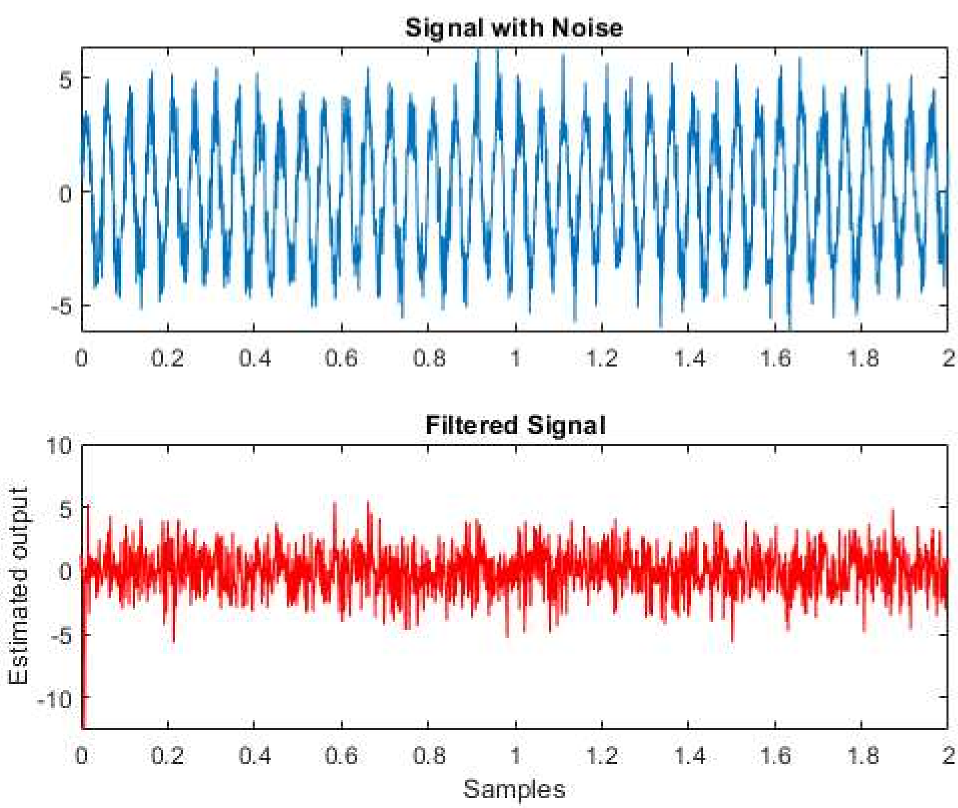
Faranak Dayyani - student number: 1002373674

Contents

Question 1**:

```
N = 2000;
fs = 1000;
[waveform_noise, time, waveform, snr_out] = sig_noise(20, 8, N);

Ts = 1/fs;
t = (0:N-1)*Ts;

%channel system order
sysorder = 2 ;

inp = randn(N,1);
n = randn(N,1);
[b,a] = butter(2,0.25);
Gz = tf(b,a,-1);

x = waveform;
y = waveform_noise;
totallength=(length(y));
%Take 60 points for training
N=60 ;

d = waveform;
%begin of algorithm
w = zeros ( sysorder  , 1 ) ;
for n = sysorder : N
        u = inp(n:-1:n-sysorder+1) ;
```

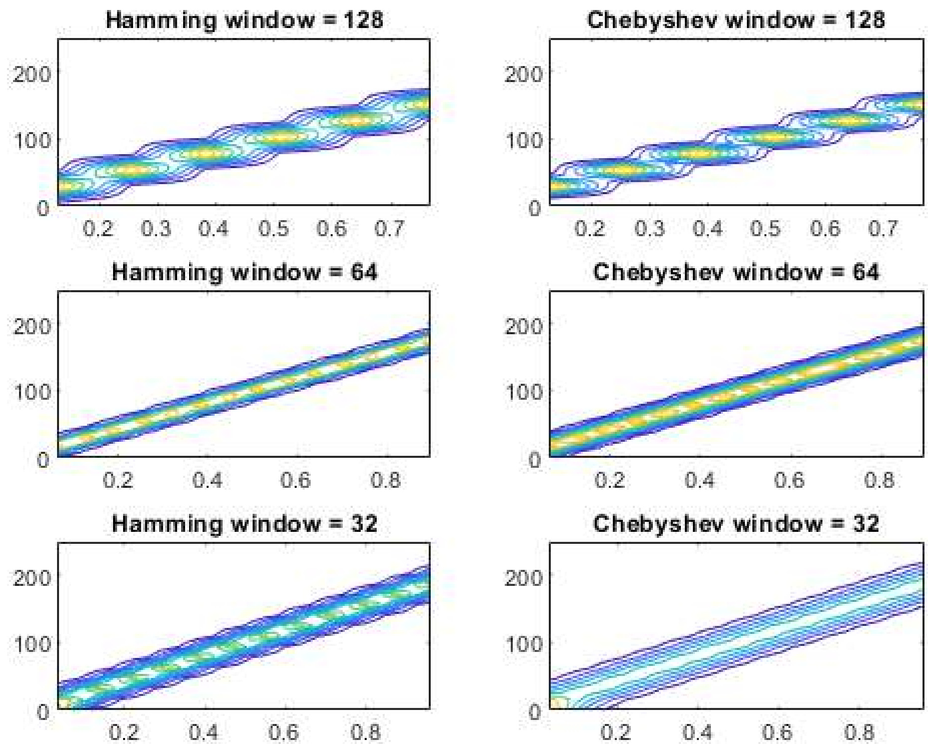
```
    y(n)= w' * u;
    e(n) = d(n) - y(n) ;
% Start with big mu for speeding the convergence then slow down to reach the correct weights
    if n < 20
        mu=0.32;
    else
        mu=0.15;
    end
        w = w + mu * u * e(n) ;
end

%check of results
for n =  N+1 : totallength
        u = inp(n:-1:n-sysorder+1) ;
    y(n) = w' * u ;
    e(n) = d(n) - y(n) ;
end

figure;
subplot(2,1,1);
plot(time, waveform_noise); title('Signal with Noise');
subplot(2,1,2);
plot(time, y,'r');
title('Filtered Signal') ;
xlabel('Samples');
ylabel('Estimated output');
```



Question 2**:

analyze a chirp signal using STFT

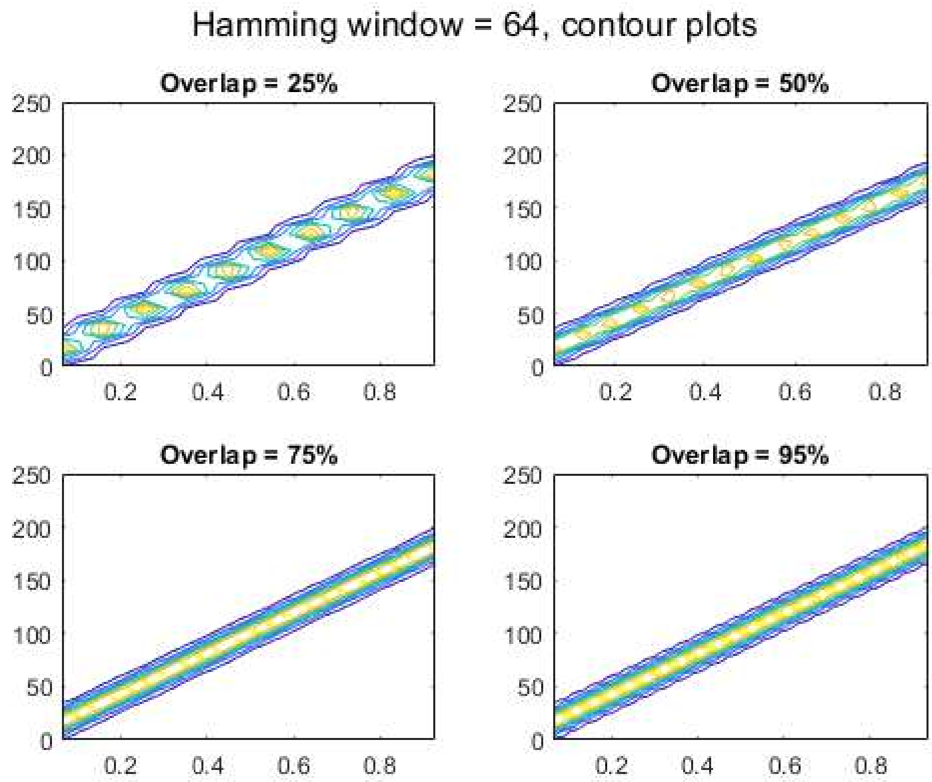
```
fs = 500;
Tt = 1;
N = 500;

Ts = 1/fs;
t = (0:(N-1))*Ts;

% frequency vector => varies from 10 to 200
freq = linspace(10, 200, N);
fc = fs;
% time vector => varies from 0 to 1.0
time = linspace(0, 1, N);

x = sin(pi*time.*freq);   %chirp signal

window_size_1 = 128;
window_size_2 = 64;
window_size_3 = 32;

% for hamming windows
ham_1 = hamming(window_size_1);
ham_2 = hamming(window_size_2);
ham_3 = hamming(window_size_3);

figure;
subplot(3,2,1);
[s1, f1, t1] = spectrogram(x, ham_1, [], window_size_1, fs);
contour(t1, f1, abs(s1)); title('Hamming window = 128');

[s2, f2, t2] = spectrogram(x, ham_2, [], window_size_2, fs);
subplot(3,2,3);
contour(t2, f2, abs(s2)); title('Hamming window = 64');

[s3, f3, t3] = spectrogram(x, ham_3, [], window_size_3, fs);
subplot(3,2,5);
contour(t3, f3, abs(s3)); title('Hamming window = 32');

% for chebyshev windows
cheb_1 = chebwin(window_size_1);
cheb_2 = chebwin(window_size_2);
cheb_3 = chebwin(window_size_3);

[s4, f4, t4] = spectrogram(x, cheb_1, [], window_size_1, fs);
subplot(3,2,2);
contour(t4, f4, abs(s4)); title('Chebyshev window = 128');

[s5, f5, t5] = spectrogram(x, cheb_2, [], window_size_2, fs);
subplot(3,2,4);
contour(t5, f5, abs(s5)); title('Chebyshev window = 64');

[s6, f6, t6] = spectrogram(x, cheb_3, [], window_size_3, fs);
subplot(3,2,6);
contour(t6, f6, abs(s6)); title('Chebyshev window = 32');

% the window size = 64 produces the narrowest time frequency spectrum
% and it has the best time-frequency resolution trade off.
% The type of window shape does make a difference. This can be seen by
% looking at the width of each plot.
```

