Assignment 3

Determining and removing drawbacks of exponential and running mean

By

Group (19)

Ahmed Baza, Skoltech, 2020

Ahmed Elgohary, Skoltech, 2020

Mohamed Sayed, Skoltech, 2020

Ziad Baraka, Skoltech, 2020

Part1: Backward exponential smoothing

As we saw in the last part of assignment II, the forward smothing introduced the shift problem in this part the same trajectory will be created and the Backward mothing will be applied to elemnate this problem.

1.1) Generating the trajectory witht the same statistical characteristics ($\sigma_w^2 = 28^2, \sigma_\eta^2 = 97^2$).

```
N1=300;     % size of the array
X1 = zeros(N1,1);   % real data
Z1 = zeros(N1,1);   % measurements
sw = 28^2;      % sigma_w^2
se = 97^2;      % sigma_Eita^2
x1 = sw / se;   % xita
alpha1 = (-x1 + sqrt(x1^2 +4*x1))/2;    % coefficient for exponential smoothing
M1 = (2-alpha1)/alpha1;                 % window size for running mean


X1(1) = 10;
Z1(1) = X1(1) + normrnd(0,sqrt(se));

for i = 2:N1
    X1(i) = X1(i-1) + normrnd(0,sqrt(sw));
    Z1(i) = X1(i) + normrnd(0,sqrt(se));
end

% forward exponential
X1_exp= zeros(N1,1);    % forward exponential smoothed trajectory.
X1_exp(1) = Z1(1);
for i = 2:N1
X1_exp(i) = X1_exp(i-1) + alpha1 * (Z1(i) - X1_exp(i-1));
end

% backward exponential
```

```
X1_back= zeros(N1,1);   % backward exponential smoothed trajectory.
X1_back(N1) = X1_exp(N1);
for i = N1-1:-1:1
X1_back(i) = X1_exp(i+1) + alpha1 * (X1_exp(i) - X1_back(i+1));
end

% running mean
X1_mean= zeros(N1,1);   % running mean smoothed trajectory.
M1 = round((M1-1)/2)*2+1;
m1=(M1-1)/2;
X1_mean(1:m1)=sum(Z1(1:m1))/m1;
X1_mean(N1-m1+1:N1)=sum(Z1(N1-m1+1:N1))/m1;
for i =  (m1+1):N1-m1
X1_mean(i) = 1/M1 * sum(Z1(i-m1:i+m1));
end

figure(1)
subplot(2,1,1)
plot(1:N1,Z1,1:N1,X1,1:N1,X1_exp,1:N1, X1_back)
title('Figure (1.1) Measurements, Real data, F_exponential, and B_exponential')
xlabel('Steps')
ylabel('Data')
legend({'measurnments','data','forward','backward'})
subplot(2,1,2)
plot(1:N1,Z1,1:N1,X1,1:N1,X1_mean)
title('Figure (1.2) Real data, measurements,running mean')
xlabel('Steps')
ylabel('Data')
legend({'data','measurnments','mean'})
```

