HW4

Computer Exercise

Part a

In Solution PDF

Part b

In Solution PDF

Part c

i. Varying n, sigma constant

Code

```
% Part a and b done on paper
% Part c_i (varying n, sigma constant)

count = 0;
kernel = 3;                    % based on w_true = [1,3,2]'
mse_vector = zeros(5,1);
w_average_matrix = zeros(kernel,5);
w_true_matrix = zeros(kernel,5);


for n = [10,20,30,40,50]

    count = count + 1;
    A = zeros(2*n+1,kernel);
    sigma = 1;
    x_interim = -n:n;
    x = transpose(x_interim*0.1);  % turning to vector

    % A generation

    for i = 1:kernel
        A(:,i) = x.^i;
    end

    % t generation
    t = A(:,1) + 3*A(:,2) + 2*A(:,3);

    w_true = pinv(A)*t;          % calculating the true w directly from t
    w_true_matrix(:,count) = w_true;

    w_average_interim = zeros(kernel,1);
```

```
    mse_interim = 0;

    for j = 1:100

        % v generation
        v = sigma*randn(2*n+1,1);

        % y generation
        y = t + v;

        w_estimated = pinv(A)*y;        % calculating the estimate of w

        w_average_interim = w_average_interim + w_estimated;  % running sum of w_average

        mse_interim = mse_interim + (norm(w_true - w_estimated))^2; % running sum of mse


    end

    mse = mse_interim/100;
    w_average = w_average_interim/100;

    mse_vector(count,1) = mse;
    w_average_matrix(:,count) = w_average;



end
```

```

n_range = [10;20;30;40;50];
disp("The matrix of true w, where each column corresponds to a value of n and the column vector the vecto
disp(w_true_matrix)
disp("The matrix of estimated w (averaged over 100 iterations, for each n), where each column corresponds
disp(w_average_matrix)
disp("The averaged MSE over 100 iterations for each n, where rows represent MSE value for each n, is")
disp(mse_vector)

plot(n_range,mse_vector,'-o')
xlabel('values of n, sigma = 1 (constant)')
ylabel('MSE')
```

Results

The matrix of true w, where each column corresponds to a value of n and the column vector the vector of w, is,

1.0000	1.0000	1.0000	1.0000	1.0000
3.0000	3.0000	3.0000	3.0000	3.0000
2.0000	2.0000	2.0000	2.0000	2.0000

As expected, it is [1,3,2]

The matrix of estimated w (averaged over 100 iterations, for each n), where each column corresponds to a value of n and the column vector the vector of averaged w estimate, is,

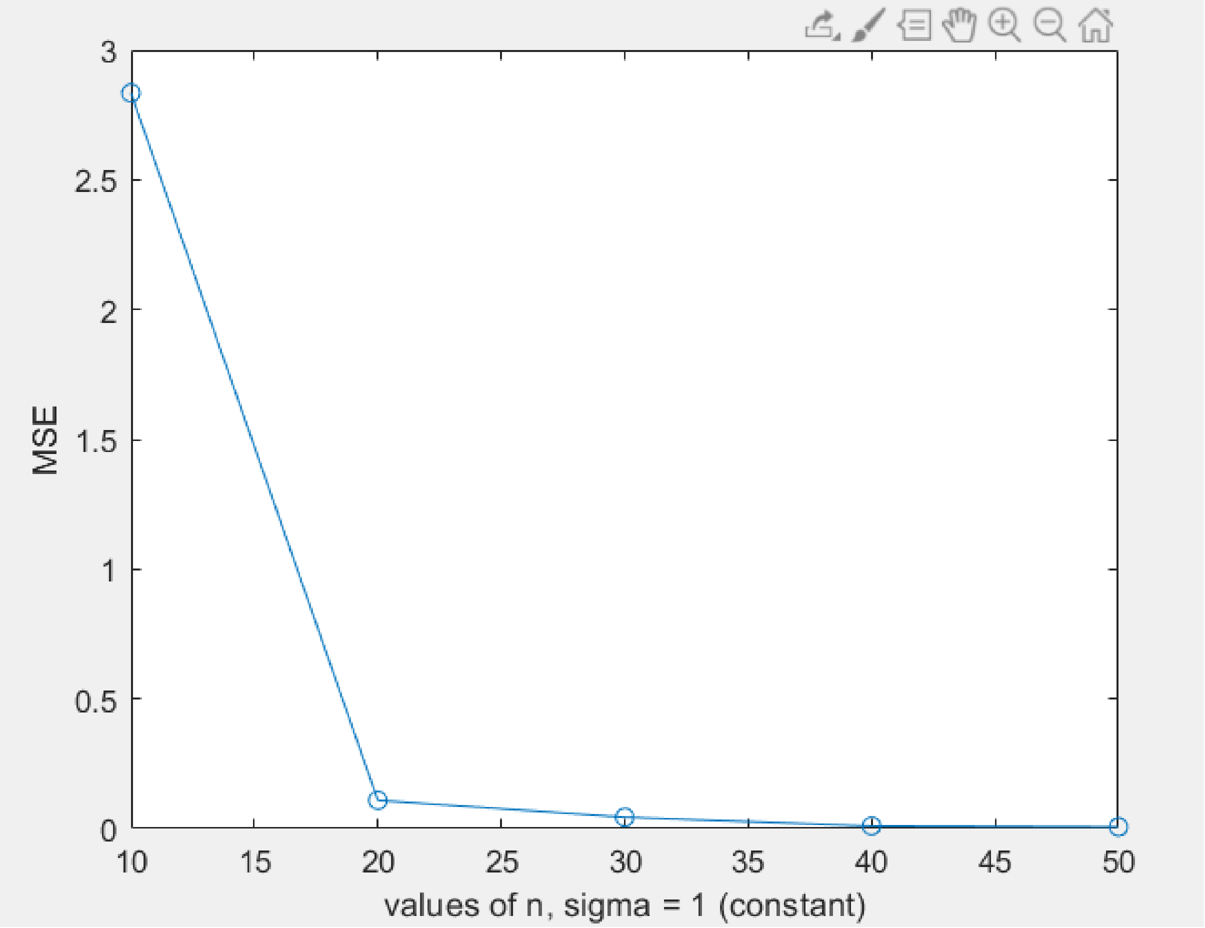
0.9333	1.0302	1.0023	1.0099	1.0042
3.0322	2.9993	2.9981	2.9999	3.0000
2.0261	1.9950	2.0001	1.9990	1.9996

As n increases, the average (over 100 trials) of estimate of **w** becomes more accurate.

The averaged MSE over 100 iterations for each n, where rows represent MSE value for each n, is,

2.3952
0.1106
0.0316
0.0174
0.0076

The MSE values also reduce as the n value increases.