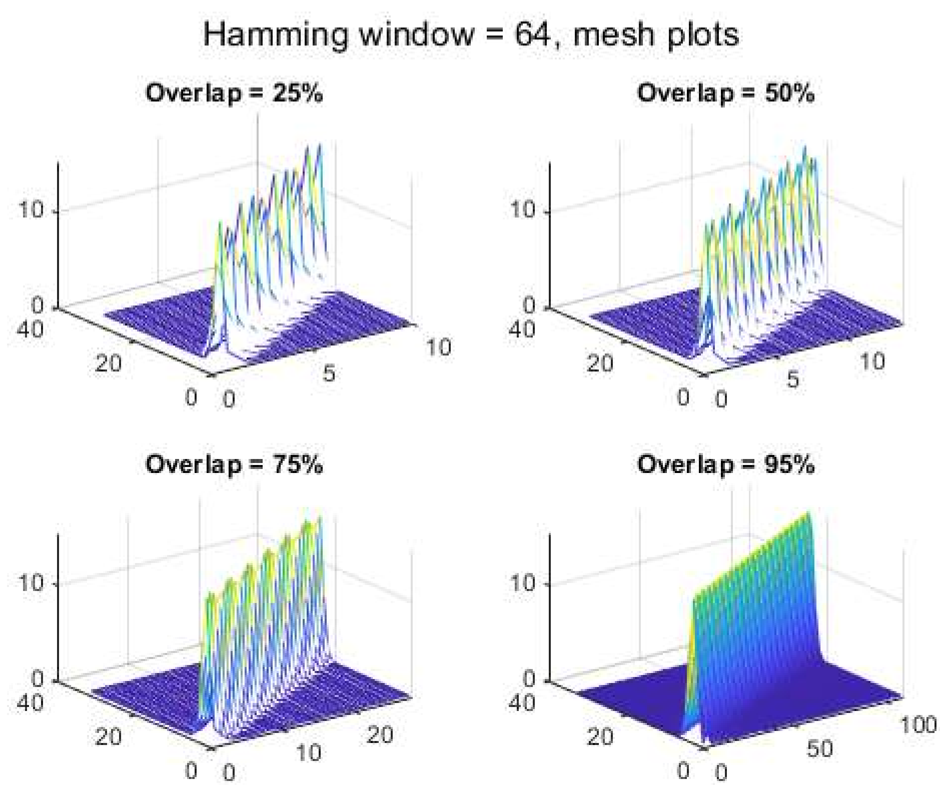


Question 3:

```
fs = 500;
Tt = 1;
N = 500;

Ts = 1/fs;
t = (0:(N-1))*Ts;

% frequency vector => varies from 10 to 200
freq = linspace(10, 200, N);
fc = fs;
% time vector => varies from 0 to 1.0
time = linspace(0, 1, N);

x = sin(pi*time.*freq);   %chirp signal

window_size = 64;
ham = hamming(window_size);

% window overlaps = 25%, 50%, 75% and 95%
% ------ overlap = 25% ------------
nov_25 = floor(window_size*0.25);
[s_25, f_25, t_25] = spectrogram(x, ham, nov_25, window_size, fs);

% ------ overlap = 50% ------------
nov_50 = floor(window_size*0.5);
[s_50, f_50, t_50] = spectrogram(x, ham, nov_50, window_size, fs);

% ------ overlap = 75% ------------
nov_75 = floor(window_size*0.75);
[s_75, f_75, t_75] = spectrogram(x, ham, nov_75, window_size, fs);
```

```matlab
% ------ overlap = 95% ------------
nov_95 = floor(window_size*0.95);
[s_95, f_95, t_95] = spectrogram(x, ham, nov_95, window_size, fs);

% use contour plots to display results
figure;
sgtitle('Hamming window = 64, contour plots');
subplot(2,2,1);
contour(t_25, f_25, abs(s_25)); title('Overlap = 25%');
subplot(2,2,2);
contour(t_50, f_50, abs(s_50)); title('Overlap = 50%');
subplot(2,2,3);
contour(t_75, f_75, abs(s_75)); title('Overlap = 75%');
subplot(2,2,4);
contour(t_95, f_95, abs(s_95)); title('Overlap = 95%');

% repeat the problem using "mesh" 3-D plot
figure;
sgtitle('Hamming window = 64, mesh plots');
subplot(2,2,1);
mesh(abs(s_25)); title('Overlap = 25%');
subplot(2,2,2);
mesh(abs(s_50)); title('Overlap = 50%');
subplot(2,2,3);
mesh(abs(s_75)); title('Overlap = 75%');
subplot(2,2,4);
mesh(abs(s_95)); title('Overlap = 95%');
```

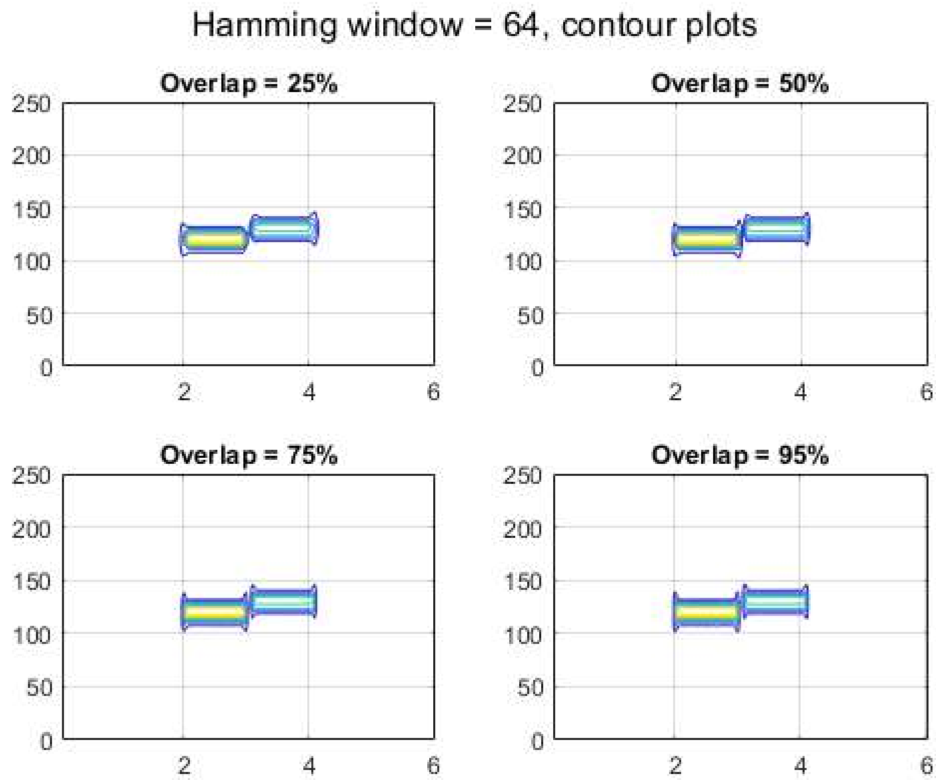




Question 4:

```
%Loading the signal data
s = load('time_freq1.mat');
dataset = s.x;
```

