The University of British Columbia

CPSC 406

Computational Optimization

Homework #5

Student name:

ALI SIAHKOOHI

Student ID:

84264167

Instructor:

Professor Ascher

Date Submitted:

November 22nd, 2016

Question #1

Part (a)

The L2 norm condition number of a matrix is the ratio of largest singular value to the smallest singular value of the matrix. We know that singular values of matrix A are the square root of the eigenvalues of $B = A^T A$. Therefore for a symmetric matrix we have:

$$\begin{aligned}
&A^T = A \Rightarrow A^T A = A^2, \\
&\Rightarrow AX = X\Lambda \Rightarrow A^T(AX) = A^T(X\Lambda) \Rightarrow A^2 X = (AX)\Lambda = X\Lambda^2, \\
&\Rightarrow (A^T A)X = X\Lambda^2, \\
&\Rightarrow \sigma_i = |\lambda_i|. \qquad (1-1)
\end{aligned}$$

There for we need to compute the largest and smallest singular value of A. We know the eigenvalues of matrix A are positive. Therefore:

- Largest singular value:

$$\sigma_{max} = |\lambda_{max}| = \lambda_{max} = 4 - 2\left(\cos(\frac{N}{N+1}\pi) + \cos(\frac{N}{N+1}\pi)\right) = 4\left(1 - \cos(\frac{N}{N+1}\pi)\right), \qquad (1-2)$$

- Smallest singular value:

$$\sigma_{min} = \min_{i=1,\dots,N^2} |\lambda_i| = \lambda_{min} = 4 - 2\left(\cos(\frac{1}{N+1}\pi) + \cos(\frac{1}{N+1}\pi)\right) = 4\left(1 - \cos(\frac{1}{N+1}\pi)\right), \qquad (1-3)$$

So we can compute the condition number of matrix A as below:

$$\kappa_2(A) = \frac{\sigma_{max}}{\sigma_{min}} = \frac{1 - \cos(\frac{N}{N+1}\pi)}{1 - \cos(\frac{1}{N+1}\pi)}. \qquad (1-4)$$

Dependency of $\kappa_2(A)$ on N:

- As N increases, the nominator, $1 - \cos(\frac{N}{N+1}\pi)$, gets close to 2,
- As N increases, the dominator, $1 - \cos(\frac{1}{N+1}\pi)$, gets close to 0,

Therefore $\kappa_2(A)$ increases as N increases.

Part (b)

i. CG MATLAB code:

```
function [x, max_alpha, itr] = CG (A, b, tol)

x = zeros(size(A, 2), 1);

r = b - A*x;
p = r;
delta = r'*r;
max_alpha = 0;
j = 0;
nb = norm(b);
while (norm(r) > tol*nb)
    j = j+1;
    s = A*p;
    alpha = delta/(s'*p) ;
    if alpha > max_alpha
        max_alpha = alpha;
    end
    x = x + alpha*p ;
    r = r - alpha*s;
    beta = 1/delta;
    delta = r'*r;
    beta = delta*beta;
    p = r + beta*p;
end

itr = j + 1;

end
```

ii. SD MATLAB code:

```
function [x, max_alpha, itr] = SD (A, b, tol)

x = zeros(size(A, 2), 1);

r = b - A*x;
max_alpha = 0;
j = 0;
nb = norm(b);
while (norm(r) > tol*nb)

    j = j+1;
    s = A*r;
    alpha = r'*r/(s'*r);
    if alpha > max_alpha
        max_alpha = alpha;
    end
    x = x + alpha*r ;
    r = r - alpha*s;

end

itr = j + 1;

end
```

iii. **HLSD MATLAB code:**

```
function [x, max_alpha, itr] = HLSD (A, b, tol)

x = zeros(size(A, 2), 1);

r = b - A*x;
s = A*r;
alpha = r'*r/(s'*r);
max_alpha = alpha;
j = 0;
nb = norm(b);
while (norm(r) > tol*nb)

    j = j+1;
    s = A*r;
    if mod(j, 2) == 0
        alpha = r'*r/(s'*r);
        if alpha > max_alpha
            max_alpha = alpha;
        end
    end
    x = x + alpha*r ;
    r = r - alpha*s;

end

itr = j + 1;

end
```

L	n	Condition Number	#itr CG	#itr SD	#itr HLSD	Max alpha CG	Max alpha SD	Max alpha HLSD
2	9	5.8284	4	31	31	0.75	0.75	0.75
3	49	25.274	10	139	48	1.75	1.75	2.6367
4	225	103.09	23	585	82	3.75	3.75	11.854
5	961	414.35	46	2384	177	7.75	7.75	47.104
6	3969	1659.4	93	9608	373	15.75	15.75	193.69

Relationship between condition number and number of iterations for different methods:

1. CG method:

In this method, as expected, the number of iterations is proportional to square root of the condition number of matrix, $O(\sqrt{\kappa})$. For example we see this relationship in the table above. The ratio of number of iterations to square root of condition number of CG is approximately 2.2 for all values n.

2. SD method:

In this method, the number of iterations is proportional to the condition number itself, $O(\kappa)$. In other words, the ratio of number of iterations to square root of condition number of SD is approximately 5.7 for all values n.

3. HLSD method

In this method it looks like that algorithm converges faster than $O(\kappa)$, In other words it performs better than the original SD algorithm. Also, as it can be seen in the table

Other observation is that, the maximum alpha size of HLSD is much larger than SD, which also emphasizes the faster convergence of HLSD comparing to SD.
Also the fact that the maximum step sizes for CG and SD are the same is reasonable. The difference between CG and SD is in the direction vector which CG chooses them to be orthogonal.

