```
%{
Assignment 4
Determining and removing drawbacks of exponential and running mean. Task 2
by
Group (19)
Ahmed Baza , skoltech, 2020
Ahmed Elgohary , skoltech, 2020
Moahmed Sayed , skoltech, 2020
Ziad Baraka , skoltech, 2020
%}

clc
clear
load('data_group5.mat'); % loading data.
Years_axis=data(:,1)+(1/12)*data(:,2);
R=data(:,3); % sunspot number
N=length(R);
RS=R;                       % arrays for the smoothing results.
for i=7:N-6  %running mean smoothing with window size= 13
  RS(i)=1/24*(R(i-6)+R(i+6))+1/12*sum(R(i-5:i+5));
end
RS(1:6)=mean(R(1:6)); RS(N-5:N)=mean(R(N-5:N));
figure()
plot(Years_axis,R,Years_axis,RS);
title('Epermental data vs Smoothed data');
legend('Epermental data','Smoothed data')
xlabel('Years')
ylabel('Number of sun Spots');

Id_RM=sum((R-RS).^2); %deviation Indecator of running mean smoothing
Iv_RM=varI(RS); % Variabelity indecator of running mean smoothing

k=1;
for alpha=0:0.01:0.9
                        %Forward Exponintial Smoothing
        RF=R;
    for i=2:N
        RF(i)=alpha*R(i)+(1-alpha)*RF(i-1);
    end
                        % Backward Exponintial Smoothing
    RB=RF;
    for i=N-1:-1:1
        RB(i)=alpha*RF(i)+(1-alpha)*RB(i+1);
    end
    Id_BS=sum((R-RB).^2); %deviation Indecator of forward-backward smoothing.
    Iv_BS=varI(RB); % Variabelity indecator of forward-backward smoothing.
    if Id_BS<Id_RM && Iv_BS<Iv_RM
        alphaM(k)=alpha;
        k=k+1;
    end
end
```

```
disp(alphaM);

figure()
for j=1:length(alphaM)
    alpha=alphaM(j);
      RF=R;
                        % forward Exponitial smothing
    for i=2:N
        RF(i)=alpha*R(i)+(1-alpha)*RF(i-1);
    end
                        % Backward Exponintial Smoothing
    RB=RF;
    for i=N-1:-1:1
        RB(i)=alpha*RF(i)+(1-alpha)*RB(i+1);
    end
    subplot(3,3,j)
    plot(Years_axis,R,Years_axis,RS,'r',Years_axis,RB,'g');
    s='M=13 vs alpha=';
    s2=sprintf('%.2f',alpha);
    s3=strcat(s,s2);
    title(s3);
    xlabel('Years')
    ylabel('sunSpots Num');
    xlim([1845 1860])
end
subplot(3,3,9)
plot(0,0,0,0,'r',0,0,'g')
axis off
legend('Epermental data','runnig mean','Forward-Backward')

load('noisy_surface.mat');
load('true_surface.mat');
figure()
mesh(true_surface) % plotting true surface.
colormap jet
colorbar
title('True Surface');  xlabel('X'); ylabel('Y'); zlabel('Z')

mesh(noisy_surface)
colormap jet
colorbar
title('Noisy Surface'); xlabel('X'); ylabel('Y'); zlabel('Z');

True=reshape(true_surface,[],1);
Noisy=reshape(noisy_surface,[],1);
Varianc=(1/(length(True)-1))*sum((Noisy-True).^2)

S=noisy_surface;
SFR=S;       %forward smoothing vector
N=length(S);
alpha=.335;
```

```
% forward smothing of raws
for i=1:N
    for j=2:N
       SFR(i,j)=alpha*S(i,j)+(1-alpha)*SFR(i,j-1);
    end
end
% backward Smothing of raws.
SBR=SFR;
for i=1:N
    for j=N-1:-1:1
       SBR(i,j)=alpha*SFR(i,j)+(1-alpha)*SBR(i,j+1);
    end
end
% forward smothing of columns.
SFC=SBR;
for j=1:N
    for i=2:N
         SFC(i,j)=alpha*SBR(i,j)+(1-alpha)*SFC(i-1,j);
    end
end
%  backward smothing of columns.
SBC=SFC;

for j=1:N
    for i=N-1:-1:1
         SBC(i,j)=alpha*SFC(i,j)+(1-alpha)*SBC(i+1,j);
    end
end

figure()
mesh(SBC)
colormap jet
colorbar
title('Reconstructed Surface')
xlabel('X')
ylabel('Y')
zlabel('Z')

Reconstructed=reshape(SBC,[],1);
Varianc2=(1/(length(Reconstructed)-1))*sum((Reconstructed-True).^2)

load('noisy_surface.mat')
load('true_surface.mat')

noisy=noisy_surface;
true=true_surface;

[x,y]=size(true);
Z=noisy;
k=0;
for alpha=0.1:0.1:0.9
```

```
    k=k+1;
    Zs1=Z;
    for j=1:y % forward smothing of raws
        for i=2:x
            Zs1(i,j)=Zs1(i-1,j)+alpha*(Z(i,j)-Zs1(i-1,j));
        end
    end
    Zs2=Zs1;

    for j=1:y % backward smothing of raws
        for i=x-1:-1:1
            Zs2(i,j)=Zs2(i+1,j)+alpha*(Zs1(i,j)-Zs2(i+1,j));
        end
    end

    Zs3=Zs2;

    for i=1:x % forward smothing of Columns
        for j=2:y
            Zs3(i,j)=Zs3(i,j-1)+alpha*(Zs2(i,j)-Zs3(i,j-1));
        end
    end

    Zs4=Zs3;

    for i=1:x % Backward smothing of Columns
        for j=y-1:-1:1
            Zs4(i,j)=Zs4(i,j+1)+alpha*(Zs3(i,j)-Zs4(i,j+1));
        end
    end

    subplot(3,3,k)
    mesh(Zs4)
    colormap jet;
    colorbar
    s= 'alpha=';
    s2=sprintf('%.2f',alpha);
    s3=strcat(s,s2);
    title(s3);
    xlabel('X'); ylabel('Y'); zlabel('Z');
end

%% Learning log code

load('noisy_surface.mat')
load('true_surface.mat')

noisy=noisy_surface;
true=true_surface;

figure()
```

```
mesh(true)
colormap jet
[x,y]=size(true);
N=x*y;
Z=noisy;
Zs1=Z;
alpha=0.335;
for j=1:y % forward smothing of Raws
    for i=2:x
        Zs1(i,j)=Zs1(i-1,j)+alpha*(Z(i,j)-Zs1(i-1,j));
    end
end
Zs3=Zs1;

for i=1:x % forward smothing of Columns
    for j=2:y
        Zs3(i,j)=Zs3(i,j-1)+alpha*(Zs1(i,j)-Zs3(i,j-1));
    end
end

hold on
mesh(Zs3)
colorbar
title('Forward smoothing vs. True surface')
xlabel('X'); ylabel('Y'); zlabel('Z');
view([-6 24])

VarianceMatrixS=(Zs3-true).^2;

VarianceS=(1/(N-1))*sum(reshape(VarianceMatrixS,[],1)) % variance of the forward↙
smoothing in raws and columns

function I = varI(s) % Function to calculate the variability indecator.
   n=length(s);
   d=1:n-2;
   for j=1:n-2
       d(j)=s(j+2)+s(j)-2*s(j+1);
   end
   I=sum(d.^2);
end
```