The above shows the MSE versus n plot with sigma constant

ii. Varying sigma, n constant (n = 20)

Code

```
% Part c_ii (varying sigma, n constant)


kernel = 3;                    % based on w_true = [1,3,2]'
mse_vector = zeros(4,1);
w_average_matrix = zeros(kernel,4);


n = 20;
A = zeros(2*n+1,kernel);
x_interim = -n:n;
x = transpose(x_interim*0.1);  % turning to vector

% A generation

for i = 1:kernel
    A(:,i) = x.^i;
end

% t generation
t = A(:,1) + 3*A(:,2) + 2*A(:,3);

mp_i = pinv(A); % as A stays same, irrespective of sigma, for a given n, we

w_true = mp_i*t;         % calculating the true w directly from t

count = 0;

 for sigma = [0.1,1,3,5]

    count = count + 1;

```

```
    w_average_interim = zeros(kernel,1);

    mse_interim = 0;

    for j = 1:100

        % v generation
        v = sigma*randn(2*n+1,1);

        % y generation
        y = t + v;

        w_estimated = mp_i*y;        % calculating the estimate of w

        w_average_interim = w_average_interim + w_estimated;  % running sum

        mse_interim = mse_interim + (norm(w_true - w_estimated))^2; % runni


    end

    mse = mse_interim/100;
    w_average = w_average_interim/100;

    mse_vector(count,1) = mse;
    w_average_matrix(:,count) = w_average;



end
```

```
sigma_range = [0.1;1;3;5];
disp("The matrix of true w, obtained same for all sigma (obviously) is")
disp(w_true)
disp("The matrix of estimated w (averaged over 100 iterations, for each sig
disp(w_average_matrix)
disp("The averaged MSE over 100 iterations for each sigma, where rows repre
disp(mse_vector)

plot(sigma_range,mse_vector,'-o')
xlabel('values of sigma, n = 20 (constant)')
ylabel('MSE')

```

The code is pretty similar to c, Part i.

Results

The matrix of true w, obtained same for all sigma (obviously) is,

```
1.0000
3.0000
2.0000
```

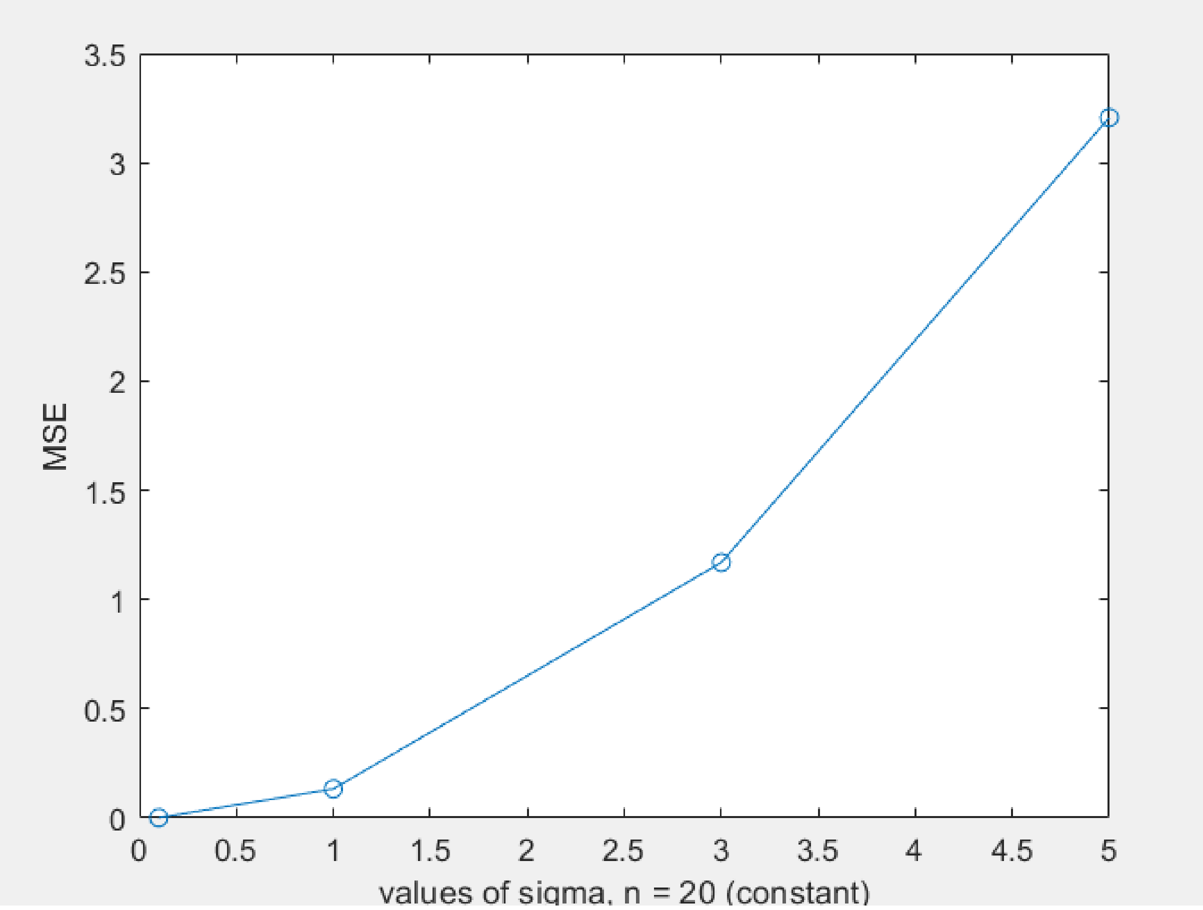
The matrix of estimated w (averaged over 100 iterations, for each sigma), where each column corresponds to a value of sigma and the column vector the vector of averaged w estimate, is,

0.9986	0.9978	1.1565	1.2948
3.0007	2.9989	2.9924	3.0163
2.0009	1.9951	1.9407	1.9570

The averaged MSE over 100 iterations for each sigma, where rows represent MSE value for each sigma, is,

```
0.0012
0.1332
1.1697
3.2069
```

The curve is,



As expected, as the variance of the added noise increases, the MSE increases and the estimate of **w** becomes more and more inaccurate.

Variance of z*sigma = $(\text{sigma})^2$ Variance of z = $(\text{sigma})^2$

Since z is standard normal, with zero mean and unit variance.

Thus, as the value of added distortion to **t** increases, error in **w**, increases.

y = t + v = t + z*sigma

Thus, by increasing sigma, we increase both the variance (how much the noise varies) as well as the overall value of the noise (the probability of larger values appearing in noise term increases as the value of sigma increases).

Thus, with increase in sigma, due to these reasons, MSE increases.

iii. For Model with x and x^3 (underfit)

PART 1 – Varying n with sigma constant

Code

```
% c_iii_Part1

% NOTE :
% We assume that the t observed is observed still from the true model with
% the added noise. Our model has changed, true model has not.

% Thus, A will be formed with respect to our new model, w corresponds to w
% of our new model, y = t + v is generated with v generated as before and t
% generated according to our true model, not the mismatched model. We
% observe x and t and we are simulating using y, the observed t. Thus y
% doesn't change despite the change in our model. Thus the rule to generate
% t still stays the same.

% w true will be obtained without the noise added and w estimate with the
% noise added.

% t according to our new mismatched model will be calculated after we
% obtain w, using the new model. t true will then be the observed values of
% t, or the values to which model is fit, i.e. true t value is the one
% generated by the true model alongside the added error (or t which will be
% used to obtain w)

% NOTE : True t and obtained t won't be stored for each n (averaged
% obviously over 100 trials) and stored, since it hasn't been asked, and we
% don't want to increase unneccessary computation. Obviously doing so
% wouldn't at all be difficult and would need around 1-2 more lines of
% code. So it has been avoided here.

% Part c_iii_Model1_Part1 (varying n, sigma constant) (Model is the one
% which is underfit to the true model, i.e. the non-noisy version of
% the true model)