The MATLAB code for this part:

```
clear; clc; close all
l = 2:6;
```

```
matrix.l = l;
N = 2.^l - 1;
tol = 1e-5;
k = 1;
for i = N



    h = 1/(i+1);
    A = delsq(numgrid('S',i+2));
    b = h^2*ones(i^2, 1);

    matrix.size(k) = i^2;
    matrix.cond(k) = (1-cos(sqrt(i^2)*pi/(sqrt(i^2)+1)))/...
        (1-cos(pi/(sqrt(i^2)+1)));

    [~, matrix.max_alpha_cg(k), matrix.itr_cg(k)] =...
        CG (A, b, tol);
    [~, matrix.max_alpha_sd(k), matrix.itr_sd(k)] =...
        SD (A, b, tol);
    [~, matrix.max_alpha_hlsd(k), matrix.itr_hlsd(k)] =...
        HLSD (A, b, tol);
    k = k+1;

end

HW1_Part_1c = table;
HW1_Part_1c.l = matrix.l';
HW1_Part_1c.n = matrix.size';
HW1_Part_1c.Condition_Number = matrix.cond';
HW1_Part_1c.Iteration_count_CG = matrix.itr_cg';
HW1_Part_1c.Iteration_count_SD = matrix.itr_sd';
HW1_Part_1c.Iteration_count_HLSD = matrix.itr_hlsd';
HW1_Part_1c.Max_alpha_CG = matrix.max_alpha_cg';
HW1_Part_1c.Max_alpha_SD = matrix.max_alpha_sd';
HW1_Part_1c.Max_alpha_HLSD = matrix.max_alpha_hlsd';

HW1_Part_1c
```

Exercise #2:

Part (a)

First I am going to write the objective function in matrix notation:

$$u^* = \arg\min_u \varphi_2(u) = \frac{h}{2}\sum_{i=1}^{N}\frac{1}{2}\left(\left(u_i - b_i\right)^2 + \left(u_{i-1} - b_{i-1}\right)^2\right) + \frac{\beta h}{2}\sum_{i=1}^{N}\left(\frac{u_i - u_{i-1}}{h}\right)^2, \qquad (2-1)$$

$$h\sum_{i=1}^{N}\left(\frac{u_i - u_{i-1}}{h}\right)^2, = \sum_{i=1}^{N}\left(\frac{u_i - u_{i-1}}{\sqrt{h}}\right)^2 = \left\| \frac{1}{\sqrt{h}} \begin{pmatrix} -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \end{pmatrix}_{N\times(N+1)} \times \begin{pmatrix} u_0 \\ u_2 \\ \vdots \\ u_N \end{pmatrix} \right\|_2^2 = \|Wu\|_2^2,$$

$$\Rightarrow \frac{\beta h}{2}\sum_{i=1}^{N}\left(\frac{u_i - u_{i-1}}{h}\right)^2 = \frac{\beta}{2}\|Wu\|_2^2. \qquad (2-2)$$

$$\frac{h}{2}\sum_{i=1}^{N}\frac{1}{2}\left(\left(u_i - b_i\right)^2 + \left(u_{i-1} - b_{i-1}\right)^2\right) = \frac{1}{2}\left(\frac{h}{2}\left(u_0 - b_0\right)^2 + \sum_{i=1}^{N-1} h\left(u_i - b_i\right)^2 + \frac{h}{2}\left(u_N - b_N\right)^2\right)$$

$$= \frac{1}{2}\left\| \begin{pmatrix} \sqrt{\frac{h}{2}}\left(u_0 - b_0\right) \\ \sqrt{h}\left(u_1 - b_1\right) \\ \vdots \\ \sqrt{h}\left(u_{N-1} - b_{N-1}\right) \\ \sqrt{\frac{h}{2}}\left(u_N - b_N\right) \end{pmatrix}_{(N+1)\times 1} \right\|_2^2 = \frac{1}{2}\left\| \begin{pmatrix} \sqrt{\frac{h}{2}} & & & \\ & \sqrt{h} & & \\ & & \sqrt{h} & \\ & & & \sqrt{\frac{h}{2}} \end{pmatrix}_{(N+1)\times(N+1)} \begin{pmatrix} u_0 - b_0 \\ u_1 - b_1 \\ \vdots \\ u_{N-1} - b_{N-1} \\ u_N - b_N \end{pmatrix}_{(N+1)\times 1} \right\|_2^2,$$

$$D = \begin{pmatrix} \sqrt{\frac{h}{2}} & & & \\ & \sqrt{h} & & \\ & & \sqrt{h} & \\ & & & \sqrt{\frac{h}{2}} \end{pmatrix}_{(N+1)\times(N+1)},$$

$$\frac{h}{2}\sum_{i=1}^{N}\frac{1}{2}\left(\left(u_i - b_i\right)^2 + \left(u_{i-1} - b_{i-1}\right)^2\right) = \frac{1}{2}\left\|D\left(u - b\right)\right\|_2^2. \qquad (2-3)$$

$$\Rightarrow u^* = \arg\min_u \varphi_2(u) = \frac{1}{2}\left\| D\left(u - b\right)\right\|_2^2 + \frac{\beta}{2}\left\| Wu \right\|_2^2. \qquad (2-4)$$

Now we can write down the gradient and hessian of the objective function.

$$\Rightarrow \nabla_u \ \varphi_2(u) = D^T D\left(u - b\right) + \beta W^T W u = D^2\left(u - b\right) + \beta W^T W u, \qquad (2-5)$$

$$\Rightarrow \nabla_u^2 \ \varphi_2(u) = D^2 + \beta W^T W. \qquad (2-6)$$

As it can be seen from equation (2 – 6), the hessian of $\varphi_2(u)$ is positive definite. Because:

$$\Rightarrow x^T \nabla_u^2 \ \varphi_2(u) x = x^T\left(D^2 + \beta W^T W\right)x = x^T D^T D x + \beta x^T W^T W x = \left\| Dx \right\|_2^2 + \beta \left\| Wx \right\|_2^2 > 0. \qquad (2-7)$$

Therefore $\varphi_2(u)$ is convex.

