads to $m(y - x) + \nabla f(x) = 0 \implies y^* = x - \frac{\nabla f(x)}{m}$ and

$$\min_y \langle \nabla f(x), (y - x) \rangle + \frac{m}{2} \|y - x\|^2 = -\frac{1}{2m} \|\nabla f(x)\|^2 \quad (2d)$$

Substituting (??) into (??), we derive:

$$\begin{aligned} f(x^*) - f(x) &\geq \langle \nabla f(x), (x^* - x) \rangle + \frac{m}{2} \|x^* - x\|^2 \\ &\geq -\frac{1}{2m} \|\nabla f(x)\|^2 \end{aligned}$$

Or equivalently,

$$\|f(x^*) - f(x)\| = f(x) - f(x^*) \leq \frac{1}{2m} \|\nabla f(x)\|^2 \quad (2e)$$

since $f(x^*) = \inf_x f(x)$.

□

2. Suppose that a vector sequence $\{\mathbf{e}^k\}$ satisfies

$$\|\mathbf{e}^{k+1} - \mathbf{e}^k\| \leq \beta \|\mathbf{e}^k - \mathbf{e}^{k-1}\|, \quad \forall k \geq \bar{k}$$

where \bar{k} is a positive integer and $\beta \in (0, 1)$ is a scalar. Show that $\{\mathbf{e}^k\}$ converges to some vector \mathbf{e}^* linearly, and in fact we have

$$\|\mathbf{e}^k - \mathbf{e}^*\| \leq q \beta^k$$

for some scalar q and all k .

Proof. • Firstly we show that $\{\mathbf{e}^k\}$ is Cauchy. For $\forall k \geq \bar{k}$, we have:

$$\|\mathbf{e}^{k+1} - \mathbf{e}^k\| \leq \beta \|\mathbf{e}^k - \mathbf{e}^{k-1}\| \leq \beta^2 \|\mathbf{e}^{k-1} - \mathbf{e}^{k-2}\| \leq \dots \leq \beta^{k+1-\bar{k}} \|\mathbf{e}^{\bar{k}} - \mathbf{e}^{\bar{k}-1}\|$$

Hence, for $\forall m, n \geq N \geq \bar{k}$, we have: (suppose $m \geq n$ w.l.o.g.)

$$\begin{aligned} \|\mathbf{e}^m - \mathbf{e}^n\| &\leq \|\mathbf{e}^m - \mathbf{e}^{m-1}\| + \dots + \|\mathbf{e}^{n+1} - \mathbf{e}^n\| \\ &\leq \left(\beta^{m-\bar{k}} + \beta^{m-1-\bar{k}} + \dots + \beta^{n+1-\bar{k}} \right) \|\mathbf{e}^{\bar{k}} - \mathbf{e}^{\bar{k}-1}\| \\ &= \beta^{-\bar{k}} \|\mathbf{e}^{\bar{k}} - \mathbf{e}^{\bar{k}-1}\| \sum_{i=n+1}^m \beta^i \end{aligned}$$

As N goes sufficiently large, we can make the partial series $\sum_{i=n+1}^m \beta^i$ arbitrarily small. Therefore, $\|\mathbf{e}^m - \mathbf{e}^n\| \rightarrow 0$ as $N \rightarrow \infty$, hence $\{\mathbf{e}^k\}$ is Cauchy.

- For any $m \geq k \geq \bar{k}$, by applying the same trick,

$$\|\mathbf{e}^m - \mathbf{e}^k\| \leq \beta^{-\bar{k}} \|\mathbf{e}^{\bar{k}} - \mathbf{e}^{\bar{k}-1}\| \sum_{i=k+1}^m \beta^i \quad (3)$$

Since $\{\mathbf{e}^k\}$ is Cauchy, it has a limit, say, $\lim_{k \rightarrow \infty} \mathbf{e}^k = \mathbf{e}^*$. Taking limit both sides for (??) as $m \rightarrow \infty$, we obtain that for $\forall k \geq \bar{k}$,

$$\|\mathbf{e}^k - \mathbf{e}^*\| \leq \beta^{-\bar{k}} \|\mathbf{e}^{\bar{k}} - \mathbf{e}^{\bar{k}-1}\| \sum_{i=k+1}^{\infty} \beta^i = \beta^{-\bar{k}} \|\mathbf{e}^{\bar{k}} - \mathbf{e}^{\bar{k}-1}\| \frac{\beta^{k+1}}{1 - \beta} := M\beta^k \quad (4)$$

with $M = \frac{\beta^{-\bar{k}+1}}{1-\beta} \|\mathbf{e}^{\bar{k}} - \mathbf{e}^{\bar{k}-1}\|$.