```

```
count = 0;
kernel = 3;                 % for t generation, the true one
mse_vector = zeros(5,1);
recon_mse_vector = zeros(5,1);
kernel_mis = 2;
w_average_matrix = zeros(kernel_mis,5);
w_true_matrix = zeros(kernel_mis,5);


for n = [10,20,30,40,50]

   count = count + 1;
   A = zeros(2*n+1,kernel);
   sigma = 1;
   x_interim = -n:n;
   x = transpose(x_interim*0.1);  % turning to vector

   % A generation, true A, A that gives true t

   for i = 1:kernel
       A(:,i) = x.^i;
   end

   % t generation
   t = A(:,1) + 3*A(:,2) + 2*A(:,3);         % true t

   % A, for our model, generation
```

```matlab

A_mis = zeros(2*n+1,kernel_mis);

for i = 1:kernel_mis
    A_mis(:,i) = x.^(2*i-1);              % our kernel model
end


w_true = pinv(A_mis)*t;
w_true_matrix(:,count) = w_true;

w_average_interim = zeros(kernel_mis,1);

mse_interim = 0;

recon_mse_interim = 0;

for j = 1:100

    % v generation
    v = sigma*randn(2*n+1,1);

    % y generation
    y = t + v;           % observed y

    w_estimated = pinv(A_mis)*y;       % calculating the estimate of w

    y_estimated = A_mis*w_estimated;

    w_average_interim = w_average_interim + w_estimated;  % running sum of w_average

    mse_interim = mse_interim + (norm(w_true - w_estimated))^2; % running sum of mse
```

```
            recon_mse_interim = recon_mse_interim + (norm(y - y_estimated))^2;

        end

        mse = mse_interim/100;
        recon_mse = recon_mse_interim/100;
        w_average = w_average_interim/100;

        mse_vector(count,1) = mse;
        recon_mse_vector(count,1) = recon_mse;
        w_average_matrix(:,count) = w_average;



    end

    n_range = [10;20;30;40;50];
    disp("The matrix of true w for our model, where each column corresponds to
    disp(w_true_matrix)
    disp("The matrix of estimated w for our model(averaged over 100 iterations,
    disp(w_average_matrix)
    disp("The averaged MSE of w over 100 iterations for each n, where rows rep
    disp(mse_vector)
    disp("The averaged MSE of reconfiguration over 100 iterations for each n, w
    disp(recon_mse_vector)

    figure(1);
    plot(n_range,mse_vector,'o-')
    xlabel('values of n, sigma = 1 (constant)')
    ylabel('MSE for w')
```

```
    figure(2);
    plot(n_range,recon_mse_vector,'o-')
    xlabel('values of n, sigma = 1 (constant)')
    ylabel('MSE for reconfiguration')
```

Results

>> c_iii_Model1_Part1

The matrix of true w for our model, where each column corresponds to a value of n and the column vector the vector of w, is,

1.0000	1.0000	1.0000	1.0000	1.0000
2.0000	2.0000	2.0000	2.0000	2.0000

The matrix of estimated w for our model (averaged over 100 iterations, for each n), where each column corresponds to a value of n and the column vector the vector of averaged w estimate, is,

0.9973	0.9585	1.0251	1.0157	0.9945
1.9959	2.0160	1.9950	1.9988	2.0002

The averaged MSE of w over 100 iterations for each n, where rows represent MSE of w value for each n, is,

2.3247

0.1036

0.0395

0.0149

0.0052

The averaged MSE of reconfiguration over 100 iterations for each n, where rows represent MSE of reconfiguration value for each n, is,