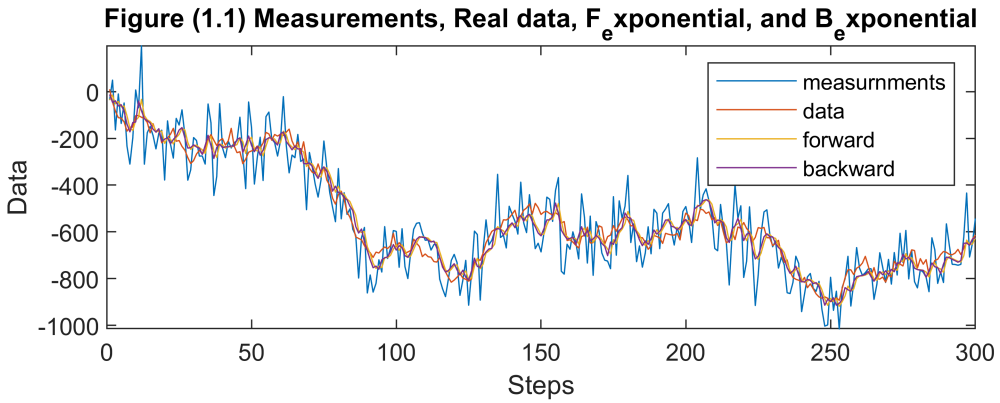


Figure (1.1) Measurements, Real data, F$_e$xponential, and B$_e$xponential

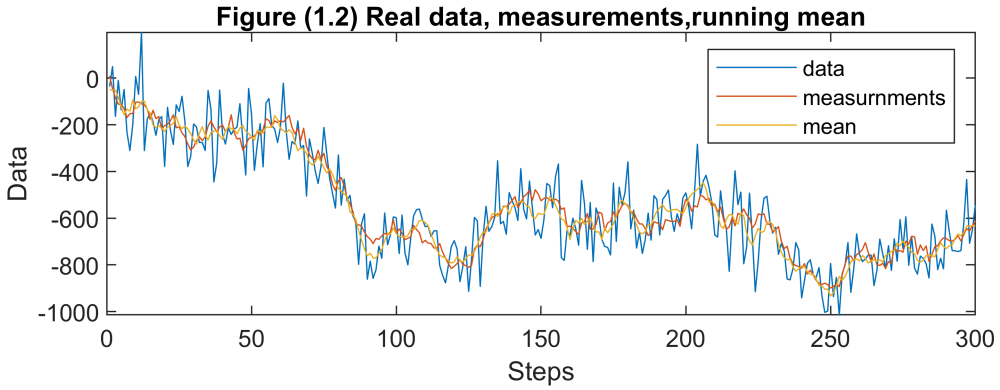


Figure (1.2) Real data, measurements,running mean

In figure(1), Plotting the 4 curves, the true trajectory, the measurements, the exponential smoothing and the running mean. After examining the plot we can see that the delay in the exponential mean method was decreased after applying the backward smoothing and that the curve now following better the true trajectory than the forward smoothing only.

The running mean can be visually compared to the exponential mean and we believe that it's following the true trajectory better –in this particular case- but it still has a gap between it and the true values in each of the peaks.

Calculating deviation and variability indicator

```
Id1_exp = sum((Z1-X1_back).^2)     % deviation indecator of backward smoothing
```

```
Id1_exp = 2.2121e+06
```

```
Iv1_array_exp = zeros(N1,1);
for i = 1:N1-2
    Iv1_array_exp(i)=(X1_back(i+2) - 2*X1_back(i+1) + X1_back(i))^2;
end
Iv1_exp = sum(Iv1_array_exp)       % variability indecator of backward smoothing
```

```
Iv1_exp = 4.3808e+05
```

```
Id1_mean = sum((Z1-X1_mean).^2)    % deviation indecator of Running mean.
```

```
Id1_mean = 2.7277e+06
```

```
Iv1_array_mean = zeros(N1,1);
for i = 1:N1-2
    Iv1_array_mean(i)=(X1_mean(i+2) - 2*X1_mean(i+1) + X1_mean(i))^2;
end
Iv1_mean = sum(Iv1_array_mean)     % variability indecator of Running mean.
```

```
Iv1_mean = 2.4491e+05
```

Using the deviation and variability indicators, the running mean had the better advantage for this problem. The deviation indicator Id was for both methods extremely close in values with an order of (1e+6); for example when it was 2.7277e+6 for the running mean, it was 2.2121e+6 for the forward-backward exponential smoothing, the exponential smoothing had better value for filtering the data but with no significant difference over the running mean. The variability indicator Iv was also of the same order of (1e +5)for both methods; for example when it was 2.4491e+5 for the running mean, it was 4.3808e+5 for the forward-backward exponential smoothing, which is a better result overall to use the running mean for our problem because the difference in Iv values was more significant than the difference in Id values.

Part 2: Drawbacks of running mean

First, we will analyze a process which rate of change is changed insignificantly and measurement noise is great. Second, we will study a cyclic process, and measurement noise is small.

2.1) Generateing a true trajectory X_i of an object motion disturbed by normally distributed random acceleration. and mesurment Z_i for the trajectory

$$X_i = X_{i-1} + V_{i-1}T + a_{i-1}\frac{T^2}{2}$$
$$V_i = V_{i-1} + a_{i-1}T$$
$$Z_i = X_i + \eta$$

Size of trajectory is 300 points.

Initial conditions: $X_1 = 5;\ V_1 = 0;\ T = 0.1;$

Variance of noise a_i : $\sigma_a^2 = 10$

Variance of noise Z_i : $\sigma_\eta^2 = 500$

```
clc; clear; close all;
N = 300;
X = zeros(N,1);     % true trajectory points
V = zeros(N,1);     % true trajectory points
Z = zeros(N,1);     % measurnments
X(1) = 5;           % initial condition
```

```
V(1) = 0;           % initial condition
Z(1) = X(1) + normrnd(0,sqrt(500));
T = 0.1;
for i = 2:N
    a = normrnd(0,sqrt(10));
    X(i) = X(i-1) + V(i-1) * T +  (a*T^2) /2;
    V(i) = V(i-1) + a*T;
    Z(i) = X(i) + normrnd(0,sqrt(500));
end

% plot
figure(2)
plot(1:N,X,1:N,Z)
title('Figure (2) Real data vs measurements')
xlabel('Steps')
ylabel('Data')
legend({'real data','measurements'})
```