We define $M_1 := \max_{1 \leq k < \bar{k}} \frac{\|\mathbf{e}^k - \mathbf{e}^*\|}{\beta^k}$, immediately we obtain:

$$\|\mathbf{e}^k - \mathbf{e}^*\| \leq M_1 \beta^k, \quad 1 \leq k < \bar{k} \quad (5)$$

Combining (??) and (??), we imply for any k , the inequality holds:

$$\|\mathbf{e}^k - \mathbf{e}^*\| \leq q\beta^k,$$

where $q := \max\{M, M_1\}$.

□

MATLAB Code Copy

Code Copy for LP

```
function x = myL1reg0(A, b, D)
% This function solves the optimization problem
% Input:
%   A: a m*n matrix
%   b: a m*1 vector
%   D: a k*n matrix
% minimize sum_{i=1}^m t_i
% such that A * x = b
% for i = 1:k,
%   - t_i \le \sum_{j=1}^n d_{ij}x_j \le t_i
% construct decision variable X = [x_1,...,x_n,s_1,...,s_m]';
%% Estimate size
[m,n] = size(A);
[k,~] = size(D);

%% Construct f, Aineq, bineq, Aeq, beq
f(n+1:n+k) = 1;
Aineq = [D,-1 * eye(k); -1 * D, -1 * eye(k)];
bineq = zeros(k+k,1);
Aeq = [A,zeros(m,k)];
beq = b;
%% Use Linprog to solve optimization
options = optimoptions('linprog','Algorithm','interior-point','MaxIter',20,...
    'OptimalityTolerance',5e-4,'ConstraintTolerance',1e-4,'Display','off');
X = linprog(f,Aineq,bineq,Aeq,beq,[],[],options);

x = X(1:n);

end
```

Code Copy for SD

```
function [x,iter] = myL1reg1(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
sigma = 5e-2; mu = 1e-1;
object = @(x,Dx) sum(sqrt(((Dx).^2+sigma))) + mu/2 * norm(A*x - b)^2;
```