1.0e+05 *

0.0006

0.0134

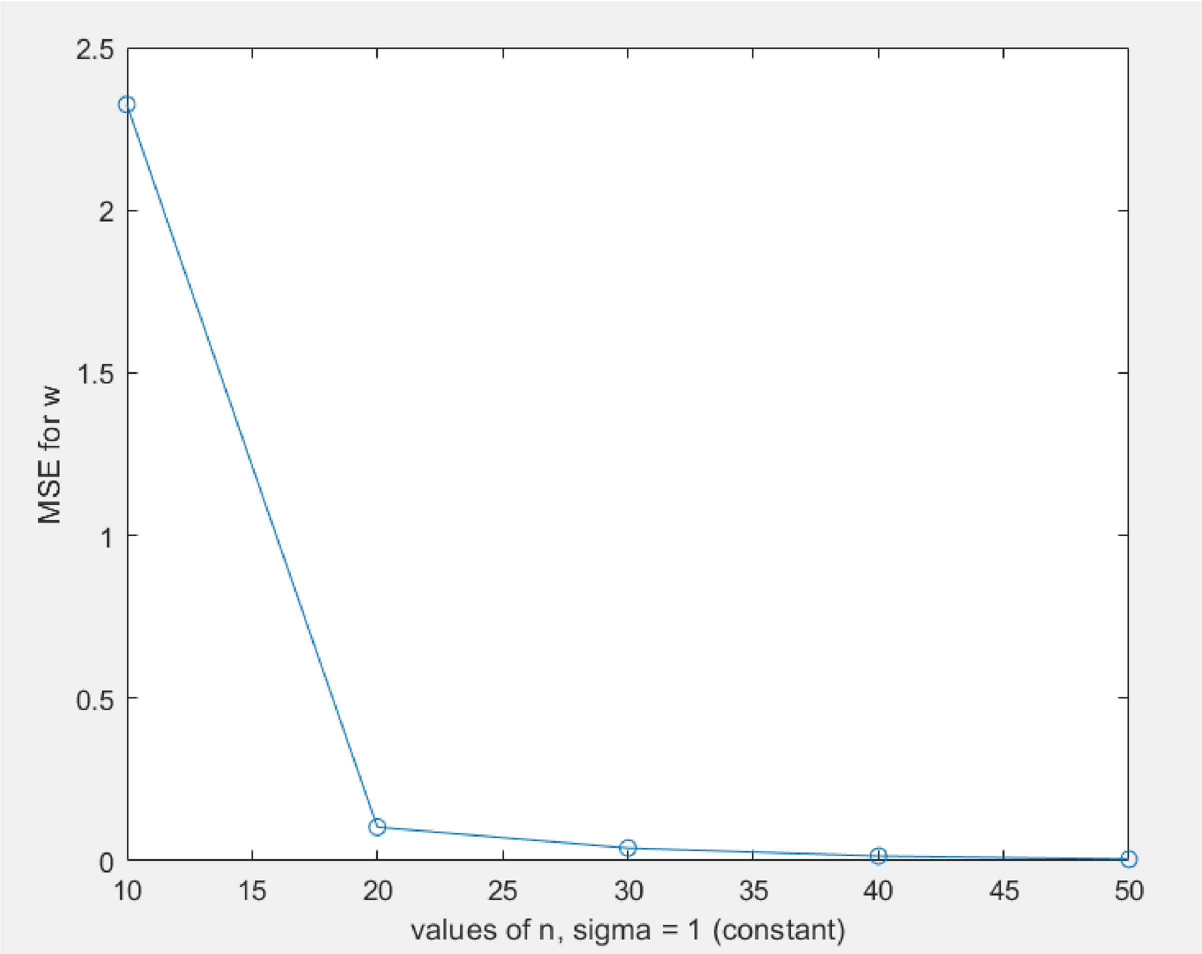
0.0955

0.3931

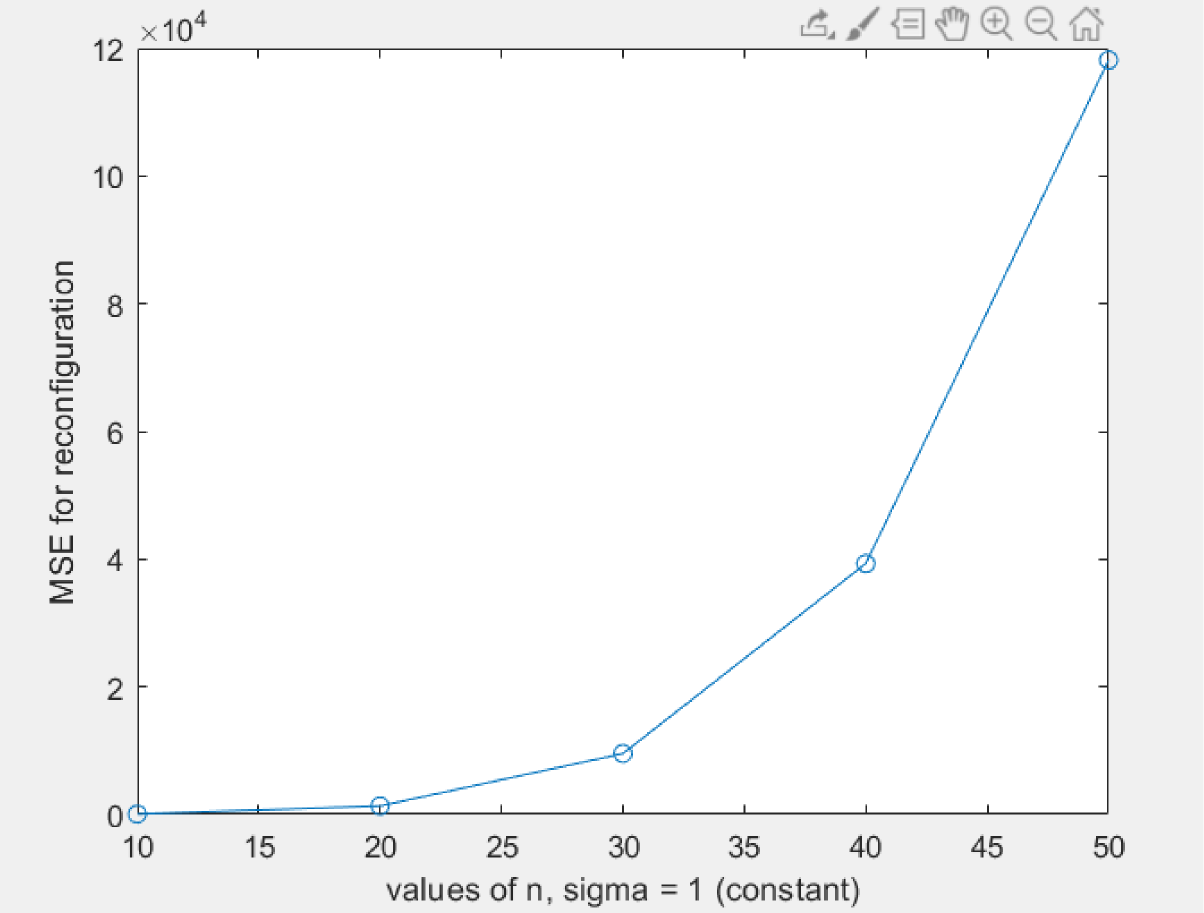
1.1815

Plots

MSE for w



MSE for Reconfiguration



The MSE for reconfiguration is extremely large, since, the coefficients of x and x^3 are still close to 1 and 2 as before, but now, we have the x^2 factor missing due to which our estimated y deviates largely from our actual y in general.

PART 2 – Varying sigma with n constant, same model

Code

```matlab
% Part c_iii_Model1_Part2 (varying sigma, n constant) (Model is the one
% which is underfit to the true model, i.e. the non-noisy version of
% the true model, with x and x^3 only)

% Everything same, but now only sigma varies


 count = 0;
 kernel = 3;                    % for t generation, the true one
 mse_vector = zeros(4,1);
 recon_mse_vector = zeros(4,1);
 kernel_mis = 2;
 w_average_matrix = zeros(kernel_mis,4);


n = 20;
A = zeros(2*n+1,kernel);
x_interim = -n:n;
x = transpose(x_interim*0.1);  % turning to vector

% A generation, true A, A that gives true t

for i = 1:kernel
    A(:,i) = x.^i;
end

% t generation
t = A(:,1) + 3*A(:,2) + 2*A(:,3);         % true t

% A, for our model, generation

A_mis = zeros(2*n+1,kernel_mis);
```

```
for i = 1:kernel_mis
    A_mis(:,i) = x.^(2*i-1);                % our kernel model
end

pm_i = pinv(A_mis);
w_true = pm_i*t;

 for sigma = [0.1,1,3,5]

    count = count + 1;

    w_average_interim = zeros(kernel_mis,1);

    mse_interim = 0;

    recon_mse_interim = 0;

    for j = 1:100

        % v generation
        v = sigma*randn(2*n+1,1);

        % y generation
        y = t + v;           % observed y

        w_estimated = pm_i*y;       % calculating the estimate of w

        y_estimated = A_mis*w_estimated;

        w_average_interim = w_average_interim + w_estimated;  % running sum of w_average
```

```
            mse_interim = mse_interim + (norm(w_true - w_estimated))^2; % runni

            recon_mse_interim = recon_mse_interim + (norm(y - y_estimated))^2;

        end

        mse = mse_interim/100;
        recon_mse = recon_mse_interim/100;
        w_average = w_average_interim/100;

        mse_vector(count,1) = mse;
        recon_mse_vector(count,1) = recon_mse;
        w_average_matrix(:,count) = w_average;



    end

    sigma_range = [0.1;1;3;5];
    disp("The true w vector for our model, which is obviously constant for all
    disp(w_true)
    disp("The matrix of estimated w for our model(averaged over 100 iterations,
    disp(w_average_matrix)
    disp("The averaged MSE of w over 100 iterations for each sigma, where rows
    disp(mse_vector)
    disp("The averaged MSE of reconfiguration over 100 iterations for each sigm
    disp(recon_mse_vector)

    figure(1);
    plot(sigma_range,mse_vector,'o-')
    xlabel('values of sigma, n = 20 (constant)')
    ylabel('MSE for w')

    figure(2);
    plot(sigma_range,recon_mse_vector,'o-')
    xlabel('values of sigma, n = 20 (constant)')
    ylabel('MSE for reconfiguration')
```

Results

>> c_iii_Model1_Part2

The true w vector for our model, which is obviously constant for all sigma, since n is constant is

    1.0000

    2.0000

The matrix of estimated w for our model(averaged over 100 iterations, for each sigma), where each column corresponds to a value of sigma and the column vector the vector of averaged w estimate, is

0.9991	0.9937	0.8778	0.7348
2.0004	2.0037	2.0665	2.1051

The averaged MSE of w over 100 iterations for each sigma, where rows represent MSE of w value for each sigma, is

0.0010

0.1453

1.0021

2.7495

The averaged MSE of reconfiguration over 100 iterations for each sigma, where rows represent MSE of reconfiguration value for each sigma, is