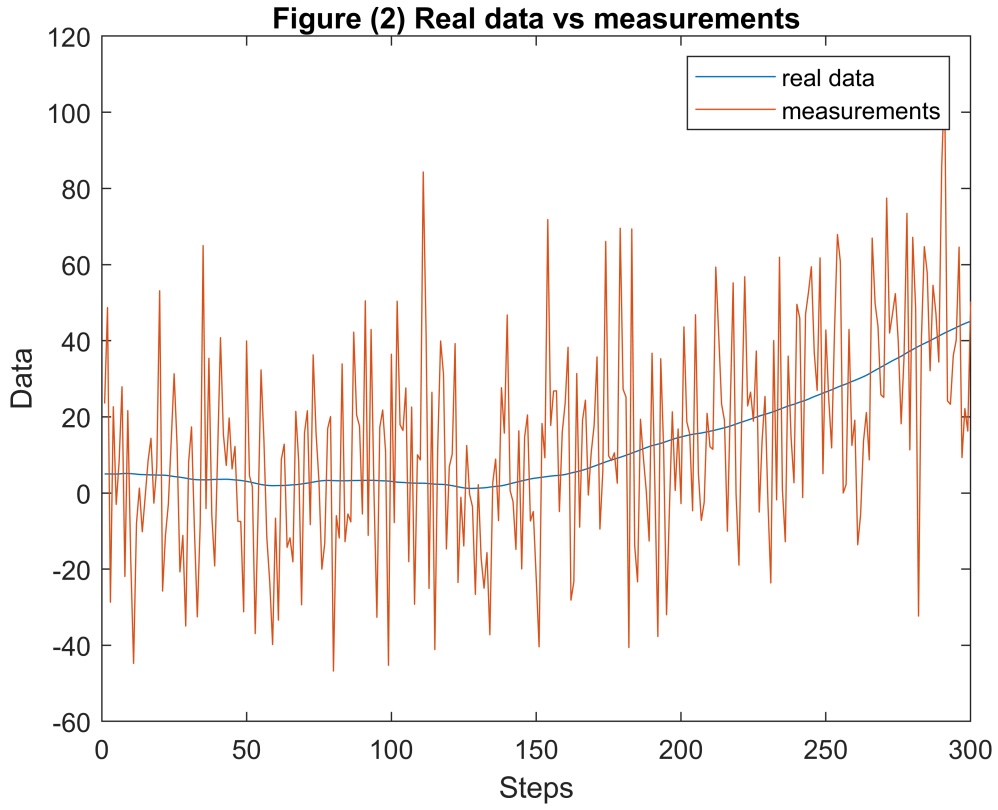


Figure (2) above shows the generated real data compared with the generated measurements using the given noise variance values.

2- Determine empirically the window size M of running mean and smoothing coefficeint α (forward exponential smoothing) that provide the best estimation of the process X using meaurnments Z.

This can be acheived by drawing with mutliple values of M, alpha and decide upon the values the gives the closest graphs to the real data.

For the next part as we will explain we used huge number of plots to compare the values visually, it would be impractical to add all of them to the report so we are just going to focus on the interesting regions we explored

```
X_exp= zeros(N,1);
X_exp(1) = Z(1);
Iv_array_exp = zeros(N,1);

figure(3);

k=1; % dummy variable (counter)
for alpha = 0.01:0.01:0.23
    for i = 2:N
        X_exp(i) = X_exp(i-1) + alpha * (Z(i) - X_exp(i-1));
    end
    Id_exp(k) = sum((Z-X_exp).^2);
    for i = 1:N-2
        Iv_array_exp(i)=(X_exp(i+2) - 2*X_exp(i+1) + X_exp(i))^2;
    end
    Iv_exp(k) = sum(Iv_array_exp);
    alpha_array(k) = alpha;

    subplot(5,5,k);
    %plot(1:N,Z,1:N,X,1:N,X_exp)
    plot(1:N,X,1:N,X_exp)
    xlabel('Steps')
    ylabel('Data')
    title(sprintf('alpha = %.2f', alpha))

    k=k+1;
end
subplot(5,5,25)
plot(0,0,0,0,0,0)
axis off
%legend({'Measurements','Real Data','Exponential Data'})
legend({'Real Data','Exponential Data'})
```

