```
%% Comparison of the exponential and running mean for random walk model

close all; clear; clc;

set(0,'defaulttextInterpreter','latex');
set(groot,'defaultAxesTickLabelInterpreter','latex');
set(groot,'defaultLegendInterpreter','latex');

%% First part: Determination of optimal smoothing constant in exponential mean

sigma_w2 = 12;
sigma_eta2 = 9;

n_1 = 300;
n_2 = 3000;

incond = 10;

rng default

[x_1, x_hat_1, z_1, alpha_1, sigma_eta2_1, sigma_w2_1] = fun(n_1, sigma_w2, ↙
sigma_eta2, incond);
[x_2, x_hat_2, z_2, alpha_2, sigma_eta2_2, sigma_w2_2] = fun(n_2, sigma_w2, ↙
sigma_eta2, incond);


% increasing the number of steps, we are getting closer to the real (given)
% sigma and eta

%Root Squared Mean Error
Error_1 = (x_1 - x_hat_1).^2;
Error_1 = sqrt(sum(Error_1)/length(Error_1));

Error_2 = (x_2 - x_hat_2).^2;
Error_2 = sqrt(sum(Error_2)/length(Error_2));

%Plots
figure(1)
plot(x_1, 'c', 'LineWidth', 1.5)
hold on
plot(z_1, 'k', 'LineWidth', 1.5)
plot(x_hat_1, 'm', 'LineWidth', 1)
grid on; grid minor
xlabel('Steps', 'FontSize', 30)
ylabel('Data', 'FontSize', 30)
legend('Trajectory', 'Measurements', 'Exponentially Smoothed Data', 'FontSize', 25)

figure(2)
plot(x_2, 'c', 'LineWidth', 1.5)
hold on
plot(z_2, 'k', 'LineWidth', 1.5)
plot(x_hat_2, 'm', 'LineWidth', 1)
grid on; grid minor
```

```
xlabel('Steps', 'FontSize', 30)
ylabel('Data', 'FontSize', 30)
legend('Trajectory', 'Measuraments', 'Exponentially Smoothed Data', 'Location',↙
'best','FontSize', 25)
xlim([0 300])

%% Second Part: Comparison of methodical errors of exponential and running mean.

% 1-2-3) Generate a true trajectory using:
% - the random walk model
% - using equation (2) {z(i)}
% 3) Determine optimal smoothing coefficient

n_3 = 300; % size of trajectory
incond = 10; % initial condition
x_n(1) = incond;

sigma_w_n = 28^2; % variance noise
sigma_eta_n = 97^2; % variance of noise measurement

a_1_n = sqrt(sigma_w_n);
a_2_n = sqrt(sigma_eta_n);

w_n = a_1_n.*randn(n_3,1);
eta_n = a_2_n.*randn(n_3,1);

for i = 2:n_3
    x_n(i) = x_n(i-1) + w_n(i); % generated trajectory RWM
end

for i=1:n_3
    z_n(i) = x_n(i) + eta_n(i); % Generate measurements of the process
end

csi_n = sigma_w_n / sigma_eta_n;
alpha_n = (-csi_n + sqrt(csi_n^2 + 4*csi_n))/2; % correct bc should be between 0,1

%% 4) Determine  the  window  size  M  (use  round  values)  that  provides
% equality  of  σRM2   and  σES2  using determined smoothing constant α

% Window size M

M = round((2-alpha_n)/alpha_n); % 7

%% 5) Apply  running  mean  using  determined  window  size  M and ...
% exponential  mean. using determined smoothing constant to measurements
% Plot true trajectory, measurementd, running and exponential mean.

% Running mean (last measurements are used)
j = (M-1)/2;

x_hat_run = zeros(n_3,1);
```

```
x_hat_run(1:j,1) = sum(z_n(1:j))/3;
x_hat_run((n_3-j+1):n_3) = sum(z_n((n_3-j+1):n_3))/3;

for i = (j+1):(n_3-j)
    x_hat_run(i) = 1/M * (z_n(i-3)+ z_n(i-2) + z_n(i-1) + z_n(i) + ...
    z_n(i+1) + z_n(i+2) + z_n(i+3));
end

% Exponential mean (all previous measurements are used)
x_hat_exp(1) = incond;

for i = 2:n_3
    x_hat_exp(i) = x_hat_exp(i-1) + alpha_n * (z_n(i) - x_hat_exp(i-1));
end

figure(3)
plot(x_n, 'k', 'LineWidth', 1.2)
hold on
plot(z_n, 'MarkerFaceColor', [0.9290 0.6940 0.1250], 'LineWidth', 1.2)
plot(x_hat_run, 'c', 'LineWidth', 1.2)
plot(x_hat_exp, 'm', 'LineWidth', 1.2)
grid on; grid minor
xlabel('Steps', 'FontSize', 30)
ylabel('Data', 'FontSize', 30)
legend('Trajectory', 'Measuraments', 'Running Mean', 'Exponential Mean',↙
'FontSize', 25)
% xlim([0 300])

figure(4)
plot(x_n, 'k', 'LineWidth', 1.2)
hold on
plot(x_hat_run, 'b', 'LineWidth', 1.2)
grid on; grid minor
xlabel('Steps', 'FontSize', 30)
ylabel('Data', 'FontSize', 30)
legend('Trajectory', 'Running Mean','Location', 'best', 'FontSize', 25)

figure(5)
plot(x_n, 'k', 'LineWidth', 1.2)
hold on
plot(x_hat_exp, 'r', 'LineWidth', 1.2)
grid on; grid minor
xlabel('Steps', 'FontSize', 30)
ylabel('Data', 'FontSize', 30)
legend('Trajectory', 'Exponential Mean', 'FontSize', 25)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                           %
%                                 FUNCTION                                  %
%                                                                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function [x, x_hat, z, alpha, sigma_eta2, sigma_w2] = fun(n, var_w, var_eta, ↙
incond)

rng default

a_1 = sqrt(var_w);
a_2 = sqrt(var_eta);

w = a_1.*randn(n,1);
eta = a_2.*randn(n,1);

x(1) = incond;

for i = 2:n
    x(i) = x(i-1) + w(i); % generated trajectory RWM
end

for i=1:n
    z(i) = x(i) + eta(i); % Generate measurements zi of the process  Xi
end

% 2) Identify  σw2  and  ση2  using  identification  method

E_v_sq_sum = [];
for i = 2:n
    E_v_sq_sum(i-1) = ( w(i) + eta(i) - eta(i-1) )^2;
end

E_v_sq = 1/(n-1) *sum(E_v_sq_sum);

E_rho_sq_sum = [];
for i=3:n
     E_rho_sq_sum(i-2) = ( w(i) + w(i-1) + eta(i) - eta(i-2) )^2;
end

E_rho_sq = 1/(n-2) *sum(E_rho_sq_sum);

% A = sigma_w^2
% B = sigma_eta^2
syms A B

eqns = [ A -  E_v_sq + 2*B == 0,...
          2*B + 2*A - E_rho_sq == 0  ];
vars = [A B];

[a, b] = solve(eqns,vars);
sigma_w2 = double(a);
sigma_eta2 = double(b);

% 3) Determine optimal smoothing coefficient in exponential smoothing

csi = sigma_w2/sigma_eta2;
```

```
alpha = (-csi + sqrt(csi^2 + 4*csi))/2; % correct bc should be between 0,1

% 4) Perform exponential smoothing with the determined smoothing coefficient

x_hat(1) = 10;

for i = 2:n
    x_hat(i) = alpha*z(i) + (1-alpha)*x_hat(i-1);
end
end
```