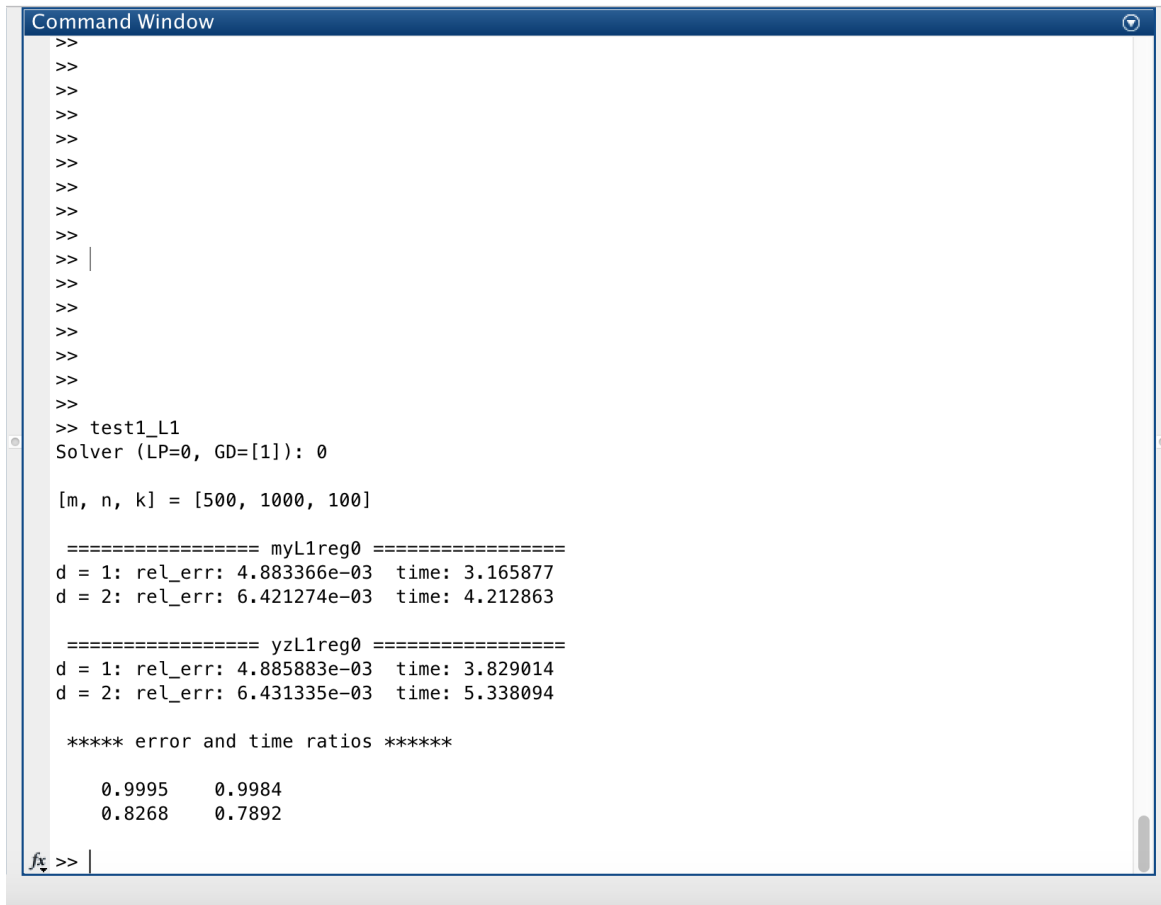
```

nabla = @(x,Dx) ((Dx./sqrt(Dx.^2 + sigma))'*D)'+(mu*(A*x - b)' * A)';
tol_2 = 1e-7;
maxiter = 50000;
beta = 0.5;
C1 = 1e-5;
%% initial setting
x = A' * ((A*A')\b);
Dx = D*x;
f = object(x,Dx);
g = nabla(x,Dx);
gnorm = norm(g); %gnorm0 = gnorm
tol_1 = gnorm * 5e-2;
alpha = 1; %initial step-size

%% Iteration Running
for iter = 1:maxiter
    delta = C1 * alpha * gnorm^2;
    Dg = D*g;
    for arm = 1:10
        % Armijo Condition
        x_try = x - alpha * g;
        Dx_try = Dx - alpha * Dg;
        f_try = object(x_try,Dx_try);
        if f_try <= f - delta, break; end
        alpha = alpha * beta;
    end
    % update function
    x_pre = x; g_pre = g; f_diff = 1 - f_try/f;
    x = x_try; Dx = Dx_try; g = nabla(x,Dx);
    f = f_try; gnorm = norm(g);
    if gnorm <= tol_1 && f_diff <= tol_2, break; end
    % BB step
    s = x - x_pre; y = g - g_pre;
    alpha = (s'*y) / (y' * y);
end
end