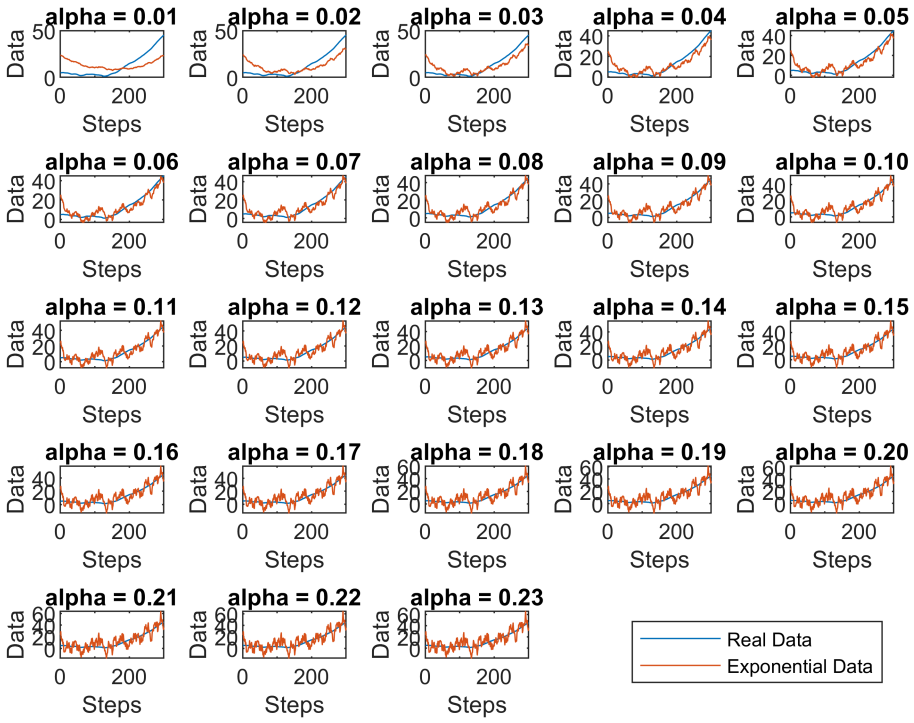


Figure (3) above shows the real data curve compared with the measurements and the exponential smoothing of these measurements. Different alpha values were used for exponential smoothing. From the figure, it is very clear that as alpha values increase as we beleive more in the measurements and have lesss noise filteration. So, less alpha values give better filteration and more closer to the real data.

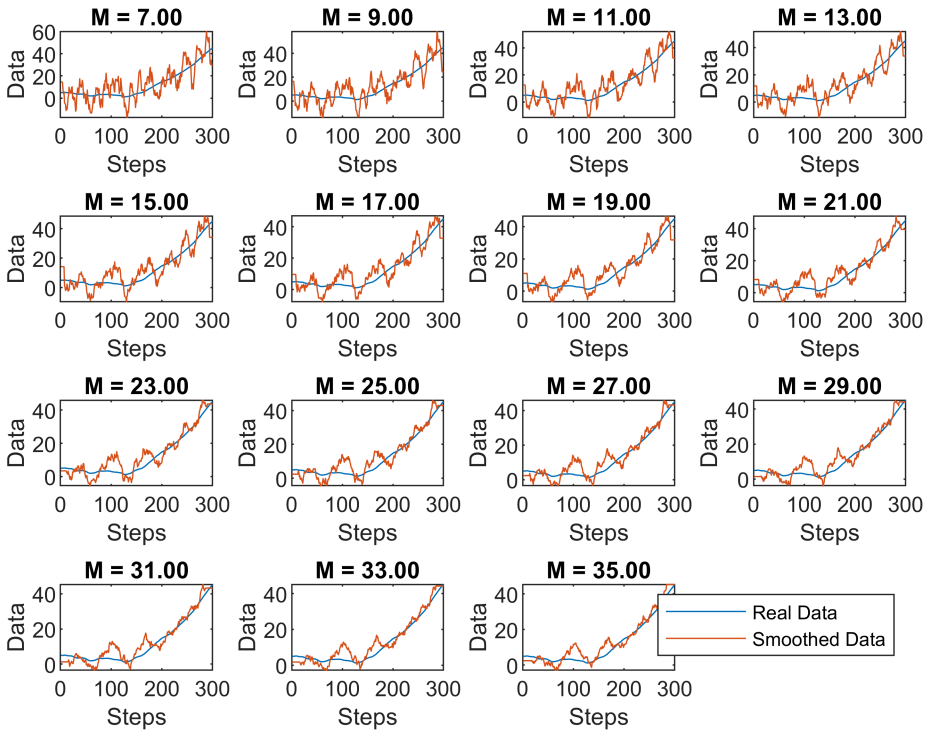
```
X_mean= zeros(N,1);

figure(4);
%title('Figure (3) Real data, Running mean data, and  measurnments')
k=1;
for M = 7:2:35
    m=(M-1)/2;
    X_mean(1:m)=sum(Z(1:m))/m;
    X_mean(N-m+1:N)=sum(Z(N-m+1:N))/m;
    for i =  (m+1):N-m
    X_mean(i) = 1/M * sum(Z(i-m:i+m));
    end
Id_mean(k) = sum((Z-X_mean).^2);
Iv_array_mean = zeros(N,1);
for i = 1:N-2
    Iv_array_mean(i)=(X_mean(i+2) - 2*X_mean(i+1) + X_mean(i))^2;
end
Iv_mean(k) = sum(Iv_array_mean);
M_array(k) = M;

subplot(4,4,k);
%plot(1:N,Z,1:N,X,1:N,X_mean)
```

```
plot(1:N,X,1:N,X_mean)
xlabel('Steps')
ylabel('Data')
title(sprintf('M = %.2f', M))

k=k+1;
end
subplot(4,4,16)
plot(0,0,0,0,0,0)
axis off
%legend({'Measurements','Real Data','Running mean Data'})
legend({'Real Data','Smoothed Data'})
```



In Figure (4) above, we can see the real data curve compared with the measurements and the smoothed measurments using the running mean method. Different window size M values were used. From this figure, it is clear that bigger window size gives more effective smoothing but more slower to change in dynamics. On the other hand, with smaller window size, the model is more dynamic but with less smoothing effect and affected more by disturbances in measurment.