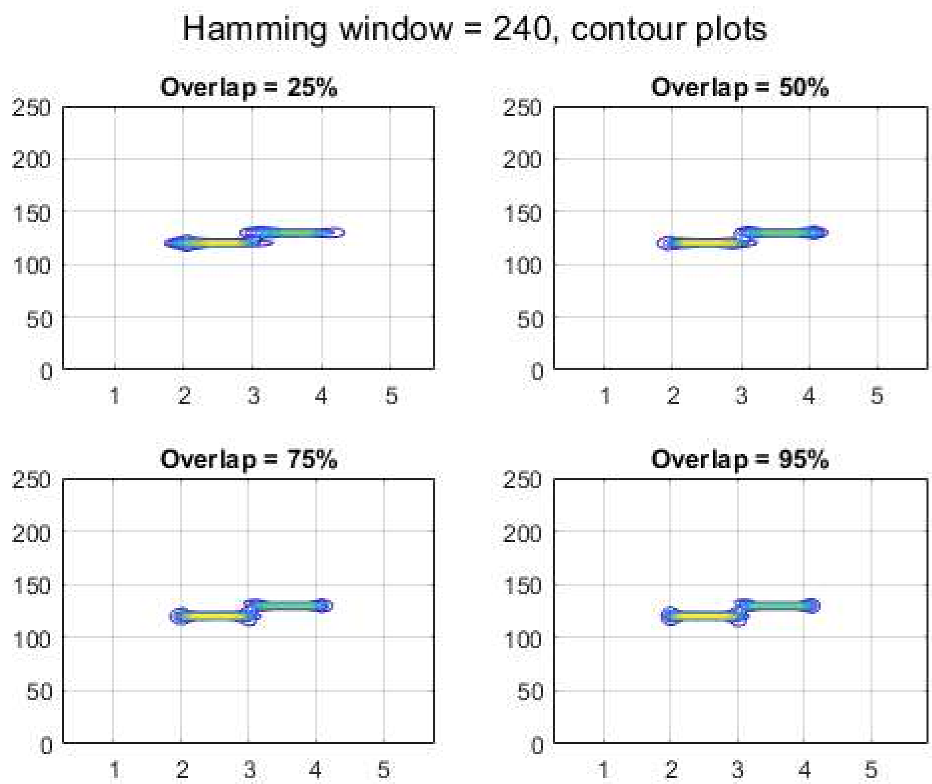
```
fs = 500;

window_size = 64;
N = length(dataset);
Ts = 1/fs;
t = (0:(N-1))*Ts;

% evaluate STFT with overlaps of 25%, 50%, 75% and 95%.
ham = hamming(window_size);

% ------ overlap = 25% ------------
ov_25 = floor(window_size*0.25);
[s_25, f_25, t_25] = spectrogram(dataset, ham, ov_25, window_size, fs);

% ------ overlap = 50% ------------
ov_50 = floor(window_size*0.5);
[s_50, f_50, t_50] = spectrogram(dataset, ham, ov_50, window_size, fs);

% ------ overlap = 75% ------------
ov_75 = floor(window_size*0.75);
[s_75, f_75, t_75] = spectrogram(dataset, ham, ov_75, window_size, fs);

% ------ overlap = 95% ------------
ov_95 = floor(window_size*0.95);
[s_95, f_95, t_95] = spectrogram(dataset, ham, ov_95, window_size, fs);

figure;
sgtitle('Hamming window = 64, contour plots');
subplot(2,2,1);
contour(t_25, f_25, abs(s_25)); title('Overlap = 25%'); grid on;
subplot(2,2,2);
contour(t_50, f_50, abs(s_50)); title('Overlap = 50%'); grid on;
subplot(2,2,3);
contour(t_75, f_75, abs(s_75)); title('Overlap = 75%'); grid on;
subplot(2,2,4);
contour(t_95, f_95, abs(s_95)); title('Overlap = 95%'); grid on;

% After zooming into each plot, it can be seen that the 95% overlap
% produces the best time separation between the two signals
```



Question 5:

```
%Loading the signal data
s = load('time_freq1.mat');
dataset = s.x;

fs = 500;

window_size = 240;
N = length(dataset);
Ts = 1/fs;
t = (0:(N-1))*Ts;

% evaluate STFT with overlaps of 25%, 50%, 75% and 95%.
ham = hamming(window_size);

% ------ overlap = 25% ------------
ov_25 = floor(window_size*0.25);
[s_25, f_25, t_25] = spectrogram(dataset, ham, ov_25, window_size, fs);

% ------ overlap = 50% ------------
ov_50 = floor(window_size*0.5);
[s_50, f_50, t_50] = spectrogram(dataset, ham, ov_50, window_size, fs);

% ------ overlap = 75% ------------
ov_75 = floor(window_size*0.75);
[s_75, f_75, t_75] = spectrogram(dataset, ham, ov_75, window_size, fs);

% ------ overlap = 95% ------------
ov_95 = floor(window_size*0.95);
[s_95, f_95, t_95] = spectrogram(dataset, ham, ov_95, window_size, fs);

figure;
```

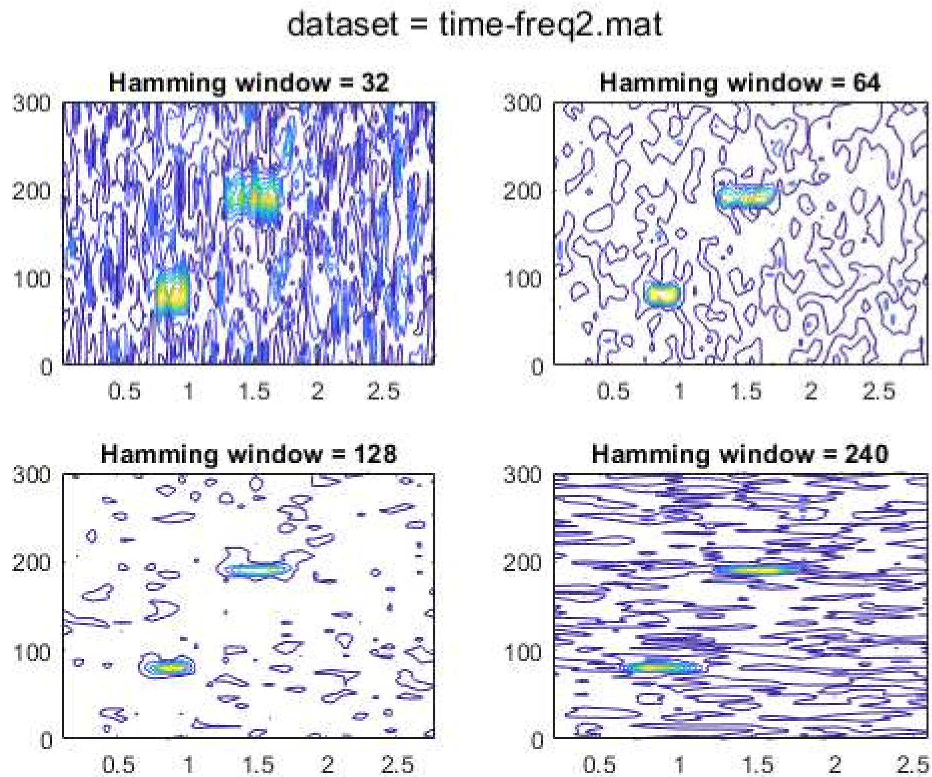
```
sgtitle('Hamming window = 240, contour plots');
subplot(2,2,1);
contour(t_25, f_25, abs(s_25)); title('Overlap = 25%'); grid on;
subplot(2,2,2);
contour(t_50, f_50, abs(s_50)); title('Overlap = 50%'); grid on;
subplot(2,2,3);
contour(t_75, f_75, abs(s_75)); title('Overlap = 75%'); grid on;
subplot(2,2,4);
contour(t_95, f_95, abs(s_95)); title('Overlap = 95%'); grid on;

% After zooming into each plot, it can be seen that the 25% overlap produces
% the best time separation between the two signals.
```



Question 6**:

```
%Loading the signal data
s = load('time_freq2.mat');
d = s.x;

fs = 600;

N = length(d);
Ts = 1/fs;
t = (0:(N-1))*Ts;

win1 = 32; ham1 = hamming(win1); [s1, f1, t1] = spectrogram(d, ham1, [], win1, fs);
win2 = 64; ham2 = hamming(win2); [s2, f2, t2] = spectrogram(d, ham2, [], win2, fs);
win3 = 128; ham3 = hamming(win3); [s3, f3, t3] = spectrogram(d, ham3, [], win3, fs);
win4 = 240; ham4 = hamming(win4); [s4, f4, t4] = spectrogram(d, ham4, [], win4, fs);

figure;
sgtitle('dataset = time-freq2.mat');
```

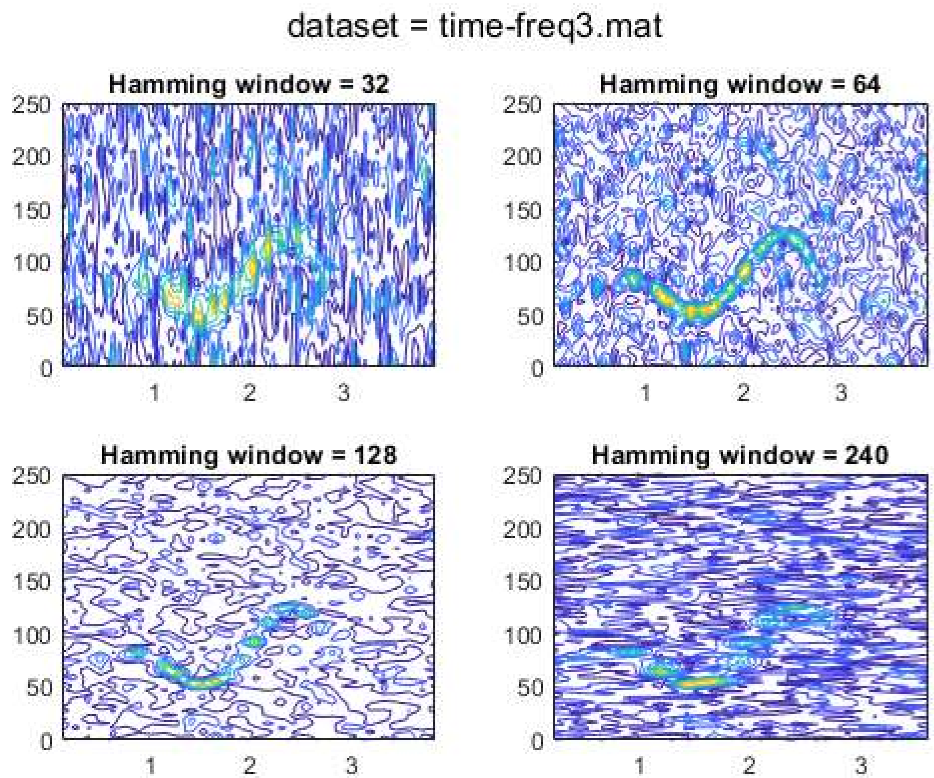
```
subplot(2,2,1); contour(t1, f1, abs(s1)); title('Hamming window = 32');
subplot(2,2,2); contour(t2, f2, abs(s2)); title('Hamming window = 64');
subplot(2,2,3); contour(t3, f3, abs(s3)); title('Hamming window = 128');
subplot(2,2,4); contour(t4, f4, abs(s4)); title('Hamming window = 240');

% the best window size to use for this problem is 128.
```