Part (b)

Here are my results in four combination of γ and noise parameter. The discussion is brought after the results.

I also print the relative error between real b_p and estimated one as shown in equation (4 – 19).

$$\text{error} = \frac{\left\| u - b_p \right\|}{\left\| b_p \right\|}. \qquad (2-7)$$

In each part I am going to plot the data versus estimated model, and print the error above.

1. $\beta = 10^{-2}$, noise $= 10^{-2}$:

```
(Exercise #2 b) - beta = 0.001, noise = 0.01 Relative error = 0.087486
```

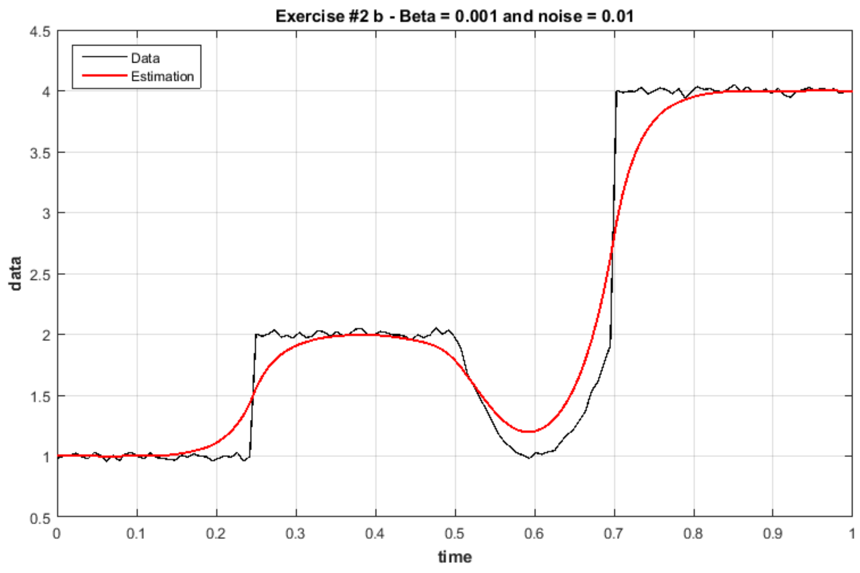

Figure (2 – 1)

2. $\beta = 10^{-3}$, noise $= 10^{-2}$:

```
(Exercise #2 b) - beta = 0.0001, noise = 0.01 Relative error = 0.040355
```

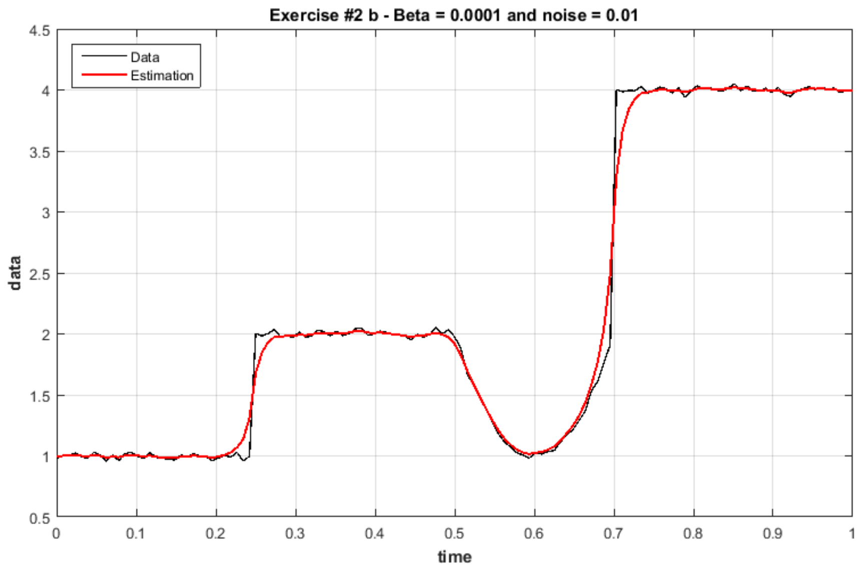

Figure (2 – 2)

3. $\beta = 10^{-2}$, noise $= 10^{-1}$:

```
(Exercise #2 b) - beta = 0.001, noise = 0.1 Relative error = 0.088406
```

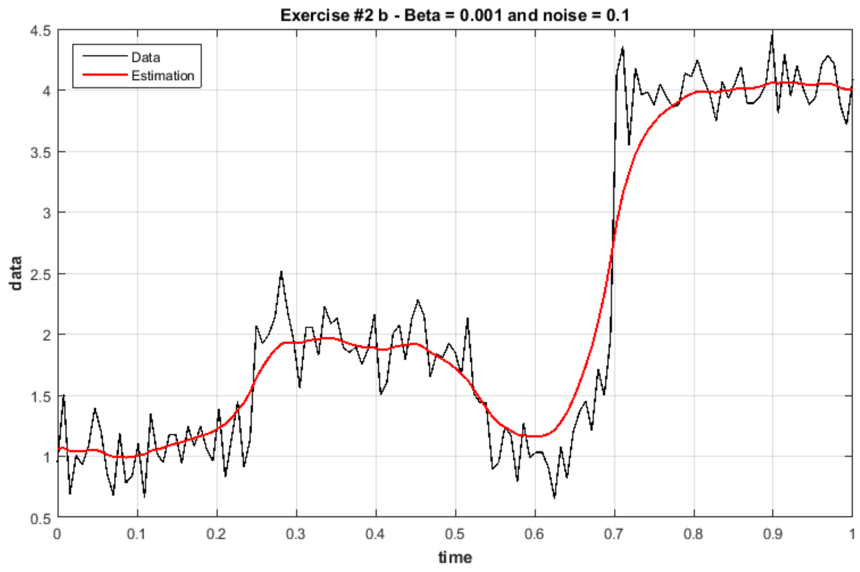

Figure (2 – 3)