```

Matlab screen printout of LP Run



```
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>> test1_L1
>> Solver (LP=0, GD=[1]): 0

[m, n, k] = [500, 1000, 100]

===== myl1reg0 =====
d = 1: rel_err: 4.883366e-03   time: 3.165877
d = 2: rel_err: 6.421274e-03   time: 4.212863

===== yzl1reg0 =====
d = 1: rel_err: 4.885883e-03   time: 3.829014
d = 2: rel_err: 6.431335e-03   time: 5.338094

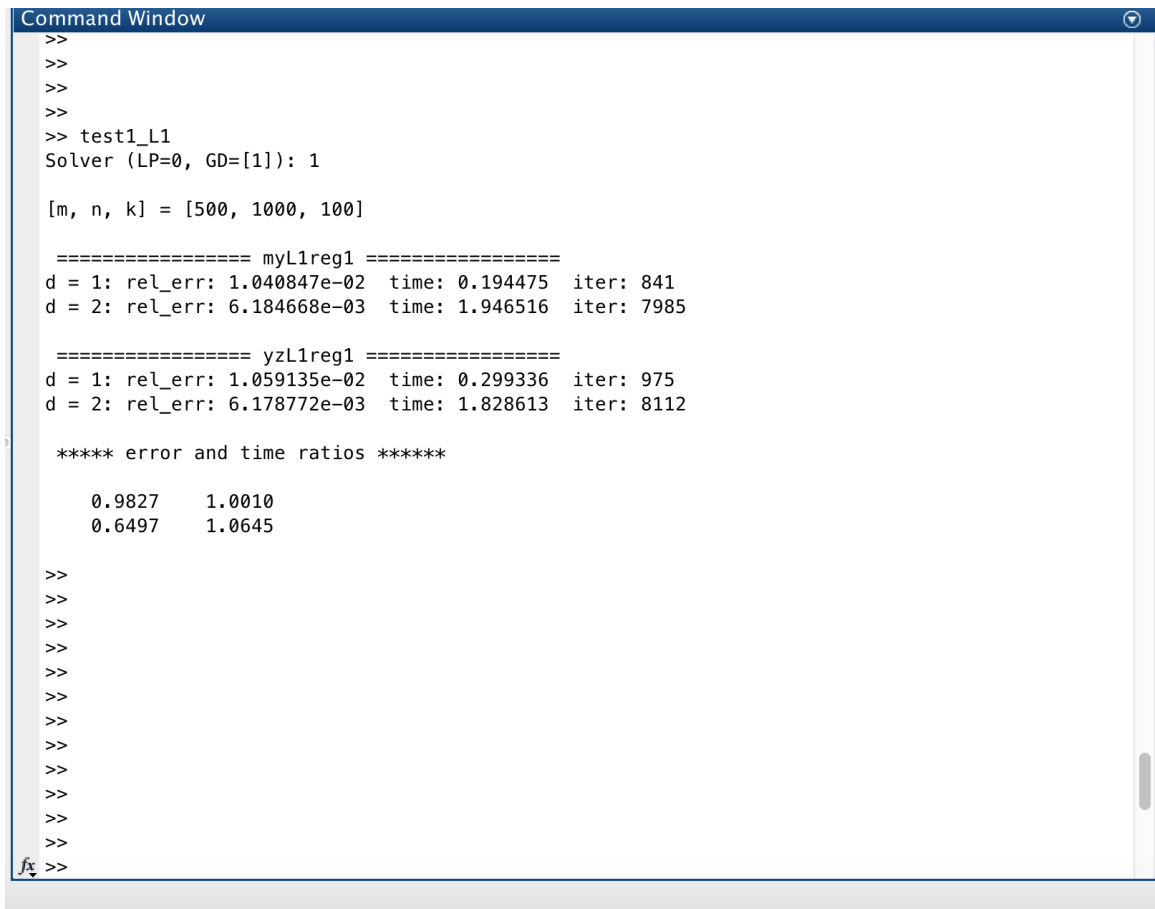
***** error and time ratios *****

    0.9995    0.9984
    0.8268    0.7892

fx >>
```

Figure 1: Printout from Run (LP)

Matlab screen printout of SD Run

A screenshot of the MATLAB Command Window showing the output of an SD Run. The window has a blue title bar with the text "Command Window". The output text is as follows:

```
>>  
>>  
>>  
>>  
>> test1_L1  
Solver (LP=0, GD=[1]): 1  
  
[m, n, k] = [500, 1000, 100]  
  
===== myL1reg1 =====  
d = 1: rel_err: 1.040847e-02  time: 0.194475  iter: 841  
d = 2: rel_err: 6.184668e-03  time: 1.946516  iter: 7985  
  
===== yzL1reg1 =====  
d = 1: rel_err: 1.059135e-02  time: 0.299336  iter: 975  
d = 2: rel_err: 6.178772e-03  time: 1.828613  iter: 8112  
  
***** error and time ratios *****  
  
    0.9827    1.0010  
    0.6497    1.0645  
  
>>  
>>  
>>  
>>  
>>  
>>  
>>  
>>  
>>  
>>  
>>  
fx>>
```

Figure 2: Printout from Run (SD)