1.0e+03 *

1.3007

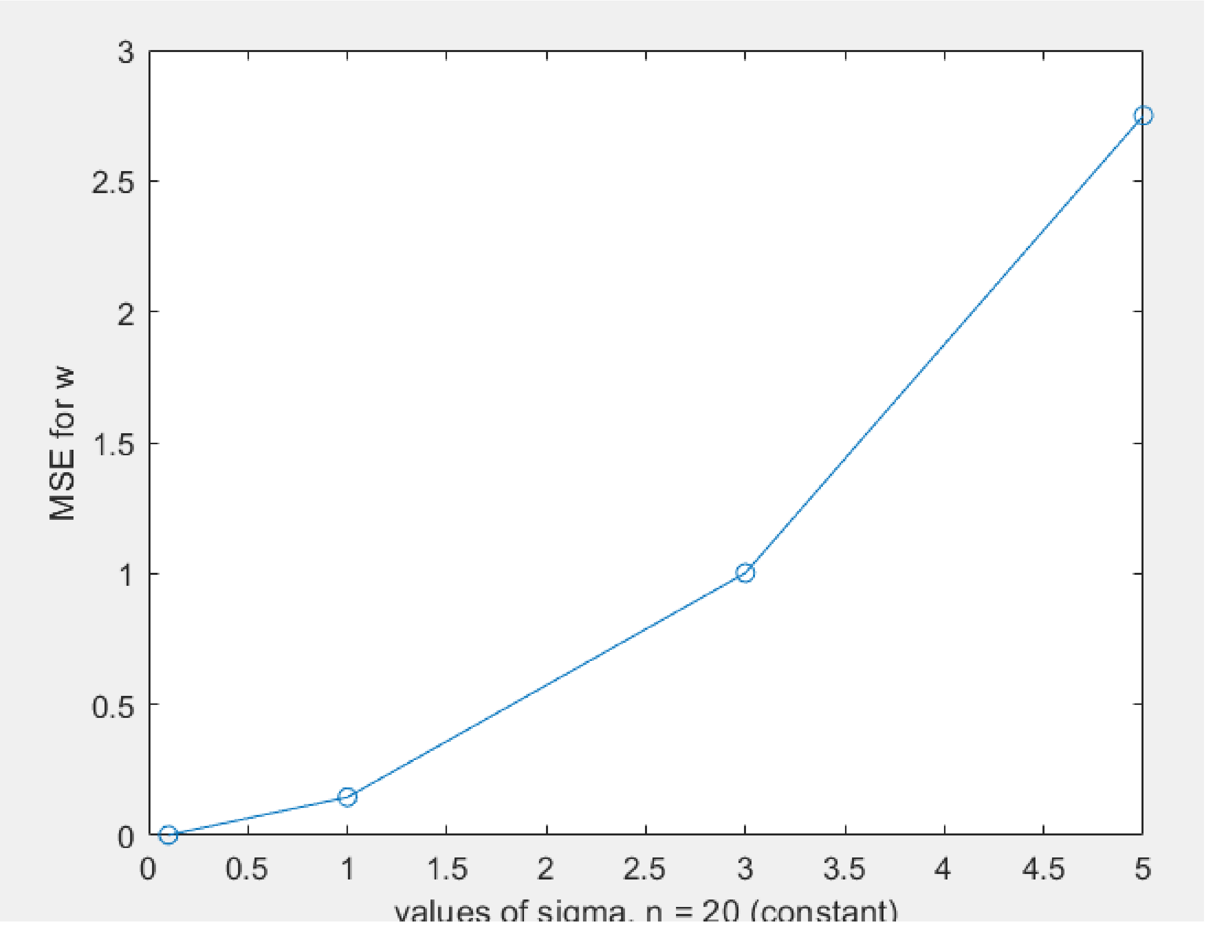
1.3314

1.6466

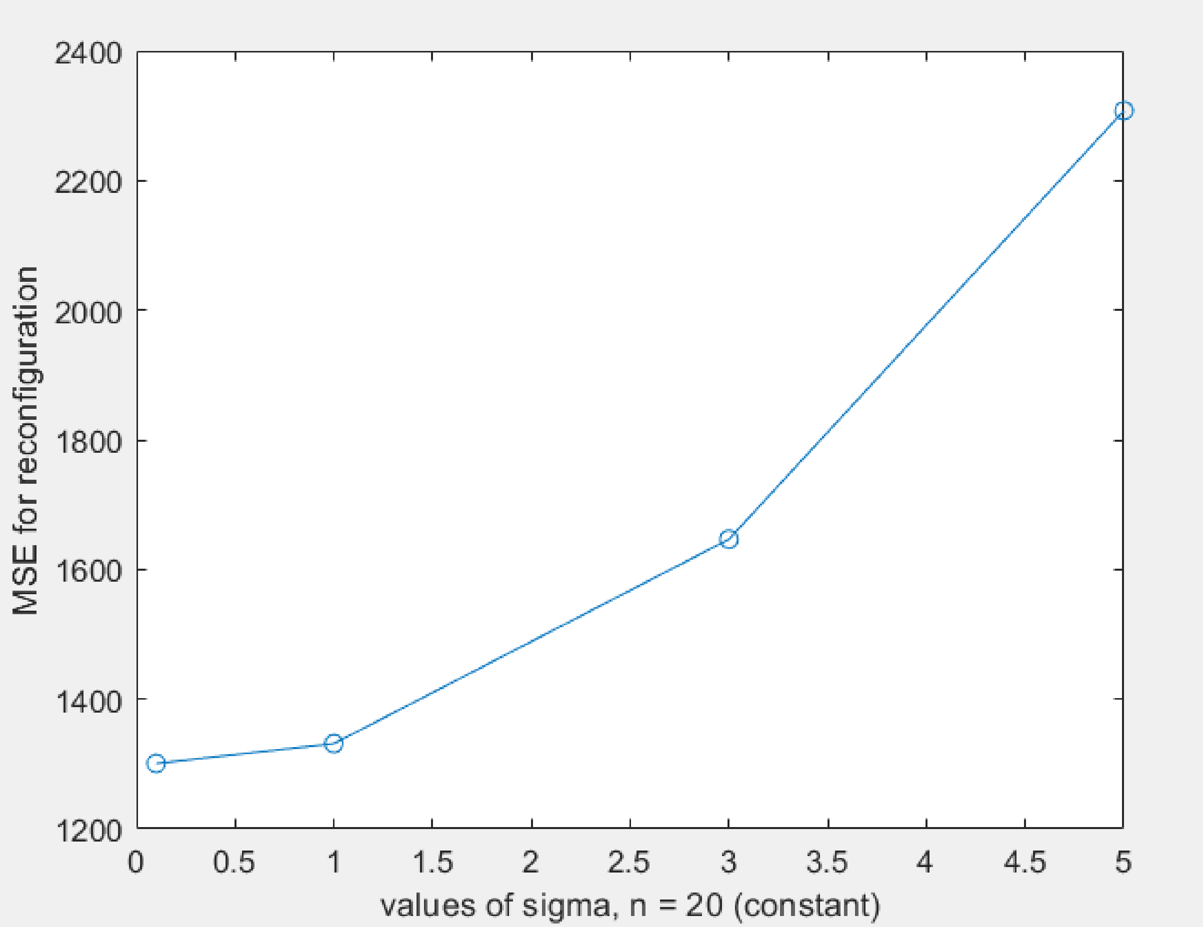
2.3078

Plots

MSE for w



MSE for Reconfiguration



As seen before, as sigma increases, both the MSE has a tendency to increase. The same previous explanation goes here as well.

iii. For Model with terms till and including x^4 (overfit)