4. $\beta = 10^{-3}$, noise $= 10^{-1}$:

```
(Exercise #2 b) - beta = 0.0001, noise = 0.1 Relative error = 0.052735
```

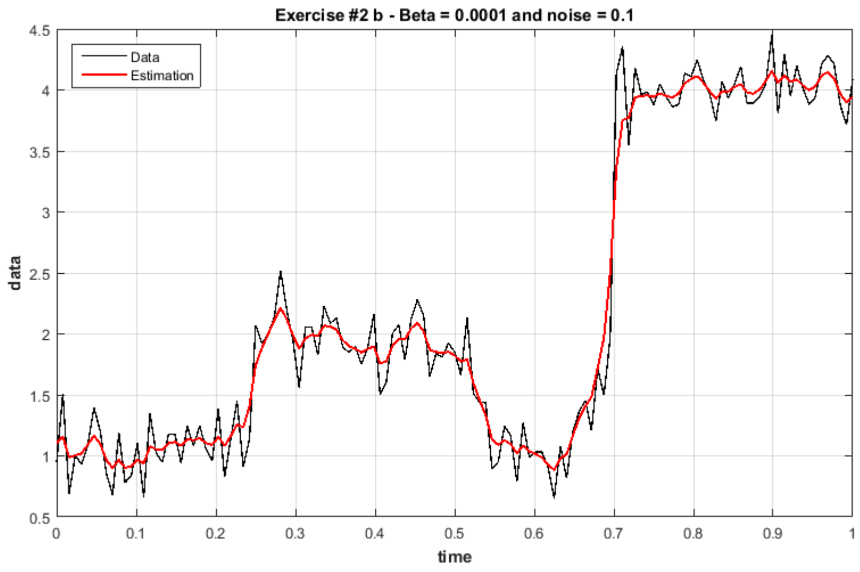

Figure (2 – 4)

Discussion:

Through figures (2 – 1) to (2 – 4) we can see the effect of choosing the regularization parameter, γ, in two noise levels.

1. $\beta = 10^{-2}$, noise $= 10^{-2}$: In figure (2 – 1), considering the noise parameter, which is small, the estimation has not fitted the data properly. Considering the noise level, we should have chosen γ smaller to let the data to get fitted properly. On the other hand if we choose γ very small, we may over fit the data and even fit the noise. Next instance shows a better selection of γ for the same noise level.

2. $\beta = 10^{-3}$, noise $= 10^{-2}$: Comparing figure (2 – 2) to figure (2 – 1), which has the same data, we can see with this regularization parameter, γ, the algorithm has fitted the data properly. But it also has fitted the noise in some sections. The estimation contains some jumps in presence of noise. Although the relative error in this part is smaller than what is reported for figure (2 – 1).

3. $\beta = 10^{-2}$, noise $= 10^{-1}$: In this case, as well as the next instance, he noise level is larger than two previous cases. So we expect the regularization parameter suitable for this problem be larger than what is optimal for two previous instances. The choice made in this case seems to be suitable. It the estimation has managed to somehow follow the trend of data and not over fitting. In other words it doesn't contain jumps related to fitting the noise. Although the relative error is high.

4. $\beta = 10^{-3}$, noise $= 10^{-1}$: as we expect, the regularization parameter is too small, in other words, our estimation is going to fit the data more precisely comparing to previous instance. As it can be seen in figure (2 – 4), the estimation has a lot of jumps due to presence of noise. In addition, these set of parameters have the worst relative error.

Therefore as the noise level increases, we need to choose γ larger in order to give more weight to having a more smooth solution without jumps. Although because of regularization, our estimation is going to have some misfit.

The MATLAB code used:

```
clear; clc; close all

N = 128;
h = 1/N;
t = linspace(0, 1, N + 1);

%%
b_p(1:(find(t>=.25, 1)-1)) = 1;

b_p(find(t>=.25, 1):(find(t>=.5, 1)-1)) = 2;

b_p(find(t>=.5, 1):(find(t>=.7, 1)-1)) = 2 - ...
    100*(t(find(t>=.5, 1):(find(t>=.7, 1)-1))-.5).* ...
    (0.7 - t(find(t>=.5, 1):(find(t>=.7, 1)-1)));

b_p(find(t>=.7, 1):(N + 1)) = 4;

%%

noise = [1e-2, 1e-2, 1e-1, 1e-1];
beta = [1e-3, 1e-4, 1e-3, 1e-4];


%%

D = eye(N+1, N+1);
D(2:N, 2:N) = diag(sqrt(2)*ones(N-1, 1));
D = sqrt(h/2)*D;

W = zeros(N, N+1);
for i = 1:N
    W(i, i) = -1;
    W(i, i+1) = 1;
end
W = W/sqrt(h);

%%

for i = 1:4

if mod(i, 2) == 1
noisev = randn(size(b_p))*mean(abs(b_p))*noise(i);
data = b_p + noisev;
end

figure
plot(t,data,'-k',...
    'LineWidth',1,...
    'MarkerSize',3,...
    'MarkerEdgeColor','b',...
    'MarkerFaceColor',[0.5,0.5,0.5])
title('Exercise #2 b - Beta = 1e-3 and noise = 1e-2')
title(['Exercise #2 b - Beta = ',...
    num2str(beta(i)),' and noise = ', num2str(noise(i))])
```

```
xlabel('time', 'FontSize', 12, 'FontWeight','bold')
ylabel('data', 'FontSize', 12, 'FontWeight','bold')

u = (D^2 + beta(i)*(W'*W))\(D^2*data');

fprintf(['(Exercise #2 b) - beta = ',num2str(beta(i)), ...
    ', noise = ', num2str(noise(i)), ' Relative error = ', ...
    num2str(norm(u - b_p')/norm(b_p)), '\n'])

hold on
plot(t, u, 'r','LineWidth',1.5)
legend('Data', 'Estimation','Location','northwest')
grid on
pause(1.2)
end
```