For choosing the best range for running window (M) for this particular problem we used some step, at first, using common sense we came to the belief that any value high enough and less than half of the number of measurements data would be sufficient for this particular problem as it comes close to a line and doesn't have a lot of dynamics, and by high enough we thought it would be something like M = 21. Secondly we experimented values for the running window we started from 3 till 49 with step +2 as shown in the figure and then used higher range of values till 151 and we came to the conclusion that a window starting form M=11 to M=101 would be following the curve but at high numbers of M the first and last set of data lose completely

the dynamics of the system. So an acceptable range might be from 11 to 49 but a very good approximation would be from 11 to 25 which is almost 1/12 to 1/8 the number of data acquired. With this window we believe the measurements' noise is well filtered and the trajectory didn't lose a lot of the dynamics.

And for choosing a better alpha we also used the same procedure we did in choosing M. using common sense we thought the value would be something between 0.05 and 0.5. And running our alpha values from 0 to 1 with a step of 0.1 we found the range of data is better approximated between 0 to 0.2, so we ran our numbers again with a smaller step of 0.01 and came to the conclusion that an alpha value between 0.08 and 0.14 was a good approximation as any value less than 0.08 was over-smoothed and started to lose they dynamics, whilst a value higher was considered too noisy.

3. Chose better smoothing method using deviation and variability indicators.

```
M_used = 11;            % chosen window size M
alpha_used = 0.08;      % chosen alpha
alpha_index=0;
for i=1:length(alpha_array)
if alpha_array(i) > alpha_used-0.01  && alpha_array(i) < alpha_used+0.01
    alpha_index=i;
end
end
Id_exp_value = Id_exp(alpha_index)  % Deviation indicator for exponential smoothing
```

```
Id_exp_value = 1.6598e+05
```

```
Iv_exp_value = Iv_exp(alpha_index)  % Variability indecator of forward smoothing
```

```
Iv_exp_value = 2.5621e+03
```

```
M_index = find(M_array == M_used);
Id_mean_value = Id_mean(M_index)    % Deviation indecator of Running mean.
```

```
Id_mean_value = 1.6325e+05
```

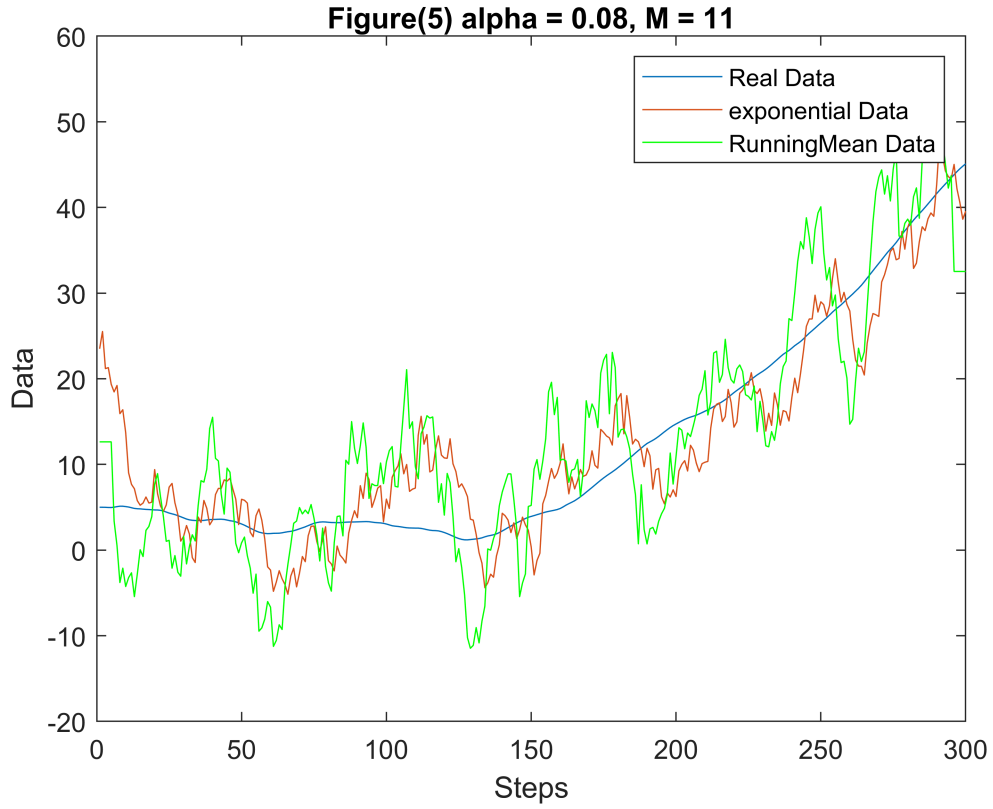
```
Iv_mean_value = Iv_mean(M_index)     % Variability indecator of Running mean.
```

```
Iv_mean_value = 6.2916e+03
```

```
% Plot
X_mean= zeros(N,1);
m=(M_used-1)/2;
X_mean(1:m)=sum(Z(1:m))/m;
X_mean(N-m+1:N)=sum(Z(N-m+1:N))/m;
for i =  (m+1):N-m
X_mean(i) = 1/M_used * sum(Z(i-m:i+m));
end

X_exp= zeros(N,1);
X_exp(1) = Z(1);
for i = 2:N
X_exp(i) = X_exp(i-1) + alpha_used * (Z(i) - X_exp(i-1));
end
```

```
figure(5)
plot(1:N,X,1:N,X_exp,1:N,X_mean,'g')
xlabel('Steps')
ylabel('Data')
title(sprintf('Figure(5) alpha = 0.08, M = 11'))
legend({'Real Data','exponential Data','RunningMean Data'})
```



