```
%% Determining and removing drawbacks of exponential and running mean

% Written by Irina Yareshko and Luca Breggion, Skoltech 2022

close all; clear; clc;

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

%% Backward exponential smoothing

rng default

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 zi of the process Xi
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

% Window size M

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

% 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

% Forward Exponential Estimates
x_hat_forw(1) = incond;

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

% Apply Backward Smoothing
x_hat_back(n_3) = x_hat_forw(end);

for i = (n_3 - 1):-1:1
    x_hat_back(i) = x_hat_back(i + 1) + alpha_n*(x_hat_forw(i) - x_hat_back(i +
1));
end

figure(1)
hold on
plot(x_n, 'k', 'LineWidth', 1.2)
plot(z_n, 'g', 'LineWidth', 1.2)
plot(x_hat_run, 'b', 'LineWidth', 1.2)
plot(x_hat_back, 'r', 'LineWidth', 1.2)
grid on; grid minor
xlabel('Steps', 'FontSize', 30)
ylabel('Data', 'FontSize', 30)
legend('Trajectory', 'Measuraments', 'Running Mean', 'Backward Exponential
Smoothing', 'FontSize', 30)

%% Calculate the indicators

% Deviation Indicators

dev_ind_back = [];
dev_ind_run = [];

for i = 1:n_3

    dev_ind_back(i) = (z_n(i) - x_hat_back(i))^2;
    dev_ind_run(i) = (z_n(i) - x_hat_run(i))^2;

end

dev_ind_back = sum(dev_ind_back);
dev_ind_run = sum(dev_ind_run);

% Variablility Indicators

var_ind_back = [];
var_ind_run = [];

for i = 1:(n_3 - 2)
```

```
    var_ind_back(i) = (x_hat_back(i + 2) - 2*x_hat_back(i + 1) + x_hat_back(i))^2;
    var_ind_run(i) = (x_hat_run(i + 2) - 2*x_hat_run(i + 1) + x_hat_run(i))^2;

end

var_ind_back = sum(var_ind_back);
var_ind_run = sum(var_ind_run);

%% Part 2. Drawbacks of running mean

%% 1. Generate a true trajectory Xi of an object motion disturbed by normally...
% distributed random acceleration

% n = n_3 = 300

x(1) = 5;
v(1) = 0;
t = 0.1;

sigma_a2 = 10; % variance of noise
sigma_eta2 = 500;

a = sqrt(sigma_a2).*randn(n_3,1);
eta = sqrt(sigma_eta2).*randn(n_3,1);

for i = 2:n_3
    v(i) = v(i - 1) + a(i - 1)*t;
    x(i) = x(i - 1) + v(i - 1)*t + (a(i - 1)*t^2) * 0.5;
end

z = [];

for i = 1:n_3
    z(i) = x(i) + eta(i);
end

% Running mean
M_guess = [25, 50, 100, 150, 200, 250];

for k = 1:length(M_guess)
      x_hat_run_n(k,:) = movmean(z, M_guess(k));
end

% Forward mean
alpha_guess = [0.01, 0.02, 0.075, 0.1, 0.15, 0.2];
x_hat_forw_n = zeros(length(alpha_guess), n_3);
x_hat_forw_n(:,1) = z(1);

for j = 1:length(alpha_guess)
    for i = 2:n_3
        x_hat_forw_n(j,i) = x_hat_forw_n(j,i - 1) + alpha_guess(j)*(z(i) -
x_hat_forw_n(j,i - 1));
    end
```

```
end

% Variability indicators for different alpha and M
var_ind_forw_n = zeros(length(alpha_guess), n_3);
var_ind_run_n = zeros(length(alpha_guess), n_3);

var_ind_forw_sum = [];
var_ind_run_sum = [];

for j = 1:length(alpha_guess)
    for i = 1:(n_3 - 2)
        var_ind_forw_n(j,i) = (x_hat_forw_n(j,i + 2) - 2*x_hat_forw_n(j,i + 1) +
x_hat_forw_n(j,i))^2;
        var_ind_run_n(j,i) = (x_hat_run_n(j,i + 2) - 2*x_hat_run_n(j,i + 1) +