Question #4
Part (b)

First I am going to bring the notation for solving this problem. Starting with the optimization problem in Question 4 for $\varphi_1(u)$ we can write:

$$u^* = \arg\min_u \varphi_1(u) = \frac{h}{2}\sum_{i=1}^{N}\frac{1}{2}\left(\left(u_i - b_i\right)^2 + \left(u_{i-1} - b_{i-1}\right)^2\right) + \gamma h \|d\|_1, \qquad (4-1)$$

The above equation can be written in matrix-form as below. Starting from total variation norm:

$$h\|d\|_1 = h\sum_{i=1}^{N}|d_i| = h\sum_{i=1}^{N}\left|\frac{u_i - u_{i-1}}{h}\right| = \sum_{i=1}^{N}|u_i - u_{i-1}|, \qquad (4-2)$$

Define z

Define z_i's and matrix W as below:

$$z_i = u_i - u_{i-1} \Rightarrow z = \begin{pmatrix} -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \end{pmatrix}_{N\times(N+1)} u, \qquad (4-3)$$

$$W = \begin{pmatrix} -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \end{pmatrix}_{N\times(N+1)} \quad u \Rightarrow z = Wu, \qquad (4-4)$$

Therefore continuing from equation (4 – 2) we can write:

$$\Rightarrow h\|d\|_1 = \sum_{i=1}^{N}|u_i - u_{i-1}| = \sum_{i=1}^{N}|z_i| = \|z\|_1, \qquad (4-5)$$

Writing the data misfit part in equation (4 – 1) in matrix form:

$$\frac{h}{2}\sum_{i=1}^{N}\frac{1}{2}\left(\left(u_i - b_i\right)^2 + \left(u_{i-1} - b_{i-1}\right)^2\right) = \frac{1}{2}\left(\frac{h}{2}\left(u_0 - b_0\right)^2 + \sum_{i=1}^{N-1} h\left(u_i - b_i\right)^2 + \frac{h}{2}\left(u_N - b_N\right)^2\right)$$

$$= \frac{1}{2}\left\|\begin{pmatrix} \sqrt{\frac{h}{2}}\left(u_0 - b_0\right) \\ \sqrt{h}\left(u_1 - b_1\right) \\ \vdots \\ \sqrt{h}\left(u_{N-1} - b_{N-1}\right) \\ \sqrt{\frac{h}{2}}\left(u_N - b_N\right) \end{pmatrix}_{(N+1)\times 1}\right\|_2^2 = \frac{1}{2}\left\|\begin{pmatrix} \sqrt{\frac{h}{2}} & & & \\ & \sqrt{h} & & \\ & & \sqrt{h} & \\ & & & \sqrt{\frac{h}{2}} \end{pmatrix}_{(N+1)\times(N+1)} \begin{pmatrix} u_0 - b_0 \\ u_1 - b_1 \\ \vdots \\ u_{N-1} - b_{N-1} \\ u_N - b_N \end{pmatrix}_{(N+1)\times 1}\right\|_2^2, \qquad (4-6)$$

Define matrix D and vector g as below:

$$D = \begin{pmatrix} \sqrt{\frac{h}{2}} & & & & \\ & \sqrt{h} & & & \\ & & & & \\ & & & \sqrt{h} & \\ & & & & \sqrt{\frac{h}{2}} \end{pmatrix}_{(N+1)\times(N+1)}, \qquad (4-7)$$

$$g = Db, \qquad (4-8)$$

Continuing from equation (4 – 6) and inserting equation (4 – 7) we get:

$$\frac{h}{2}\sum_{i=1}^{N}\frac{1}{2}\left(\left(u_i - b_i\right)^2 + \left(u_{i-1} - b_{i-1}\right)^2\right) = \frac{1}{2}\left\|D\left(u-b\right)\right\|_2^2 = \frac{1}{2}\left\|Du - Db\right\|_2^2 = \frac{1}{2}\left\|Du - g\right\|_2^2, \qquad (4-9)$$

By inserting equations (4 – 5) and (4 – 9) into equation (4 – 1) we get:

$$\left(u^*, z^*\right) = \arg\min_{u,z} \varphi_1(u) = \frac{1}{2}\left\|Du - g\right\|_2^2 + \gamma\left\|z\right\|_1, \quad \text{s.t. } Wu = z \qquad (4-10)$$

In this question I am going to use ADMM to solve the above problem. The above problem can be seen in the splitting algorithm form. Because the first term is only a function of u and the second term is a function of z. The augmented Lagrangian for equation (4 – 10) is as below:

$$L_A(u, z, y, \nu) = \frac{1}{2}\left\|Du - g\right\|_2^2 + \gamma\left\|z\right\|_1 + y^T\left(Wu - z\right) + \frac{\nu}{2}\left\|Wu - z\right\|_2^2, \qquad (4-11)$$