PART 1 – Varying n with sigma constant

Code

```
% Part c_iii_Model2_Part1 (varying n, sigma constant) (Model is the one
% which is overfit to the true model)

% Things same as before

count = 0;
kernel = 3;                   % for t generation, the true one
mse_vector = zeros(5,1);
recon_mse_vector = zeros(5,1);
kernel_mis = 5;               % the overfit model
w_average_matrix = zeros(kernel_mis,5);
w_true_matrix = zeros(kernel_mis,5);


for n = [10,20,30,40,50]

    count = count + 1;
    A = zeros(2*n+1,kernel);
    sigma = 1;
    x_interim = -n:n;
    x = transpose(x_interim*0.1);  % turning to vector

    % A generation, true A, A that gives true t

    for i = 1:kernel
        A(:,i) = x.^i;
    end

    % t generation
    t = A(:,1) + 3*A(:,2) + 2*A(:,3);          % true t

    % A, for our model, generation
```

```
    A_mis = zeros(2*n+1,kernel_mis);

    for i = 1:kernel_mis
        A_mis(:,i) = x.^(i-1);              % our kernel model
    end


    w_true = pinv(A_mis)*t;
    w_true_matrix(:,count) = w_true;

    w_average_interim = zeros(kernel_mis,1);

    mse_interim = 0;

    recon_mse_interim = 0;

    for j = 1:100

        % v generation
        v = sigma*randn(2*n+1,1);

        % y generation
        y = t + v;           % observed y

        w_estimated = pinv(A_mis)*y;       % calculating the estimate of w

        y_estimated = A_mis*w_estimated;

        w_average_interim = w_average_interim + w_estimated;  % running sum of w

        mse_interim = mse_interim + (norm(w_true - w_estimated))^2; % running su
```

```
            recon_mse_interim = recon_mse_interim + (norm(y - y_estimated))^2

        end

        mse = mse_interim/100;
        recon_mse = recon_mse_interim/100;
        w_average = w_average_interim/100;

        mse_vector(count,1) = mse;
        recon_mse_vector(count,1) = recon_mse;
        w_average_matrix(:,count) = w_average;



    end

    n_range = [10;20;30;40;50];
    disp("The matrix of true w for our model, where each column corresponds t
    disp(w_true_matrix)
    disp("The matrix of estimated w for our model(averaged over 100 iteration
    disp(w_average_matrix)
    disp("The averaged MSE of w over 100 iterations for each n, where rows re
    disp(mse_vector)
    disp("The averaged MSE of reconfiguration over 100 iterations for each n,
    disp(recon_mse_vector)

    figure(1);
    plot(n_range,mse_vector,'o-')
    xlabel('values of n, sigma = 1 (constant)')
    ylabel('MSE for w')

```

```

    figure(1);
    plot(n_range,mse_vector,'o-')
    xlabel('values of n, sigma = 1 (constant)')
    ylabel('MSE for w')

    figure(2);
    plot(n_range,recon_mse_vector,'o-')
    xlabel('values of n, sigma = 1 (constant)')
    ylabel('MSE for reconfiguration')



```

Results

>> c_iii_Model2_Part1

The matrix of true w for our model, where each column corresponds to a value of n and the column vector the vector of w, is

0.0000	0.0000	-0.0000	0.0000	-0.0000
1.0000	1.0000	1.0000	1.0000	1.0000
3.0000	3.0000	3.0000	3.0000	3.0000
2.0000	2.0000	2.0000	2.0000	2.0000
0.0000	0.0000	-0.0000	-0.0000	-0.0000

The matrix of estimated w for our model(averaged over 100 iterations, for each n), where each column corresponds to a value of n and the column vector the vector of averaged w estimate, is

-0.0809	-0.0246	-0.0109	-0.0084	0.0028
0.9675	1.0193	0.9783	1.0082	0.9960
3.3731	3.0623	3.0005	2.9953	2.9994
2.0593	1.9947	2.0051	1.9995	2.0006
-0.2965	-0.0184	-0.0003	0.0005	-0.0000

The averaged MSE of w over 100 iterations for each n, where rows represent MSE of w value for each n, is

14.9549

0.4332

0.1175

0.0672

0.0372

The averaged MSE of reconfiguration over 100 iterations for each n, where rows represent MSE of reconfiguration value for each n, is

