```
%% 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');

%% I Part: Comparison of the traditional 13-month running mean with the forward-
%backward exponential smoothing for approximation of 11-year sunspot cycle

%% Download monthly mean sunspot number

data = load('data_group4.mat');
years = data.data(:,1); % year
m = data.data(:,2); % month
m_sun =  data.data(:,3); % monthly sunspot number
n = length(years);

%% 2) Make smoothing of monthly mean data by 13-month running mean

m_sun_mean = zeros(length(m_sun),1);
m_sun_mean(1:6) = 1/6*sum(m_sun(1:6));

m_sun_mean((length(m_sun_mean) - 5):length(m_sun_mean)) = 1/6 * sum(m_sun(length
(m_sun)-5:length(m_sun)));

for i = 7:(n - 6)
    m_sun_mean(i) = 1/24*(m_sun(i-6) + m_sun(i+6)) + 1/12*(m_sun(i-5) + m_sun(i-4)
+ m_sun(i-3) + m_sun(i-2) + m_sun(i-1) + ...
        m_sun(i) + m_sun(i+5) + m_sun(i+4) + m_sun(i+3) + m_sun(i+2) + m_sun(i+1));
end

 dev_13_step = sum((m_sun-m_sun_mean).^2);

 for i = 1:(n - 2)
    var_13_step(i) = (m_sun_mean(i + 2) - 2*m_sun_mean(i + 1) + m_sun_mean(i))^2;
 end
 var_13_step = sum(var_13_step);

%% 3) Make forward-backward exponential smoothing of monthly mean sunspot number.
% Is there a smoothing constant α that provides better results compared to
% 13-month running mean according to deviation and variability indicators?

alpha = [0.05, 0.1, 0.15, 0.2, 0.25, 0.3];
x_forw = zeros(length(alpha), n);
x_forw(:,1) = m_sun(1);

x_back = zeros(length(alpha), n);
x_back(:,n) = x_forw(end);
```

```
for j = 1:length(alpha)
    for i = 2:n
        x_forw(j,i) = x_forw(j,i - 1) + alpha(j)*(m_sun(i) - x_forw(j,i - 1));
    end
    for i = (n - 1):-1:1
        x_back(j,i) = x_back(j,i + 1) + alpha(j)*(x_forw(j,i) - x_back(j,i + 1));
    end
end

% Variability indicators for different alpha
var_forw_sum = [];

for j = 1:length(alpha)
    for i = 1:(n - 2)
        var_forw(j,i) = (x_forw(j,i + 2) - 2*x_forw(j,i + 1) + x_forw(j,i))^2;
    end
    var_forw_sum(j) = sum(var_forw(j,:));
end

% Deviation indicators for different alpha
dev_forw = zeros(length(alpha), n);
dev_forw_sum = [];

for j = 1:length(alpha)
    for i = 1:n
        dev_forw(j,i) = (m_sun(i) - x_forw(j,i))^2;
    end
    dev_forw_sum(j) = sum(dev_forw(j,:));
end

figure(1)
plot(m_sun, 'c', 'LineWidth', 1.2)
hold on
plot(m_sun_mean, 'k', 'LineWidth', 1.2)
%plot(x_back(1,:), 'LineWidth', 1.2)
%plot(x_back(2,:), 'LineWidth', 1.2)
%plot(x_back(3,:), 'LineWidth', 1.2)
plot(x_back(4,:),'m', 'LineWidth', 1.2)
%plot(x_back(5,:), 'LineWidth', 1.2)
%plot(x_back(6,:), 'LineWidth', 1.2)
grid on; grid minor
xlabel('Month cycle number', 'FontSize', 30)
ylabel('Monthly sunspot number', 'FontSize', 30)
legend('Measurements', '13-month Running mean', 'FB exponential with $\alpha$ =
0.2', 'FontSize', 30, 'interpreter', 'latex')

%% II Part: 3d surface filtration using forward-backward smoothing
%% 1-2) Download surface data and Plot noisy and true surface for visualization
purposes

tr = load('true_surface.mat');
no = load('noisy_surface.mat');
```

```
true = tr.true_surface;
noise = no.noisy_surface;
x = [1:1:length(true)];

figure(2)
mesh(x,x,true)
colormap jet
xlabel('Y', 'FontSize', 30)
ylabel('X','FontSize', 30)
zlabel('Z', 'FontSize', 30)

figure(3)
mesh(x,x,noise)
colormap jet
xlabel('Y','FontSize', 30)
ylabel('X','FontSize', 30)
zlabel('Z','FontSize', 30)

%% 3) Determine the variance of deviation of noisy surface from the true one.
% Hint: You may reshape the matrix (difference between the noisy and true surface)
% into one array ("reshape command") and then determine the variance of obtained
array.

var = sum((reshape((true-noise),...
    [1,length(true)*length(true)])).^2)/(length(true)*length(true)-1);

% Diff_square_matrix=(noise-true).^2;
% [row, col] = size(Diff_square_matrix);
% Diff_square_list=reshape(Diff_square_matrix,[row*col,1]);
% Deviation_for_noisy = (1/(row*col-1))*sum(Diff_square_list)

%% 4) Apply forward-backward exponential smoothing to filter noisy surface
measurements.
% 5) Compare visually the obtained estimation results and true surface.
% 6) Determine the variance of deviation of smoothed surface from the true one.

alpha = 0.335;

[smoothed, var_smoothed] = forward_backward(true, noise, alpha);

figure(4)
mesh(x,x,smoothed)
colormap jet
xlabel('Y','FontSize', 30)
ylabel('X','FontSize', 30)
zlabel('Z','FontSize', 30)

%% 7) Try greater and smaller values of smoothing coefficient α and explain
% the affect on estimation results

alpha = [0.2:0.05:0.8];
var_s = zeros(1, length(alpha));
```

```
for i=1:length(alpha)
    [~, var_s(i)] = forward_backward(true, noise, alpha(i));
end

figure(5)
plot(alpha, var_s, 'ko-','LineWidth', 1.5)
grid on; grid minor
xlabel('Alpha', 'FontSize',30)
ylabel('Variance', 'FontSize',30)
xlim([0.2 0.8])

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                            %
%                                  FUNCTION                                  %
%                                                                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [smoothed, var] = forward_backward(true, noise, alpha)

%Step 1. Forward raws

raw_forw = zeros(length(true), length(true));
raw_forw(1, :) = noise(1, :);

for i = 2:length(true)
    raw_forw(i, :) = raw_forw(i - 1, :) + alpha*(noise(i, :) - raw_forw(i - 1, :));
end
%Step 2. Backward raws
raw_back = zeros(length(true), length(true));
raw_back(end, :) = raw_forw(end, :);

for i = (length(true) - 1):-1:1
    raw_back(i, :) = raw_back(i + 1, :) + alpha*(raw_forw(i, :) - raw_back(i + 1, ↙
:));
end
%Step 3. Forward columns
col_forw = zeros(length(true), length(true));
col_forw(:,1) = raw_back(:,1);

for i = 2:length(true)
    col_forw(:, i) = col_forw(:,i - 1) + alpha*(raw_back(:, i) - col_forw(:,i - ↙
1));
end

%Step 4. Backwards columns

col_back = zeros(length(true), length(true));
col_back(:, end) = col_forw(:, end);

for i = (length(true) - 1):-1:1
    col_back(:, i) = col_back(:, i + 1) + alpha*(col_forw(:, i) - col_back(:,i + ↙
1));
end
```

```
smoothed = col_back;

var = sum((reshape((true-smoothed),...
    [1,length(true)*length(true)])).^2)/(length(true)*length(true)-1);
end
```