Based on The split augmented Lagrangian (SAL) method, or ADMM, we can write the kth Iteration of solving the problem (4 – 10) as below:

$$
\begin{aligned}
&u_{k+1} = \arg\min_{u} L_A(u, z_k, y_k, \nu_k), \\
&z_{k+1} = \arg\min_{z} L_A(u_{k+1}, z, y_k, \nu_k), \\
&y_{k+1} = y_k + \nu_k\left(u_{k+1} - z_{k+1}\right), \\
&\nu_{k+1} = \eta \nu_k, \quad \eta \geq 1,
\end{aligned}
\tag{4 – 12}
$$

- Subproblem for finding u_{k+1}:

$$
\begin{aligned}
&\frac{\partial}{\partial u} L_A(u, z_k, y_k, \nu_k) = D^T\left(Du - g\right) + W^T y_k + \nu_k W^T\left(Wu - z_k\right) = 0, \\
&\Rightarrow \left(D^T D + \nu_k W^T W\right) u = D^T b - W^T y_k + \nu_k W^T z_k, \\
&\Rightarrow u_{k+1} = \left(D^T D + \nu_k W^T W\right) \backslash \left(D^T b - W^T y_k + \nu_k W^T z_k\right).
\end{aligned}
\tag{4 – 13}
$$

- Subproblem for finding z_{k+1}:

$$
\begin{aligned}
&z_{k+1} = \arg\min_{z} L_A(u_{k+1}, z, y_k, \nu_k) = \arg\min_{z} s(z), \\
&s(z) = \gamma \|z\|_1 - {y_k}^T z + \frac{\nu_k}{2}\|Wu_{k+1} - z\|_2^2,
\end{aligned}
\tag{4 – 14}
$$

Since z_i’s are decoupled in the equation above, s(z) can be written as below:

$$
\begin{aligned}
&s(z) = \gamma \sum_{i=1}^{N} |z_i| - \sum_{i=1}^{N} {y_i}^{(k)} z_i + \frac{\nu_k}{2} \sum_{i=1}^{N} \left(W_i u_{k+1} - z_i\right)^2 = \sum_{i=1}^{N} \left(\gamma |z_i| - {y_i}^{(k)} z_i + \frac{\nu_k}{2}\left(W_i u_{k+1} - z_i\right)^2 \right), \\
&s_i\left(z_i\right) = \gamma |z_i| - {y_i}^{(k)} z_i + \frac{\nu_k}{2}\left(W_i u_{k+1} - z_i\right)^2, \\
&\Rightarrow s(z) = \sum_{i=1}^{N} s_i\left(z_i\right).
\end{aligned}
\tag{4 – 15}
$$

In the equation above, W_i is the ith row of matrix W. By considering two states which ${z_i}^{(k+1)}$ is nonnegative or not, we can come up with the following soft thresholding algorithm for finding z_{k+1}:

$$
\underset{i=1,2,\ldots,N}{{z_i}^{(k+1)}} = \begin{cases} \max\left\{W_i u_{k+1} + {\nu_k}^{-1}\left({y_i}^{(k)} - \gamma\right), 0\right\}, & W_i u_{k+1} + {\nu_k}^{-1} {y_i}^{(k)} \geq 0 \\ \min\left\{W_i u_{k+1} + {\nu_k}^{-1}\left({y_i}^{(k)} + \gamma\right), 0\right\}, & W_i u_{k+1} + {\nu_k}^{-1} {y_i}^{(k)} < 0 \end{cases}.
\tag{4 – 16}
$$

Therefore the kth iteration for solving optimization problem in (4 – 10) using ADMM is as below:

$$\begin{aligned}
&u_{k+1} = \left(D^T D + \nu_k W^T W\right) \backslash \left(D^T b - W^T y_k + \nu_k W^T z_k\right), \\
&\underset{i=1,2,\dots,N}{z_i^{(k+1)}} = \begin{cases} \max\left\{W_i u_{k+1} + \nu_k^{-1}\left(y_i^{(k)} - \gamma\right), 0\right\}, & W_i u_{k+1} + \nu_k^{-1} y_i^{(k)} \geq 0 \\ \min\left\{W_i u_{k+1} + \nu_k^{-1}\left(y_i^{(k)} + \gamma\right), 0\right\}, & W_i u_{k+1} + \nu_k^{-1} y_i^{(k)} < 0 \end{cases}, \qquad (4-17) \\
&y_{k+1} = y_k + \nu_k \left(u_{k+1} - z_{k+1}\right), \\
&\nu_{k+1} = \eta \nu_k, \quad \eta \geq 1,
\end{aligned}$$

Part (c)

Here are my results in four combination of γ and noise parameter. The discussion is brought after the results. For the following instances I have used the stopping criteria in equation below through ADMM iterations. In addition, I have chosen $\eta = 1.001$.

$$\|Wu - z\|_2 \leq 10^{-8} \|Wu\|_2 . \qquad (4-18)$$

I also print the relative error between real b_p and estimated one as shown in equation (4 – 19).

$$\text{error} = \frac{\|u - b_p\|}{\|b_p\|}. \qquad (4-19)$$

In each part I am going to plot the data versus estimated model, and print the error above.