14.9236

35.6173

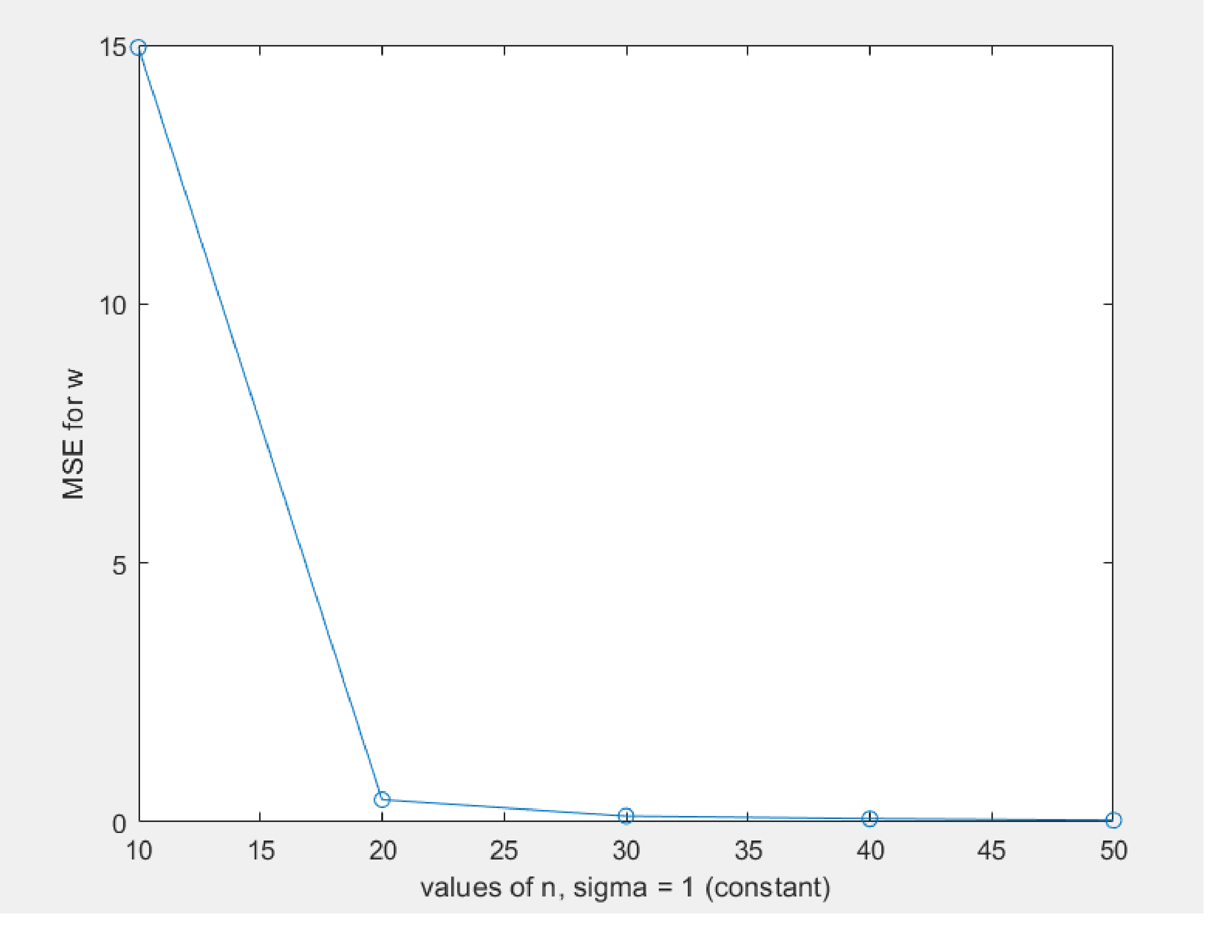
56.9790

76.7711

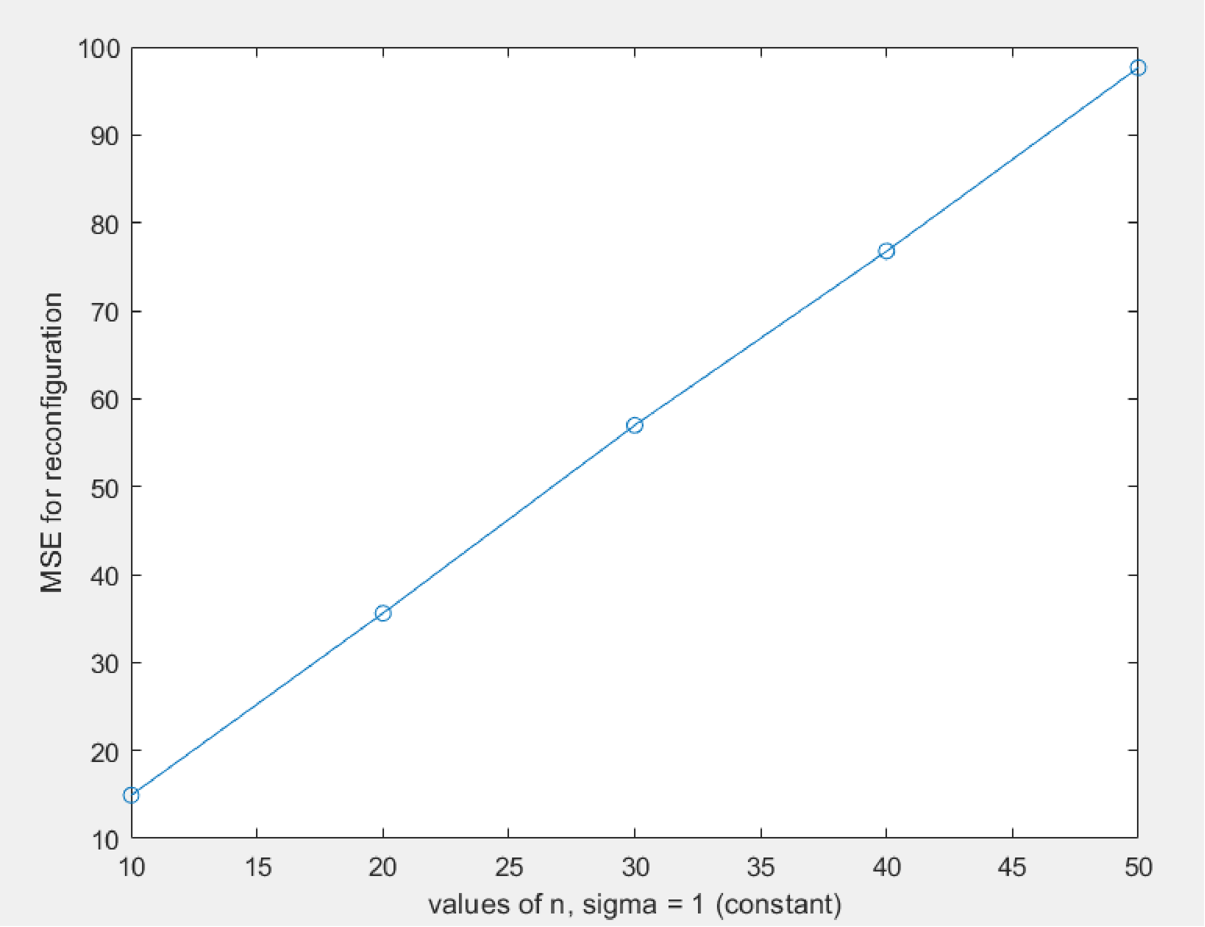
97.6454

Plots

MSE for w



MSE for Reconstruction



As can be seen the MSE for the overfit case is much less than that for the underfit case. Hence, it is obvious that if MSE of reconstruction is our main criteria than overfit model should be selected since it gives us an MSE which is extremely less in comparison to the underfit model.

PART 1 – Varying sigma with n constant

Code

```
% Part c_iii_Model2_Part2 (varying sigma, n constant) (Model is the one
% which is overfit to the true model)

% Everything same, but now only sigma varies


 count = 0;
 kernel = 3;                 % for t generation, the true one
 mse_vector = zeros(4,1);
 recon_mse_vector = zeros(4,1);
 kernel_mis = 5;
 w_average_matrix = zeros(kernel_mis,4);


n = 20;
A = zeros(2*n+1,kernel);
x_interim = -n:n;
x = transpose(x_interim*0.1);  % turning to vector

% A generation, true A, A that gives true t

for i = 1:kernel
    A(:,i) = x.^i;
end

% t generation
t = A(:,1) + 3*A(:,2) + 2*A(:,3);         % true t

% A, for our model, generation

A_mis = zeros(2*n+1,kernel_mis);

```

```
for i = 1:kernel_mis
    A_mis(:,i) = x.^(i-1);              % our kernel model
end

pm_i = pinv(A_mis);
w_true = pm_i*t;

 for sigma = [0.1,1,3,5]

    count = count + 1;

    w_average_interim = zeros(kernel_mis,1);

    mse_interim = 0;

    recon_mse_interim = 0;

    for j = 1:100

        % v generation
        v = sigma*randn(2*n+1,1);

        % y generation
        y = t + v;          % observed y

        w_estimated = pm_i*y;       % calculating the estimate of w

        y_estimated = A_mis*w_estimated;

        w_average_interim = w_average_interim + w_estimated;  % running sum

        mse_interim = mse_interim + (norm(w_true - w_estimated))^2; % runni
```

```
        recon_mse_interim = recon_mse_interim + (norm(y - y_estimated))^2;

    end

    mse = mse_interim/100;
    recon_mse = recon_mse_interim/100;
    w_average = w_average_interim/100;

    mse_vector(count,1) = mse;
    recon_mse_vector(count,1) = recon_mse;
    w_average_matrix(:,count) = w_average;



end

sigma_range = [0.1;1;3;5];
disp("The true w vector for our model, which is obviously constant for all
disp(w_true)
disp("The matrix of estimated w for our model(averaged over 100 iterations,
disp(w_average_matrix)
disp("The averaged MSE of w over 100 iterations for each sigma, where rows
disp(mse_vector)
disp("The averaged MSE of reconfiguration over 100 iterations for each sigı
disp(recon_mse_vector)

figure(1);
plot(sigma_range,mse_vector,'o-')
xlabel('values of sigma, n = 20 (constant)')
ylabel('MSE for w')
```

```

figure(2);
plot(sigma_range,recon_mse_vector,'o-')
xlabel('values of sigma, n = 20 (constant)')
ylabel('MSE for reconfiguration')
```

Results

>> c_iii_Model2_Part2

The true w vector for our model, which is obviously constant for all sigma, since n is constant is

0.0000

1.0000

3.0000

2.0000

0.0000

The matrix of estimated w for our model(averaged over 100 iterations, for each sigma), where each column corresponds to a value of sigma and the column vector the vector of averaged w estimate, is

-0.0032	-0.0155	0.1085	-0.0525
1.0011	1.0133	0.8659	1.0348
3.0037	3.0665	2.9011	3.0538
2.0001	2.0015	2.0399	1.9652
-0.0007	-0.0222	0.0186	-0.0108

The averaged MSE of w over 100 iterations for each sigma, where rows represent MSE of w value for each sigma, is

0.0043

0.3726

3.6304

10.5548

The averaged MSE of reconfiguration over 100 iterations for each sigma, where rows represent MSE of reconfiguration value for each sigma, is