Now using the best value in our range which was M=11 and alpha=0.08, comparing the deviation and variability for our problem, both methods had the same order for this particular problem, the deviation indicator was of (1e+5) order and the variability indicator was of (1e+3) order. So both methods were similar in indicator orders which mean they can be similarly used for this problem, but if we look closer to the numbers for example in one of the runs we can see that in the running mean Id=1.6325e+5, and for the exponential smoothing Id=1.6598e+5, also Iv=6.2916e+3 in running mean and in exponential smoothing Iv=2.5621e+3. Which means that the exponential smoothing was slightly better than the running mean.

Second trajectory

4&5. Generate cyclic trajectory and measurements of the process

```
Nc = 200;       % Size of this cyclic trajectory
A = zeros(Nc,1);
Xc = zeros(Nc,1);   % Data for the cyclic trajectory
Zc = zeros(Nc,1);   % measurnments

T = 32;         % Period of estimation
```

```
w = 2*pi/T;     % (Omega) angle frequency
A(1) = 1;         %  Initial condition
Xc(1) = A(1) +  sin(w + 3);
Zc(1) = Xc(1) + normrnd(0,sqrt(0.05));

for i = 2:Nc
    A(i) = A(i-1) +  normrnd(0,0.08);
    Xc(i) = A(i) +  sin(w*i + 3);
    Zc(i) = Xc(i) + normrnd(0,sqrt(0.05));
end

% plot
figure(6)
plot(1:Nc,Xc,1:Nc,Zc)
title('Figure (6) Real cyclic data vs measurnments')
xlabel('Steps')
ylabel('Data')
legend({'real data','measurements'})
```