5. $\gamma = 10^{-2}$, noise $= 10^{-2}$:

```
(Exercise #4 c) - gamma = 0.01, noise = 0.01 Relative error = 0.032089
```

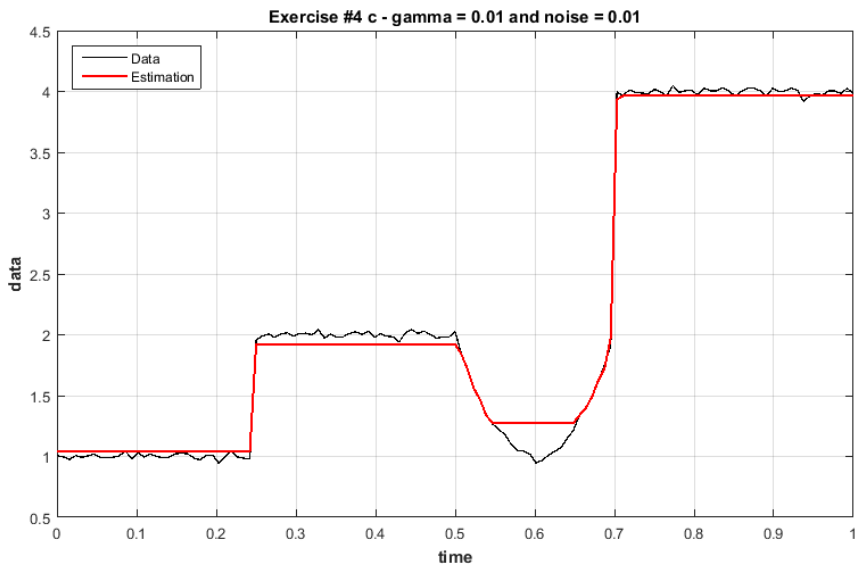

Figure (4 – 1)

6. $\gamma = 10^{-3}$, noise $= 10^{-2}$:

```
(Exercise #4 c) - gamma = 0.001, noise = 0.01 Relative error = 0.0053941
```

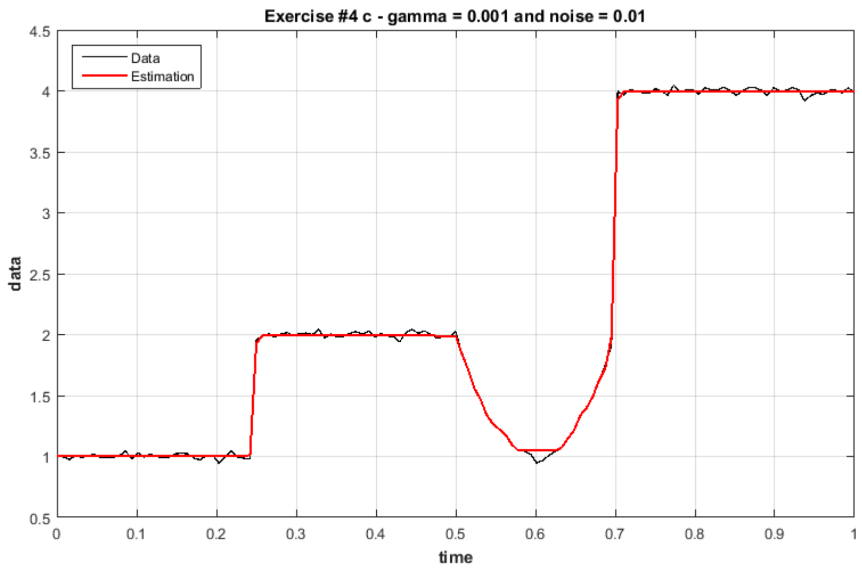

Figure (4 – 2)

7. $\gamma = 10^{-2}$, noise $= 10^{-1}$:

```
(Exercise #4 c) - gamma = 0.01, noise = 0.1 Relative error = 0.043518
```

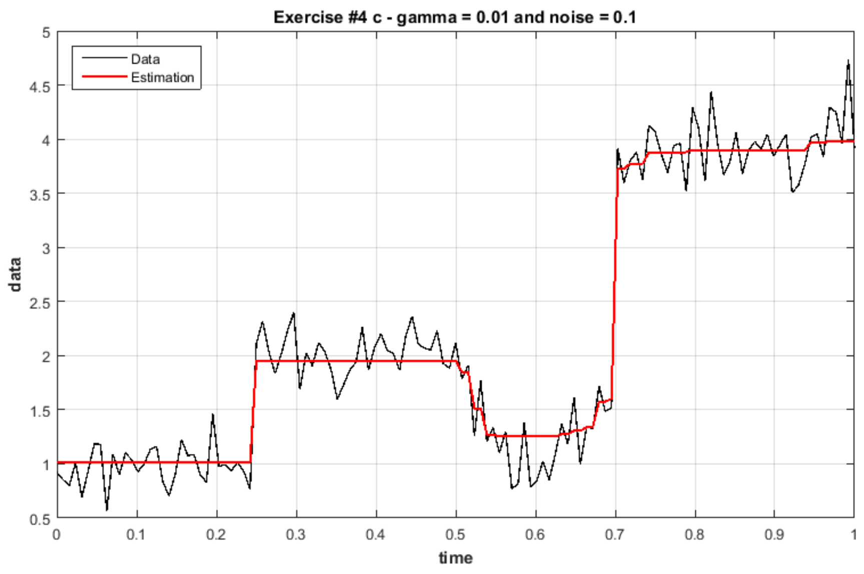

Figure (4 – 3)

8. $\gamma = 10^{-3}$, noise $= 10^{-1}$:

```
(Exercise #4 c) - gamma = 0.001, noise = 0.1 Relative error = 0.049586
```