x_hat_run_n(j,i))^2;
    end
    var_ind_forw_sum(j) = sum(var_ind_forw_n(j,:));
    var_ind_run_sum(j) = sum(var_ind_run_n(j,:));
end

% Deviation indicators for different alpha and M
dev_ind_forw_n = zeros(length(alpha_guess), n_3);
dev_ind_run_n = zeros(length(alpha_guess), n_3);

dev_ind_forw_sum = [];
dev_ind_run_sum = [];

for j = 1:length(alpha_guess)
    for i = 1:n_3
        dev_ind_forw_n(j,i) = (z(i) - x_hat_forw_n(j,i))^2;
        dev_ind_run_n(j,i) = (z(i) - x_hat_run_n(j,i))^2;
    end
    dev_ind_forw_sum(j) = sum(dev_ind_forw_n(j,:));
    dev_ind_run_sum(j) = sum(dev_ind_run_n(j,:));
end

% we are taking into account the highest M (5th line)

figure(2)
plot(x, 'r', 'LineWidth', 1.2)
hold on
plot(z, 'k', 'LineWidth', 1.2)
plot(x_hat_run_n(5,:), 'c', 'LineWidth', 1.2)
grid on; grid minor
xlabel('Steps', 'FontSize', 30)
ylabel('Data', 'FontSize', 30)
legend('Trajectory', 'Measurements', 'Running Mean', 'FontSize', 30)

figure(3)
plot(x, 'r', 'LineWidth', 1.2)
hold on
plot(z, 'k', 'LineWidth', 1.2)
plot(x_hat_forw_n(1,:), 'c', 'LineWidth', 1.2)
```

```
grid on; grid minor
xlabel('Steps', 'FontSize', 30)
ylabel('Data', 'FontSize', 30)
legend('Trajectory', 'Measurements', 'Exponential Mean', 'FontSize', 30)

%% 4) Second trajectory: Generate cyclic trajectory Xi according to the equation
% 5) Generate measurements zi of the process Xi
%6) Apply running mean with window size M=13 to measurements zi.

clear sigma_eta2
clear eta

a = 1;
n_4 = 200;
T = 32;
sigma_w2= 0.08^2;
sigma_eta2= 0.05;
M_4 = 13;

[x_sin, z_4, x_hat_run_4] = t_fun(T,sigma_w2, sigma_eta2, a, n_4, M_4);

% plot of the results
figure(4)
plot(x_sin, 'r', 'LineWidth', 1.2)
hold on
plot(z_4, 'k', 'LineWidth', 1.2)
plot(x_hat_run_4, 'c', 'LineWidth', 1.2)
grid on; grid minor
xlabel('Steps', 'FontSize', 30)
ylabel('Data', 'FontSize', 30)
legend('Trajectory', 'Measurements', 'Running Mean', 'FontSize', 30)

%% Determine the period of oscillations for which running mean with given
% for every group window size M

M_new = 19;
T_new = 15; % produces inverse oscillations
%T_new = 19; % leads to the loss of oscillations (zero oscillations)
%T_new = 25; % changes the oscillations insignificantly
[x_sin, z_4, x_hat_run_4] = t_fun(T_new,sigma_w2, sigma_eta2, a, n_4, M_new);

% plot
figure(5)
plot(x_sin, 'r', 'LineWidth', 1.2)
hold on
plot(z_4, 'k', 'LineWidth', 1.2)
plot(x_hat_run_4(1,:), 'c', 'LineWidth', 1.2)
grid on; grid minor
xlabel('Steps', 'FontSize', 30)
ylabel('Data', 'FontSize', 30)
legend('Trajectory', 'Measurements', 'Running Mean 15', 'FontSize', 30)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

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

function [x_sin, z_4, x_hat_run_4] = t_fun(T,sigma_w2, sigma_eta2, a, n_4, M_4)

omega = 2*pi/T;
w = sqrt(sigma_w2).*randn(n_4,1);
A(1) = a;

for i = 2:n_4
    A(i) = A(i-1) + w(i);
end

x_sin = [];
for i = 1:n_4
    x_sin(i) = A(i) * sin(omega*i + 3);
end

eta = sqrt(sigma_eta2).*randn(n_4,1);

z_4 = [];
for i = 1:n_4
    z_4(i) = x_sin(i) + eta(i);
end

x_hat_run_4 = movmean(z_4, M_4);

end
```