Question 7:

```
%Loading the signal data
s = load('time_freq3.mat');
d = s.x;

% signal vector x that consists of an unknown signal buried in noise.
fs = 500;
N = 2000;

N = length(d);
Ts = 1/fs;
t = (0:(N-1))*Ts;

win1 = 32; ham1 = hamming(win1); [s1, f1, t1] = spectrogram(d, ham1, [], win1, fs);
win2 = 64; ham2 = hamming(win2); [s2, f2, t2] = spectrogram(d, ham2, [], win2, fs);
win3 = 128; ham3 = hamming(win3); [s3, f3, t3] = spectrogram(d, ham3, [], win3, fs);
win4 = 240; ham4 = hamming(win4); [s4, f4, t4] = spectrogram(d, ham4, [], win4, fs);

figure;
sgtitle('dataset = time-freq3.mat');
subplot(2,2,1); contour(t1, f1, abs(s1)); title('Hamming window = 32');
subplot(2,2,2); contour(t2, f2, abs(s2)); title('Hamming window = 64');
subplot(2,2,3); contour(t3, f3, abs(s3)); title('Hamming window = 128');
subplot(2,2,4); contour(t4, f4, abs(s4)); title('Hamming window = 240');
```

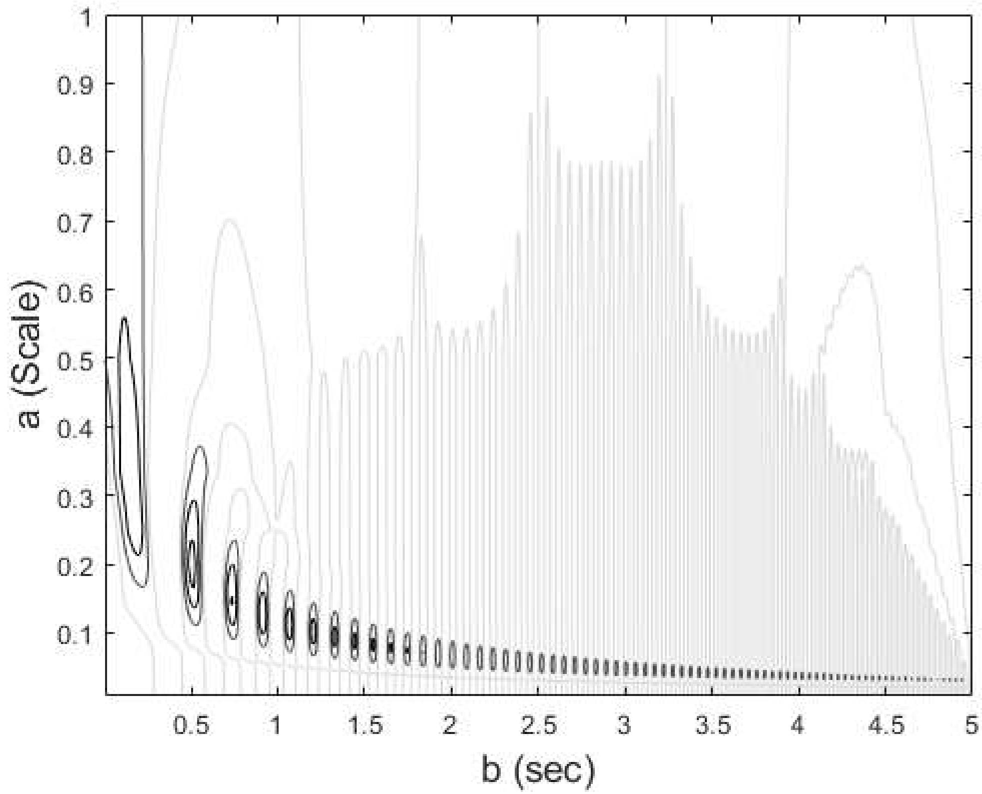
```
% Window size 64 defines the signal best.
```

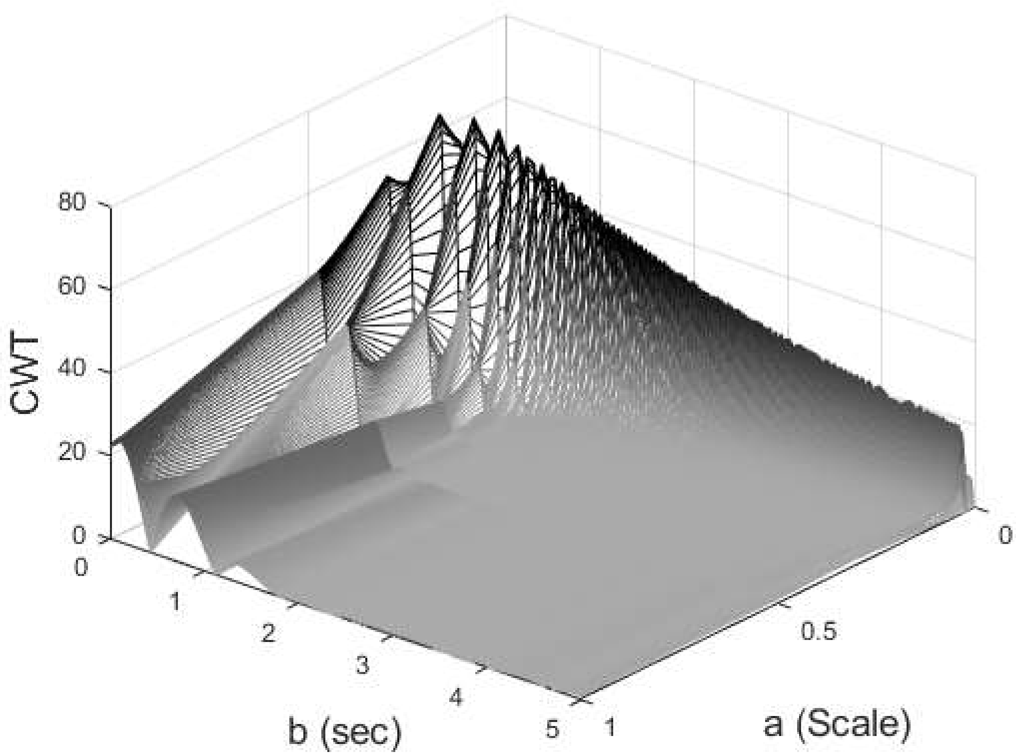


Question 8**:

```
fs = 200; % Sample frequency
N = 1000; % Signal length and half wavelet length
n = N/4; % Signal length divided by 4
resol_level = 120; % Number of levels of a
decr_a = 1; % Decrement for a
a_init = 1; % Initial a
wo = pi * sqrt(2/log2(2));
b = (1:N)/fs; % Time vector for wavelet and plotting

freq = linspace(2, 30, N); % frequency vector => varies from 2 to 30Hz
time = linspace(0, 5, N); % time vector => varies from 0 to 5s
x = sin(pi*time.*freq);   %chirp signal

% Calculate Continuous Wavelet Transform
for k = 1:resol_level
    a(k) = a_init/(k*decr_a); % Set scale
    t = b/a(k); % Time vector for Wavelet
    wav = (exp(-t.^2).* cos(wo*t))/sqrt(a(k)); % Generate Morlet Wavelet
    psi = [fliplr(wav) wav(2:end)]; % Make symmetrical about 0
    wlet(k,:) = psi;
    CW_Trans(:,k) =conv(x,psi,'same'); % Remove extra points from each end
end

figure;
colormap(flipud(gray)); % Invert colormap for better viewing
contour(b,a,CW_Trans'); % Contour plot
ylabel('a (Scale)','FontSize',14);
```

```
xlabel('b (sec)','FontSize',14);
caxis([-5 30]);
figure; colormap(flipud(gray));
mesh(a,b,abs(CW_Trans)); % Plot in 3 dimensions
xlabel('a (Scale)','FontSize',14);
ylabel('b (sec)','FontSize',14);
zlabel('CWT','FontSize',14)
view([130 37]);
caxis([-20 40]);

% in the first figure (contour plot), it can be seen that the resolution
% increases as the frequency increases.
```