0.3584

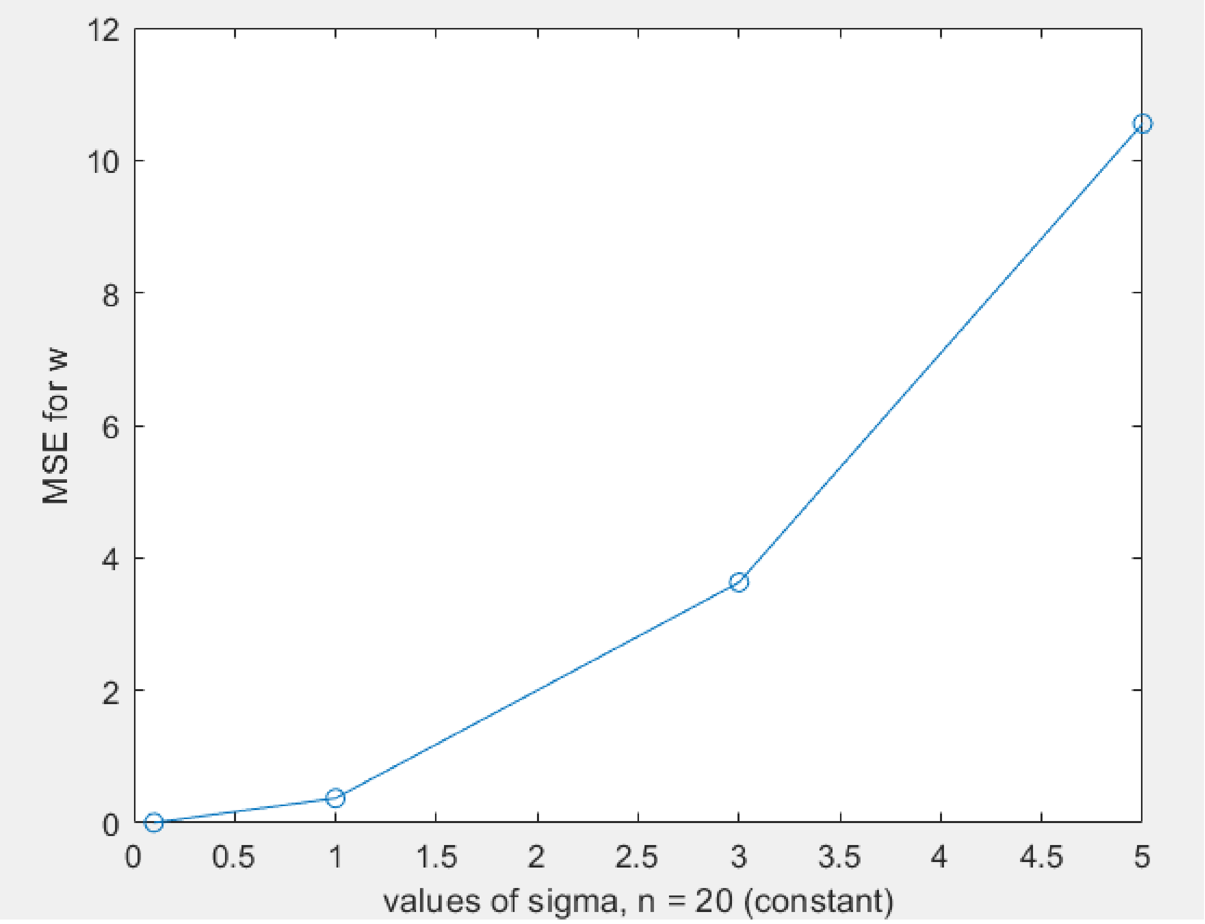
37.0618

318.1502

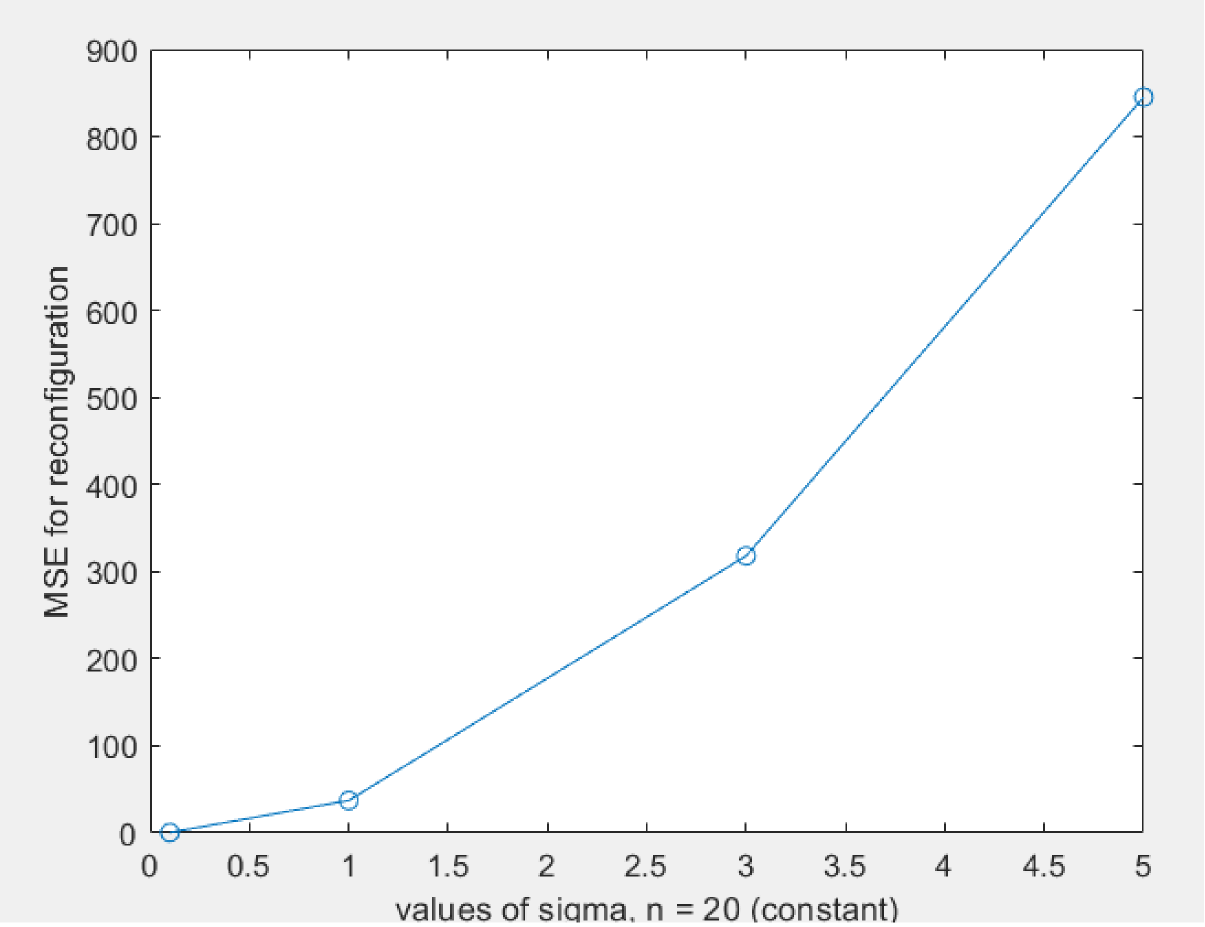
845.2264

Plots

MSE for w



MSE for Reconstruction



Similar results with similar explanation

iv. Comment on Results

It is definitely worse, MSE wise, to underestimate/underfit the model. Thus, the underestimated/underfit model had an excruciatingly worse MSE compared to the overfit/overestimated version which had a comparatively much better MSE.

However, with overfitting we do have an issue where the model is so overfit, that we can't have good model generalization to the test data, increasing the test MSE. This fact is also known as the bias-variance trade-off in machine learning algorithms.

Underfit model, have, higher training MSE but generalize better and may have low test MSE, sometimes even lower than the Overfit model.

-----------------------------END---