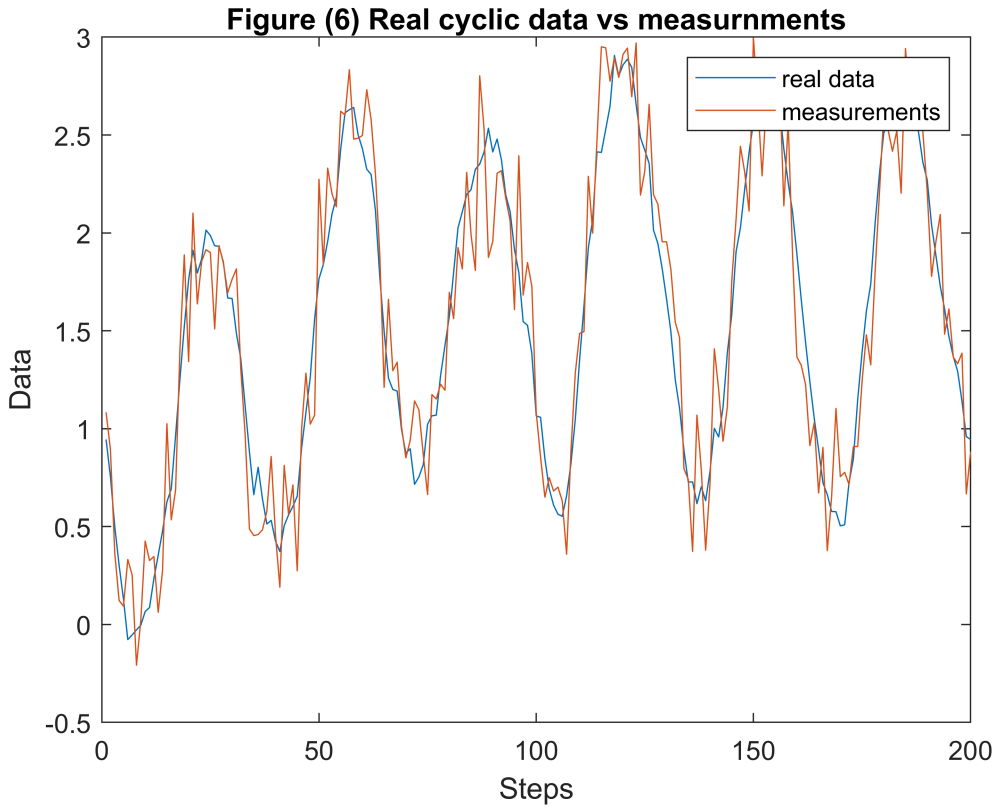


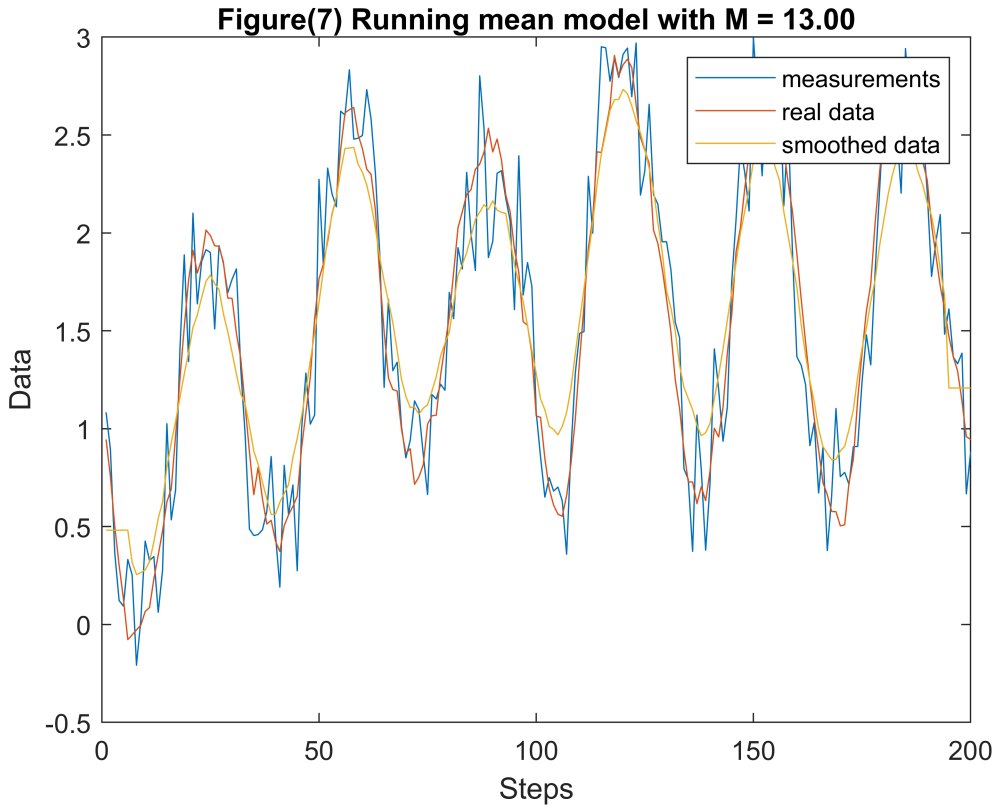
Figure (6) above shows the generated cyclic data compared with the its generated measurements using the given noise variance values.

6. Apply running mean with window size M=13 to these measurements

```
Mc= 13;     % window size used to smooth the cyclic trajectory data
mc=(Mc-1)/2;
X_mean_c = zeros(Nc,1);
X_mean_c(1:mc)=sum(Zc(1:mc))/mc;
```

```
X_mean_c(Nc-mc+1:Nc)=sum(Zc(Nc-mc+1:Nc))/mc;
for i =  (mc+1):Nc-mc
X_mean_c(i) = 1/Mc * sum(Zc(i-mc:i+mc));
end

figure(7)
plot(1:Nc,Zc,1:Nc,Xc,1:Nc,X_mean_c)
xlabel('Steps')
ylabel('Data')
legend({'measurements','real data','smoothed data'})
title(sprintf('Figure(7) Running mean model with M = %.2f', Mc))
```



IN figure (7) above, we compare the smoothed data using the running mean method with window size M=13 with both the real data and its measurements. This M value gives quite good results as model is much closer to the real data with effective nosie filteration. However, there are still some difference that apperas more clearly around the peaks and the troughs.

7.Determine the period of oscillations for which running mean with given window size M = 21

a) produces inverse oscillations

b) leads to the loss of oscillations (zero oscillations)