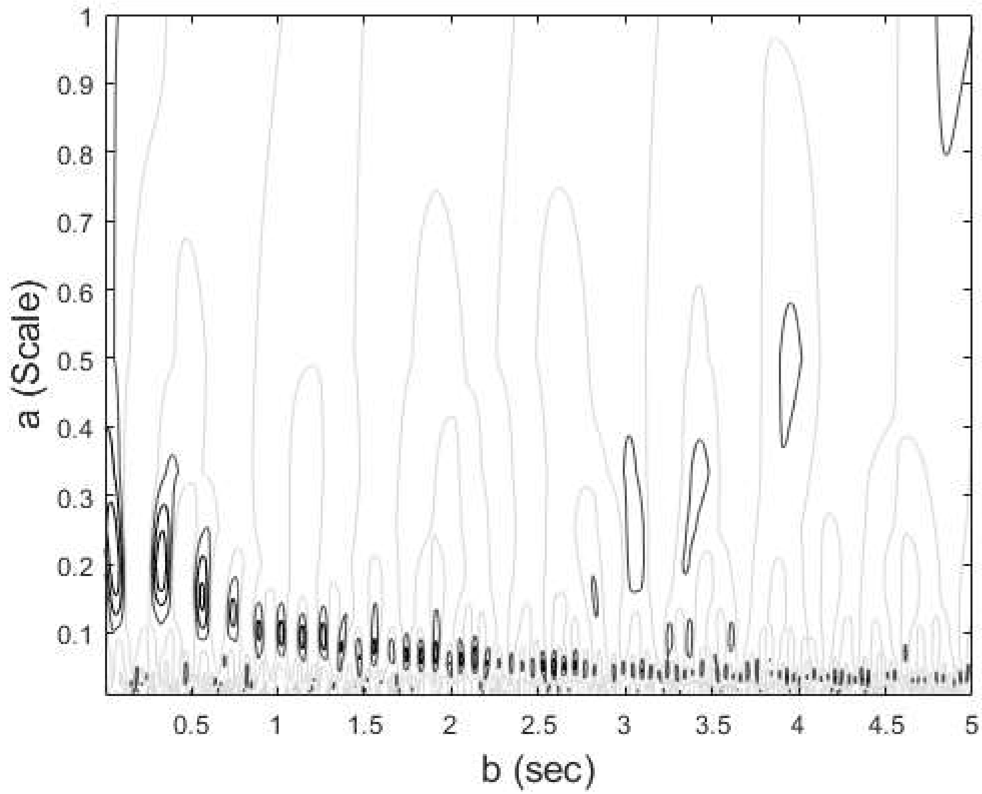


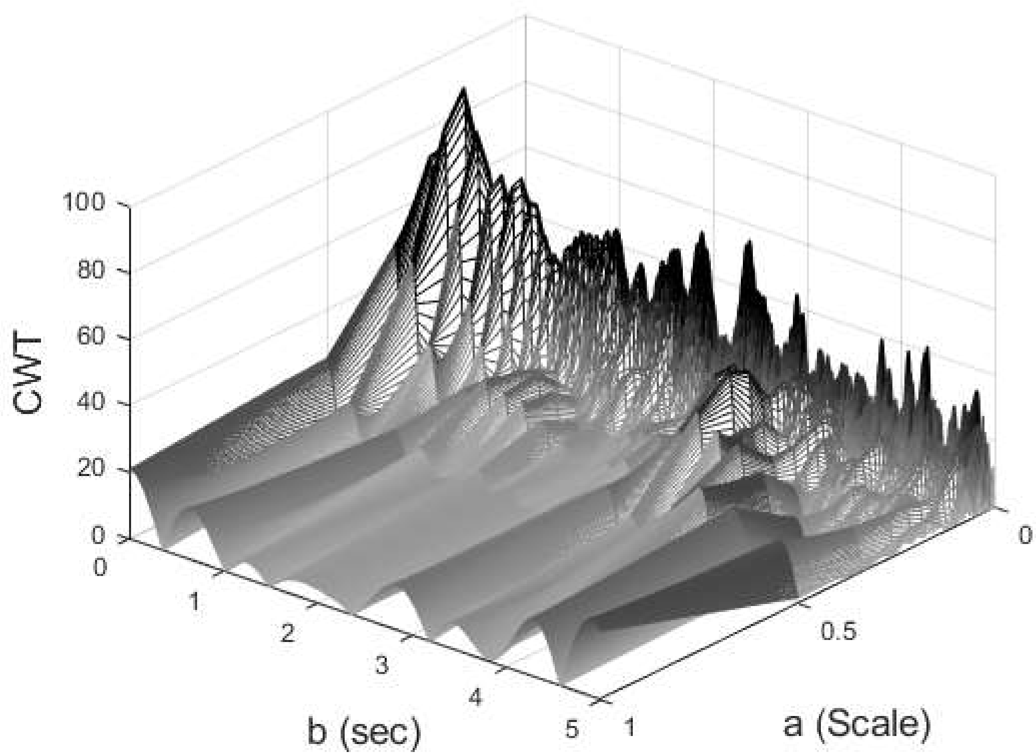


Question 9:

```
%Loading the signal data
s = load('time_freq5.mat');
x = s.x;
```

```
fs = 200; % Sample frequency
N = 1000; % Signal length and half wavelet length
n = N/4; % Signal length divided by 4
resol_level = 120; % Number of levels of a
decr_a = 1; % Decrement for a
a_init = 1; % Initial a
wo = pi * sqrt(2/log2(2));
b = (1:N)/fs; % Time vector for wavelet and plotting

% Calculate Continuous Wavelet Transform
for k = 1:resol_level
    a(k) = a_init/(k*decr_a); % Set scale
    t = b/a(k); % Time vector for Wavelet
    wav = (exp(-t.^2).* cos(wo*t))/sqrt(a(k)); % Generate Morlet Wavelet
    psi = [fliplr(wav) wav(2:end)]; % Make symmetrical about 0
    wlet(k,:) = psi;
    CW_Trans(:,k) =conv(x,psi,'same'); % Remove extra points from each end
end

colormap(flipud(gray)); % Invert colormap for better viewing
contour(b,a,CW_Trans'); % Contour plot
ylabel('a (Scale)','FontSize',14);
xlabel('b (sec)','FontSize',14);
caxis([-5 30]);
figure; colormap(flipud(gray));
mesh(a,b,abs(CW_Trans)); % Plot in 3 dimensions
xlabel('a (Scale)','FontSize',14);
ylabel('b (sec)','FontSize',14);
zlabel('CWT','FontSize',14)
view([130 37]);
caxis([-20 40]);

% Note the severe degradation of the resulting time-scale plot.
```

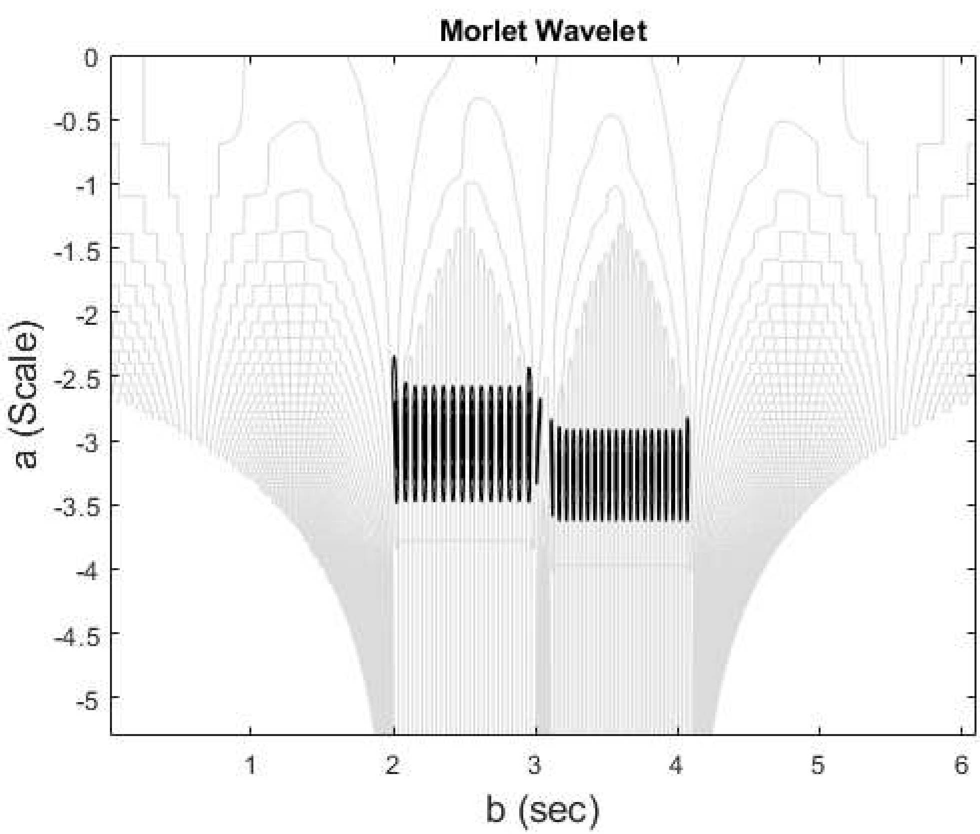




Question 10 part 1:

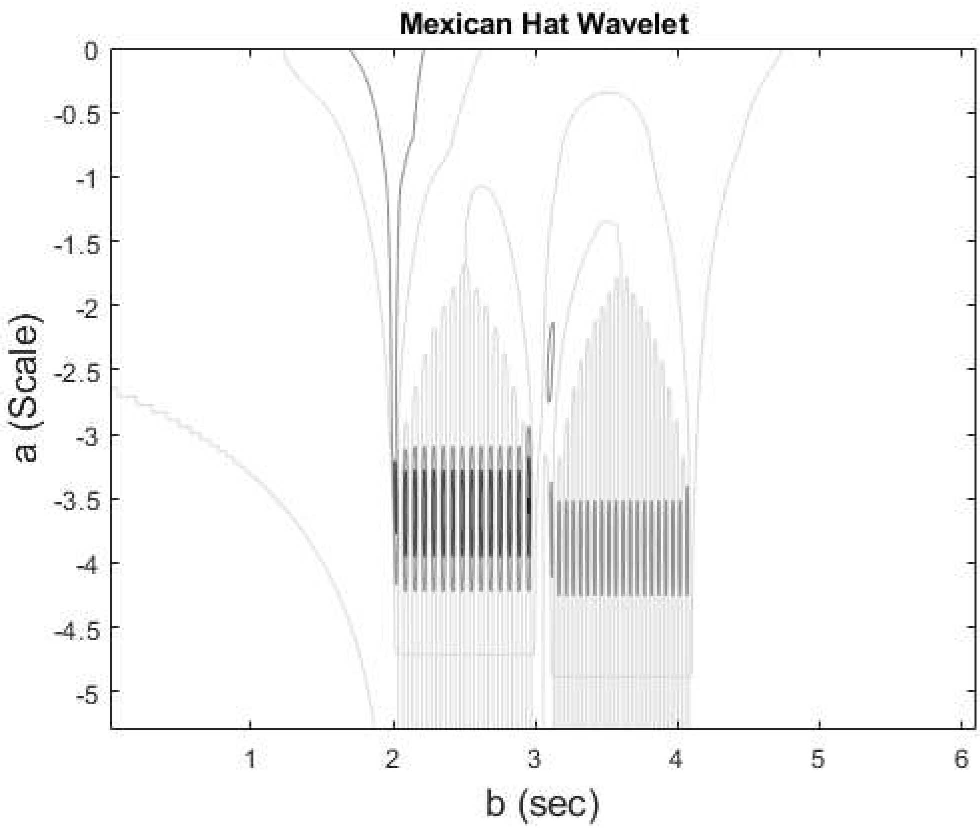
```
clear all
%Loading the signal data
s = load('time_freq4.mat');
```

```
x = s.x;

fs = 500; % Sample frequency
N = length(x);
n = N/4; % Signal length divided by 4
resol_level = 200; % Number of levels of a
decr_a = 1; % Decrement for a
a_init = 1; % Initial a
wo = pi * sqrt(2/log2(2));
b = (1:N)/fs; % Time vector for wavelet and plotting

% Calculate Continuous Wavelet Transform
for k = 1:resol_level
    a(k) = a_init/(k*decr_a); % Set scale
    t = b/a(k); % Time vector for Wavelet

    wav = (exp(-t.^2).* cos(wo*t))/sqrt(a(k)); % Generate Morlet Wavelet

    psi = [fliplr(wav) wav(2:end)]; % Make symmetrical about 0
    wlet(k,:) = psi;
    CW_Trans(:,k) =conv(x,psi,'same'); % Remove extra points from each end
end
a = log(a);

figure;
colormap(flipud(gray)); % Invert colormap for better viewing
contour(b,a,CW_Trans'); % Contour plot
title('Morlet Wavelet');
ylabel('a (Scale)','FontSize',14);
xlabel('b (sec)','FontSize',14);
caxis([-5 30]);
```



Question 10 part 2:

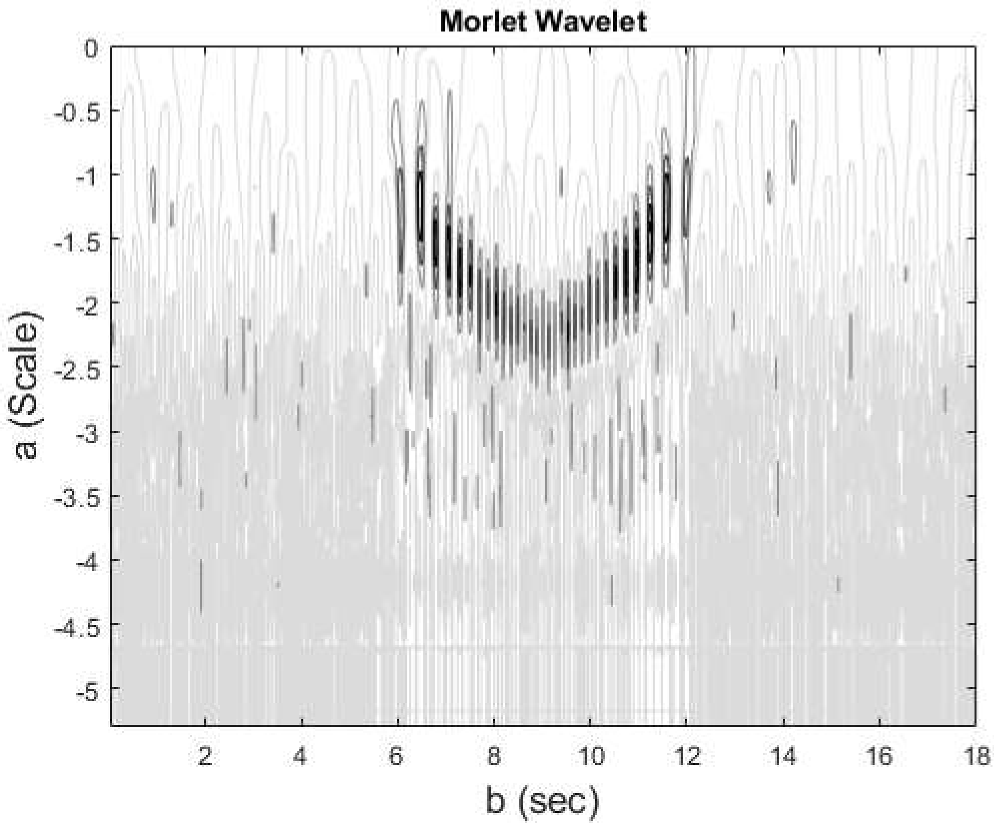
```
clear all
s = load('time_freq4.mat');
x = s.x;
fs = 500; % Sample frequency
N = length(x);
n = N/4; % Signal length divided by 4
resol_level = 200; % Number of levels of a
decr_a = 1; % Decrement for a
a_init = 1; % Initial a
wo = pi * sqrt(2/log2(2));
b = (1:N)/fs; % Time vector for wavelet and plotting

% Calculate Mexican hat Wavelet Transform
for k = 1:resol_level
    a(k) = a_init/(k*decr_a); % Set scale
    t = b/a(k); % Time vector for Wavelet

    wav = (1-2*(t.^2)).*(exp(-t.^2));   % mexican hat wavelet equation

    psi = [fliplr(wav) wav(2:end)]; % Make symmetrical about 0
    wlet(k,:) = psi;
    CW_Trans(:,k) =conv(x,psi,'same'); % Remove extra points from each end
end
a = log(a);

figure;
colormap(flipud(gray)); % Invert colormap for better viewing
contour(b,a,CW_Trans'); % Contour plot
title('Mexican Hat Wavelet');
ylabel('a (Scale)','FontSize',14);
xlabel('b (sec)','FontSize',14);
caxis([-5 30]);

% There is greater time separation for the Mexican Hat Wavelet Transform
% compared to the Continuous Wavelet Transform.
% The frequency separation is just slightly greater in Continuous Wavelet
% Transform in comparison with Mexican Hat Wavelet Transform.
```



Question 11** part 1:

```
clear all
%Loading the signal data
s = load('time_freq6.mat');
x = s.x;

fs = 100;
N = length(x);
n = N/4; % Signal length divided by 4
resol_level = 200; % Number of levels of a
decr_a = 1; % Decrement for a
a_init = 1; % Initial a
wo = pi * sqrt(2/log2(2));
b = (1:N)/fs; % Time vector for wavelet and plotting

% Calculate Continuous Wavelet Transform
for k = 1:resol_level
    a(k) = a_init/(k*decr_a); % Set scale
    t = b/a(k); % Time vector for Wavelet

    wav = (exp(-t.^2).* cos(wo*t))/sqrt(a(k)); % Generate Morlet Wavelet

    psi = [fliplr(wav) wav(2:end)]; % Make symmetrical about 0
    wlet(k,:) = psi;
    CW_Trans(:,k) =conv(x,psi,'same'); % Remove extra points from each end
end

a = log(a);

colormap(flipud(gray)); % Invert colormap for better viewing
contour(b,a,CW_Trans'); % Contour plot
title('Morlet Wavelet');
```

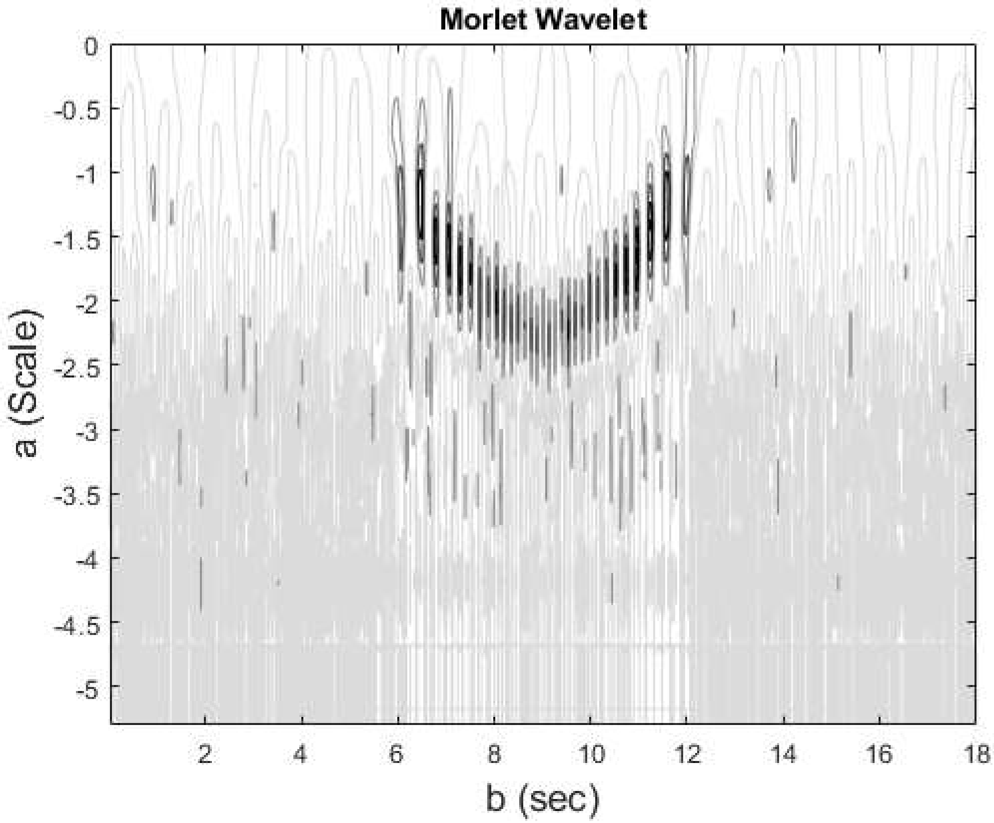
```
ylabel('a (Scale)','FontSize',14);
xlabel('b (sec)','FontSize',14);
caxis([-5 30]);
```