Output figures from LP solvers

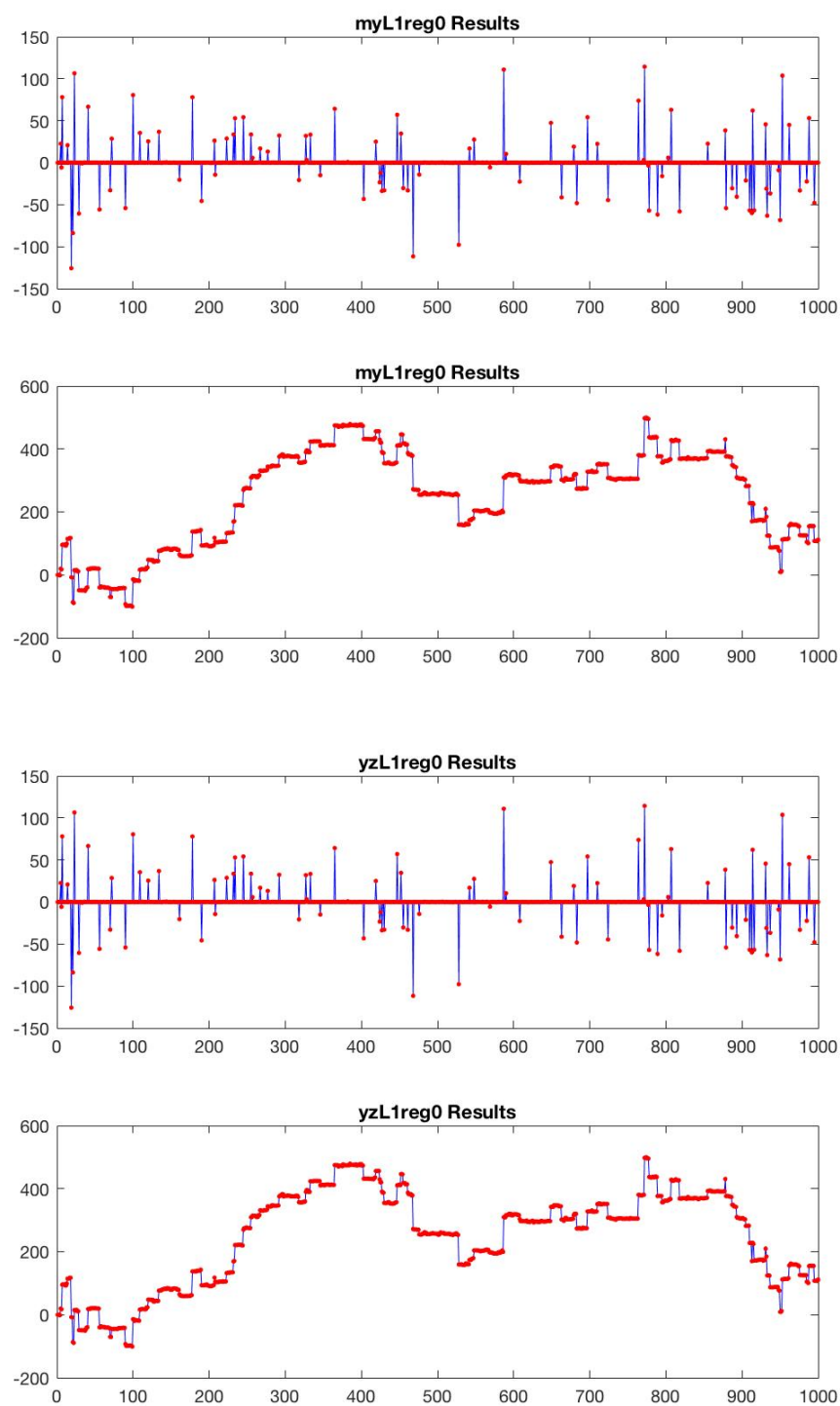


Figure 3: Output figures from LP solvers

Output figures from SD solvers

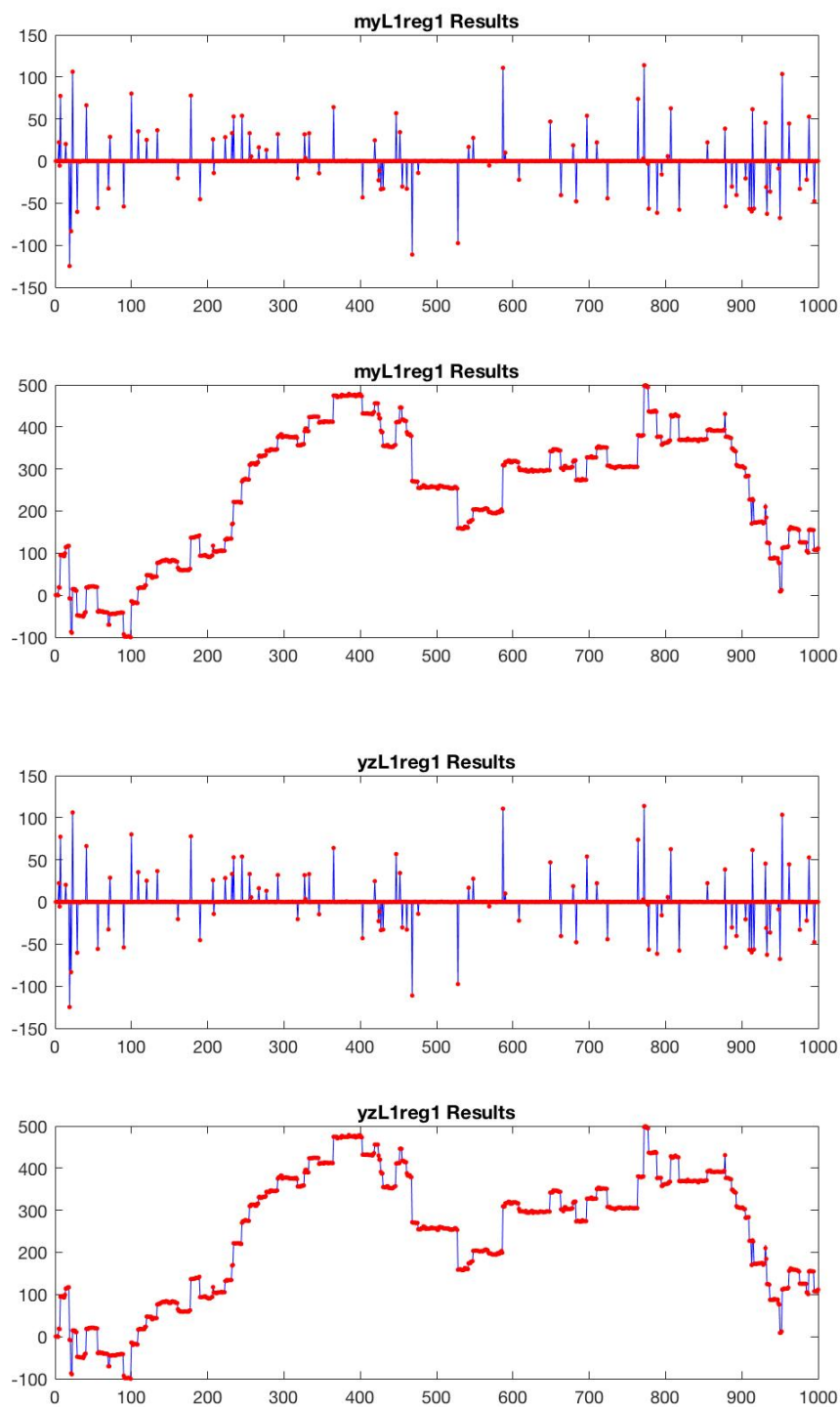


Figure 4: Output figures from SD solvers

A short summary

Introduction This project is to recover a desired signal by L1 Regularization method. Firstly we consider a linear programming approach; then we consider an unconstrained minimization model, which can be solved by steepest gradient descent.

Practice Requirement The LP approach cannot meet the optimal solution in practice in general as the constraint requires the *exact* solution to the under-determined linear system $\mathbf{Ax} = \mathbf{b}$. However, in practice there are always some errors between $\mathbf{Ax} = \mathbf{b}$ due to the noise. Alternatively, the SD model can handle such case well via adding the penalty term $\|\mathbf{Ax} - \mathbf{b}\|_2^2$. This penalty term gives us the approximate solution to the linear system.

Running Time The running time for LP approach is generally slower than that for SD approach. That's because for this large-scale optimization problem, the LP solver has to handle large size matrices (necessarily not be sparse), which is not efficient. In contrast, the SD solver only need to compute operations on vectors (gradient) per iteration, which is efficient.

Choice of Step-size However, the disadvantage of SD approach is that the choice of step-size is a kind of art. There is no principal way that can recommend the step-size for all or even most problems. Careful analysis on the optimization problem is necessary. A bad example is that if we disable BB steps and always initialize $\alpha = 1$ in the back tracking, the running time is sensitive to the scale of optimization problem. Look at the screen printout of this runs:

```

Command Window

>> test1_L1
Solver (LP=0, GD=[1]): 1

[m, n, k] = [500, 1000, 100]

===== myL1reg1 =====
d = 1: rel_err: 1.176728e-02 time: 3.286274 iter: 3663
d = 2: rel_err: 6.300340e-03 time: 21.788607 iter: 27098

===== yzL1reg1 =====
d = 1: rel_err: 1.059135e-02 time: 0.228093 iter: 975
d = 2: rel_err: 6.178772e-03 time: 2.135832 iter: 8112

***** error and time ratios *****

    1.1110    1.0197
   14.4076   10.2015

fg >>

```

Figure 5: screen printout of disabled-BB-step results

Efficient LP Approach The advantage of LP approach is that there is indeed a mature way to solve this problem in polynomial time. The interior point method is one of the best way to solve the large-scale LP problem.