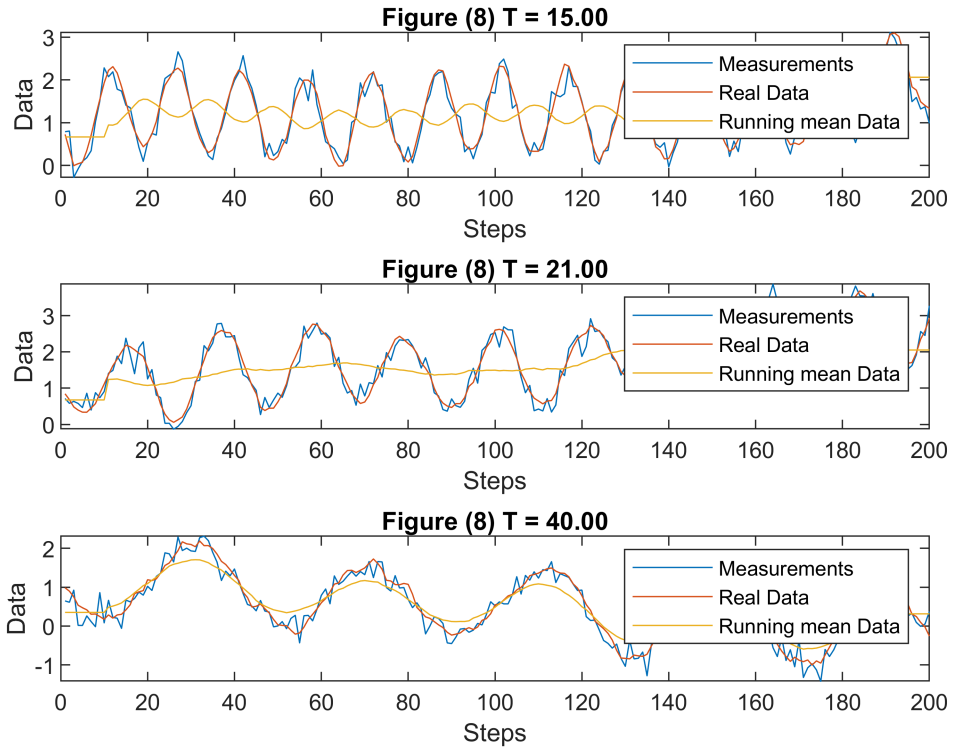
c) changes the oscillations insignificantly

```
figure(8);

T_array=[15 21 40];

for k = 1:3
T = T_array(k);
    w = 2*pi/T;     % (Omega) angle frequency
    A1 = 1;         %  Initial condition
    Xc(1) = A(1) +  sin(w + 3);
    Zc(1) = Xc(1) + normrnd(0,sqrt(0.05));

    for i = 2:Nc
        A(i) = A(i-1) +  normrnd(0,0.08);
        Xc(i) = A(i) +  sin(w*i + 3);
        Zc(i) = Xc(i) + normrnd(0,sqrt(0.05));
    end
    Mc = 21;  % worked with group 4 data
    mc=(Mc-1)/2;
    X_mean_c(1:mc)=sum(Zc(1:mc))/mc;
    X_mean_c(Nc-mc+1:Nc)=sum(Zc(Nc-mc+1:Nc))/mc;
    for i =  (mc+1):Nc-mc
        X_mean_c(i) = 1/Mc * sum(Zc(i-mc:i+mc));
    end
    subplot(3,1,k)
    plot(1:Nc,Zc,1:Nc,Xc,1:Nc,X_mean_c)
    xlabel('Steps')
    ylabel('Data')
    title(sprintf('Figure (8) T = %.2f', T))
    legend({'Measurements','Real Data','Running mean Data'})
end
```



Figure(8) above shows the smoothed data using running mean method compared with the real data and its measurements. For a given window size M, different period of oscillations values were used to generate the cyclic data, its measurments, and finally smoothing it using this M value.

T = [11:20] Period to produce inverse oscillations

T = 21 Period that leads to loss of oscillations

T > 21 Period to slightly change oscillations

For M =21 we can see that

a) For a range where the period would produce an inverse oscillations T would be T=11 to 20 because less than that would produce an accurate oscillation but total loss of values.

b) If we use T=21 steps then the smoothing produces zero oscillations as the period equals the running window and it cannot capture the physics in that case.

c) For a range higher than 21 steps for T the smoothing should not disturb the process and capture it but it gives only the shift in the peaks and troughs. And that shift decreases as T gets higher and increases as it comes close to M.

Learning log:

In this assignment we have continued on the previous one to have a better understanding of smoothing methods and how they work with different trajectories, also it gave us the practical sense of smoothing with exponential or running mean smoothing. in addition to comparing the running mean with the exponential smoothing and to what extent we can depend on them for our calculations and how their behavior is affected by their parameters. We also tried to decide upon the suitable M and alpha values for the smoothing of our generation data. We also learnt how the non suitable window size M for the running mean method can lead to some fatal problems when using cyclic data. It might invert the oscillations or even leads to the loss of these oscillations. Finally, summarizing this conclusion in bullet points to rely on in future analysis they would be:

1- What makes the forward smoothing shifts and how the backward smoothing helps solving that shift problem.

2- Determining which method is better by using the deviation and variability indicators and that was more evident when we tried different values of running mean windows and alphas.

3- In choosing the best running window we were stuck because our system was relatively not too complex to give us the sense to reflect upon in the future so as a further step we decided to increase the variance of the acceleration so we can see how this problem practically be addressed with more noisy trajectories, so to say. Although we came to almost the same conclusion which is the M value would rely between 11 and 23 but this reassured us about our results and also heightened our practical sense of numbers in case of running mean windows. The higher values of M which gave us good smoothing results in the almost linear trajectory failed to follow the dynamics of the new system with the higher random acceleration variance.

4- The periodic motion part also helped us know how the behavior of smoothing would change compared to almost polynomial trajectories. In part 7 we reflected on what was discussed in class and using our common sense we were able to define the appropriate range required, which also we tried by hand in Matlab and plugged in different values from the different range we decided and had seen how the behavior was identical to our prediction.