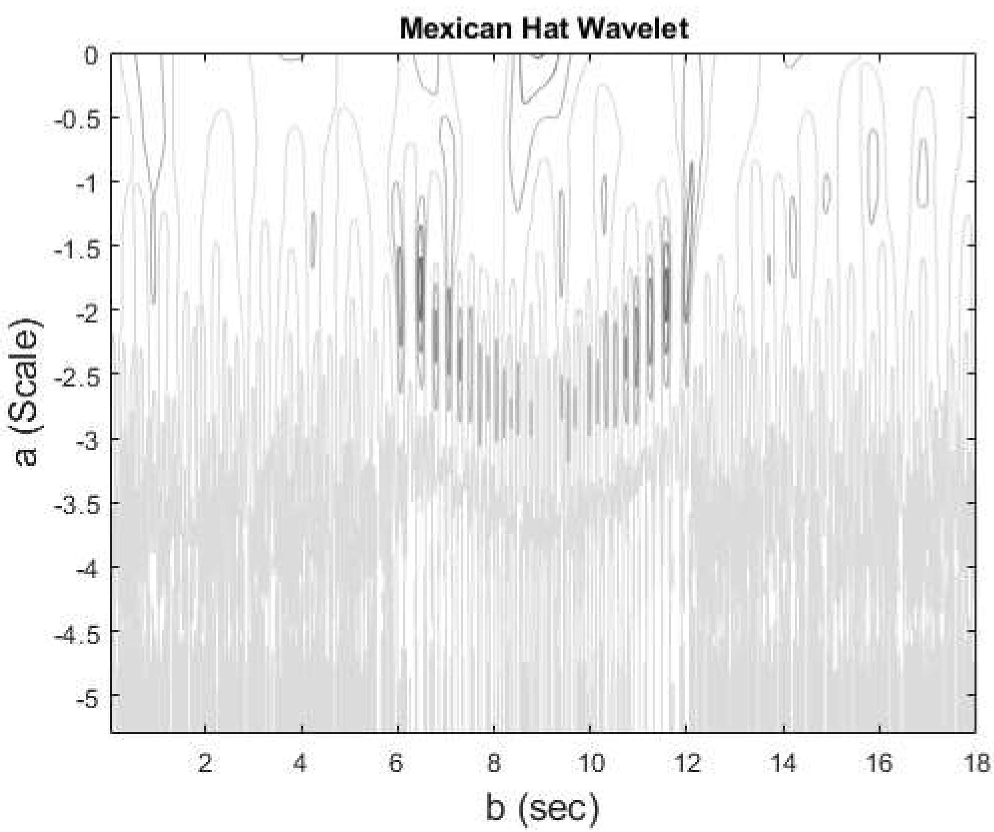


Question 11** part 2:

```
clear all
s = load('time_freq6.mat');
x = s.x;

fs = 100;
N = length(x);
n = N/4; % Signal length divided by 4
resol_level = 200; % Number of levels of a
decr_a = 1; % Decrement for a
a_init = 1; % Initial a
wo = pi * sqrt(2/log2(2));
b = (1:N)/fs; % Time vector for wavelet and plotting

% Calculate Mexican hat Wavelet Transform
for k = 1:resol_level
    a(k) = a_init/(k*decr_a); % Set scale
    t = b/a(k); % Time vector for Wavelet

    wav = (1-2*(t.^2)).*(exp(-t.^2));   % mexican hat wavelet equation

    psi = [fliplr(wav) wav(2:end)]; % Make symmetrical about 0
    wlet(k,:) = psi;
    CW_Trans(:,k) =conv(x,psi,'same'); % Remove extra points from each end
end

a = log(a);
```

```
figure;
colormap(flipud(gray)); % Invert colormap for better viewing
contour(b,a,CW_Trans'); % Contour plot
title('Mexican Hat Wavelet');
ylabel('a (Scale)','FontSize',14);
xlabel('b (sec)','FontSize',14);
caxis([-5 30]);

% The continuous Wavelet Transform best shows the signal since the points
% are more well-defined and has a higher resolution overall compared to
% Mexican Hat Wavelet Transform.
```



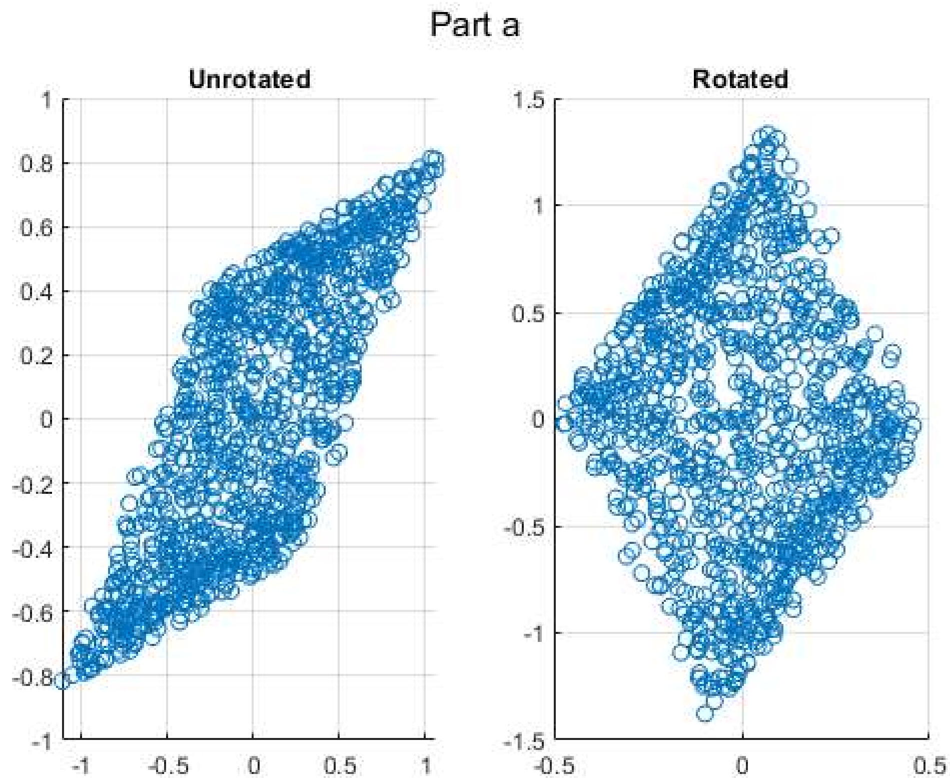


Question 12**

part a:

```
clear all
%Loading the signal data
s = load('p9_1_data.mat');
x = s.X;
x1 = x(1,:);
x2 = x(2,:);

fs = 500;
N = length(x1);

% after trial and error with different rotation values for phi, 0.867 was
% chosen as the value with the lowest covariance value.
% the phi values that were tested were in the range of: 0.4-0.9.

figure;
sgtitle('Part a');
subplot(1,2,1); scatter(x(1,:),x(2,:)); title('Unrotated'); grid on;

y = rotation(x, 0.867);
c = cov(y');
subplot(1,2,2); scatter(y(1,:), y(2,:)); title('Rotated'); grid on;

disp("-------- Part a --------");
disp("Variance for phi = 0.867:"); disp(c);
```

```
-------- Part a --------
Variance for phi = 0.867:
    0.0419   -0.0041
   -0.0041    0.3487
```