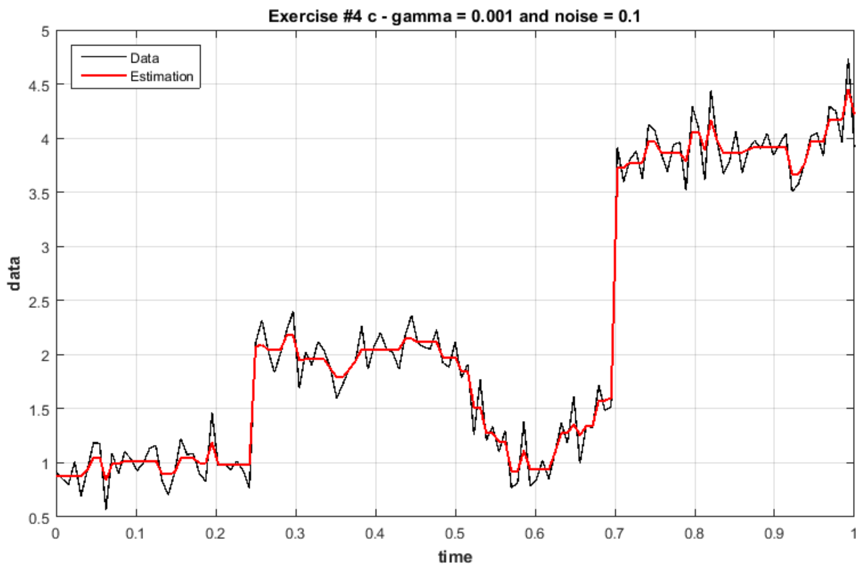

Figure (4 – 4)

Discussion:

Through figures (4 – 1) to (4 – 4) we can see the effect of choosing the regularization parameter, γ, in two noise levels.

5. $\gamma = 10^{-2}$, $\text{noise} = 10^{-2}$: In figure (4 – 1), considering the noise parameter, which is small, the estimation has not fitted the data properly. For example in t = 0.6, the estimation has missed an important structure. Considering the noise level, we should have chosen γ smaller to let the data to get fitted properly. On the other hand if we choose γ very small, we may over fit the data and even fit the noise. Next instance shows a better selection of γ for the same noise level.

6. $\gamma = 10^{-3}$, $\text{noise} = 10^{-2}$: Comparing figure (4 – 2) to figure (4 – 1), which has the same data, we can see with this regularization parameter, γ, the algorithm has fitted the data properly, while not over fitting it. Also the relative error in this part is smaller than what is reported for figure (4 – 1). By enforcing the derivative of u to be sparse, we have an estimation which is piecewise constant, except for $0.5 \le t \le 0.7$ which is also the case for the original data without noise. In addition, it can also be seen from figure (2 – 1) that we haven't fitted the noise, again because of enforcing the derivative to be spars. This is in contrast to what we would get if we had used L2 norm regularization.

7. $\gamma = 10^{-2}$, $\text{noise} = 10^{-1}$: In this case, as well as the next instance, he noise level is larger than two previous cases. So we expect the regularization parameter suitable for this problem be larger than what is optimal for two previous instances. The choice made in this case seems to be suitable. It the estimation has managed to follow the trend of data and not over fitting.

8. $\gamma = 10^{-3}$, $\text{noise} = 10^{-1}$: as we expect, the regularization parameter is too small, in other words, our estimation is going to fit the data more precisely comparing to previous instance. As it can be seen in figure (4 – 4), the first derivative of estimation is not sparse since our choice for γ is small, therefore the estimation has a lot of jumps due to presence of noise.

Therefore as the noise level increases, we need to choose γ larger in order to give more weight to having a sparse solution rather to over fitting the data.

The MATLAB code used:

```
clear; clc; close all

N = 128;
h = 1/N;
t = linspace(0, 1, N + 1);

%%
b_p(1:(find(t>=.25, 1)-1)) = 1;

b_p(find(t>=.25, 1):(find(t>=.5, 1)-1)) = 2;

b_p(find(t>=.5, 1):(find(t>=.7, 1)-1)) = 2 - ...
    100*(t(find(t>=.5, 1):(find(t>=.7, 1)-1))-.5).* ...
    (0.7 - t(find(t>=.5, 1):(find(t>=.7, 1)-1)));

b_p(find(t>=.7, 1):(N + 1)) = 4;

%%
eta = 1.001;
tol = 1e-8;
noise = [1e-2, 1e-2, 1e-1, 1e-1];
gamma = [1e-2, 1e-3, 1e-2, 1e-3];


%%

D = eye(N+1, N+1);
D(2:N, 2:N) = diag(sqrt(2)*ones(N-1, 1));
D = sqrt(h/2)*D;

W = zeros(N, N+1);
for i = 1:N
    W(i, i) = -1;
    W(i, i+1) = 1;
end

%%

for i = 1:4

if mod(i, 2) == 1
noisev = randn(size(b_p))*mean(abs(b_p))*noise(i);
data = b_p + noisev;
end

figure
plot(t,data,'k','LineWidth',1)
title('Exercise #4 c - Beta = 1e-3 and noise = 1e-2')
title(['Exercise #4 c - gamma = ',...
    num2str(gamma(i)),' and noise = ', num2str(noise(i))])
xlabel('time', 'FontSize', 12, 'FontWeight','bold')
ylabel('data', 'FontSize', 12, 'FontWeight','bold')
```

```
k = 0;
nu = 1;
u = zeros(N+1, 1);
z = ones(N, 1);
y = zeros(N, 1);
while( norm(W*u - z) > tol*norm(W*u))

    B = D'*D + nu*(W'*W);
    c = D'*(D*data') - W'*y + nu*W'*z;

    u = B\c;

    zz = W*u + y/nu;

    ind_p = find(zz>=0);
    ind_n = find(zz<0);

    z(ind_p) = max(zz(ind_p) - gamma(i)/nu, 0);
    z(ind_n) = min(zz(ind_n) + gamma(i)/nu, 0);

    y = y + nu*(W*u - z);
    nu = nu * eta;

end
fprintf(['(Exercise #4 c) - gamma = ',num2str(gamma(i)), ...
    ', noise = ', num2str(noise(i)), ' Relative error = ', ...
    num2str(norm(u - b_p')/norm(b_p)), '\n'])
hold on
plot(t, u, 'r','LineWidth',1.5)
legend('Data', 'Estimation','Location','northwest')
grid on
% pause(1.2)
end
```