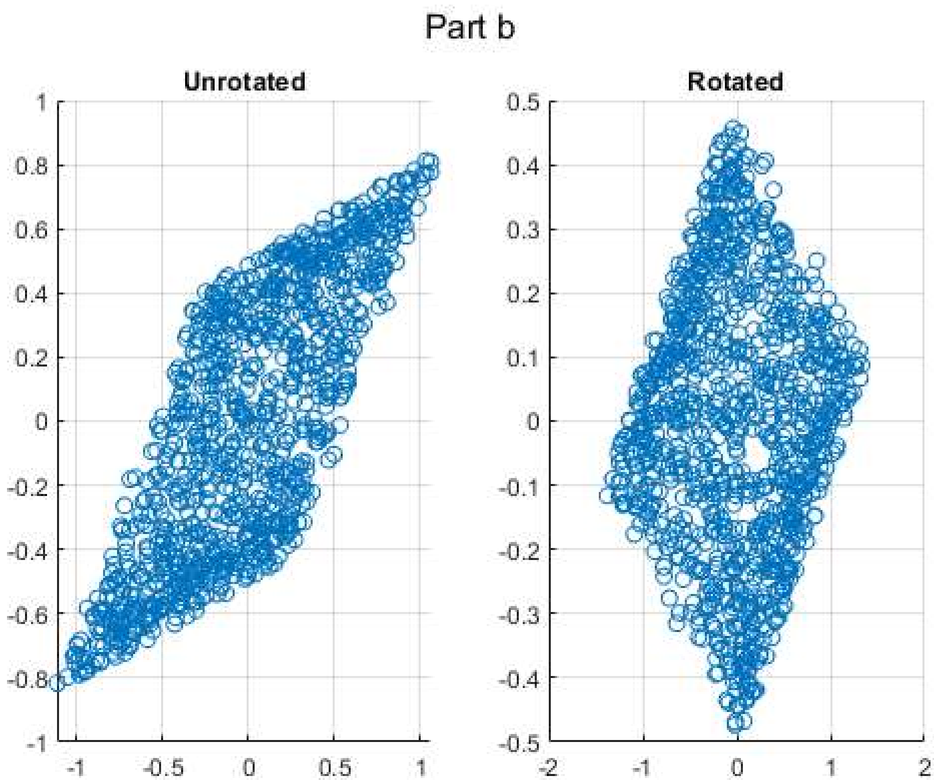


part b:

```
clear all
s = load('p9_1_data.mat');
X = s.X;
% Assign constants
N = 1000;       % Number points (2 sec of data)
fs = 500;       % Sample frequency
t = (1:N)/fs;


for i = 1:2             % Center data (i.e., remove means)
   X(i,:) = X(i,:) - mean(X(i,:));
end

%Find principal components
[U,S,pc]= svd(X,'econ');
eigen = diag(S).^2;
for i = 1:2
   pc(:,i) = pc(:,i) * sqrt(eigen(i));
end


% Calculate Eigenvlaue ratio
total_eigen = sum(eigen);
for i = 1:2
   pct(i) = 100 * sum(eigen(i:2))/total_eigen;
end
disp(pct)

S = cov(pc);    % Display covariance matrix

figure;
```

```
sgtitle('Part b');
subplot(1,2,1); scatter(X(1,:),X(2,:)); title('Unrotated'); grid on;
subplot(1,2,2); scatter(pc(:,1), pc(:,2)); title('Rotated'); grid on;

disp("-------- Part b --------");
disp("Variance:"); disp(S);

% The covariance values of the principal components of PCA are closer to
% zero compared to those found by manually rotating the data.
% Therefore, applying PCA can be considered an effective way for determining the
% principal components as well as minimizing covariance.
```

```
  100.0000   10.7236

-------- Part b --------
Variance:
    0.3488    0.0000
    0.0000    0.0419
```



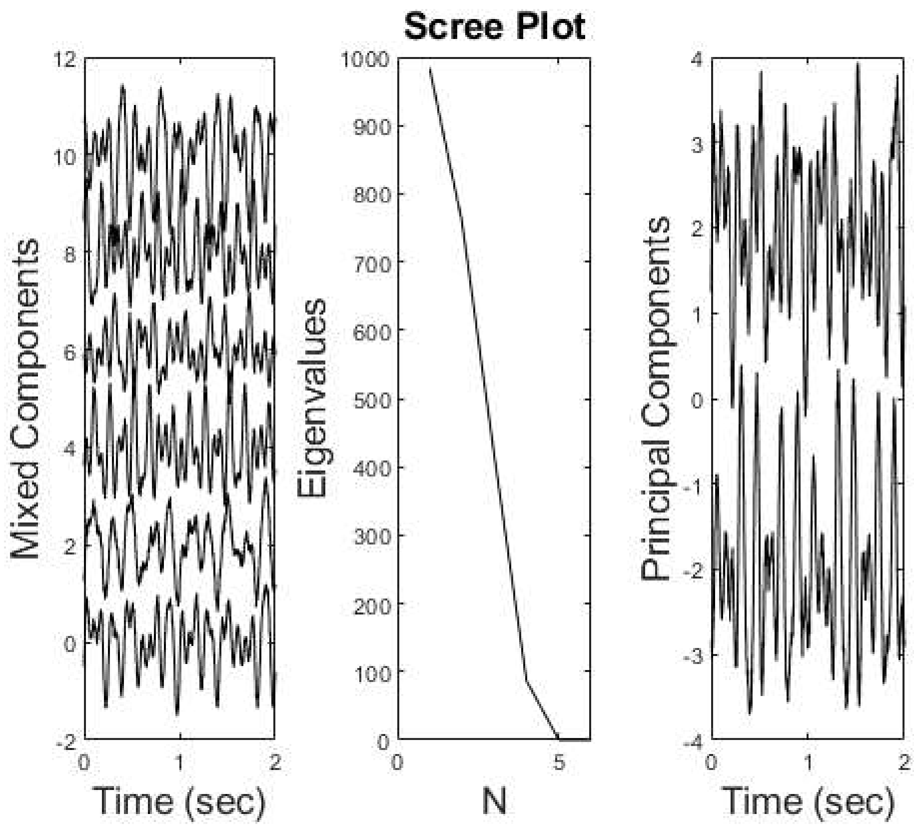
Question 13**:

```
%Loading the signal data
s = load('p9_2_data.mat');
X = s.X;

x1 = X(1,:);
fs = 500;
N = length(x1);
t = (1:N)/fs;
```

```
% PCA code:
for i = 1:6              % Center data (i.e., remove means)
   X(i,:) = X(i,:) - mean(X(i,:));
end

figure;
subplot(1,3,1);
for i = 1:6
     plot(t,X(i,:)+2*(i-1),'k');
      hold on;
end
xlabel('Time (sec)','FontSize',14)
ylabel('Mixed Components','FontSize',14);


%Find principal components
[U,S,pc]= svd(X,'econ');
eigen = diag(S).^2;
for i = 1:6
   pc(:,i) = pc(:,i) * sqrt(eigen(i));
end

subplot(1,3,2);
plot(eigen,'k');
xlabel('N','FontSize',14);
ylabel('Eigenvalues','FontSize',14);
title('Scree Plot','FontSize',14);

% Calculate Eigenvlaue ratio
total_eigen = sum(eigen);
for i = 1:6
   pct(i) = 100 * sum(eigen(i:6))/total_eigen;
end
disp(pct)

% Print Scaled Eigenvalues and Covariance matrix of Principal components
%       for comparison
%
S = cov(pc)     % Display covariance matrix
% Plot Principal Components and Original Data
subplot(1,3,3);
plot(t,pc(:,1)-2,'k',t,pc(:,2)+2,'k');  %Displace for clarity
xlabel('Time (sec)','FontSize',14)
ylabel('Principal Components','FontSize',14);

% By taking the sum of the various eigenvalues, it can be seen that last 3
% components of the variance contribute to less than 5% of the variance.
% Therefore, the actual dimension of the data can be estimated to be 3.
% By looking at the Scree plot, it can be seen that the point of inflection
% is at N=4 and then approaching zero for N>4. Therefore, the dimension can
% be estimated to be 3.
```

```
  100.0000   56.2172   22.4365    3.8575    0.0000    0.0000


S =

    0.9847    0.0000   -0.0000    0.0000    0.0000   -0.0000
```

```
    0.0000    0.7598    0.0000    0.0000   -0.0000   -0.0000
   -0.0000    0.0000    0.4179    0.0000    0.0000   -0.0000
    0.0000    0.0000    0.0000    0.0868    0.0000   -0.0000
    0.0000   -0.0000    0.0000    0.0000    0.0000   -0.0000
   -0.0000   -0.0000   -0.0000   -0.0000   -0.0000    0.0000
```

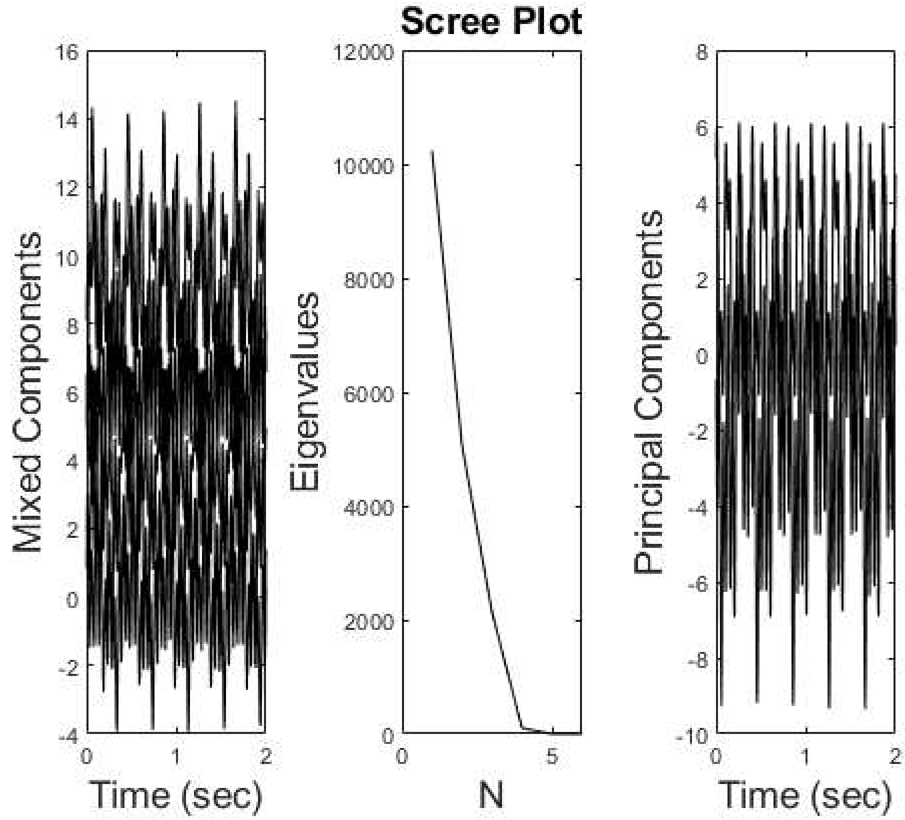


Question 14:

```
%Loading the signal data
s = load('prob9_3_data.mat');
X = s.X;
x1 = X(1,:);
fs = 500;
N = length(x1);
t = (1:N)/fs;

% PCA code:
for i = 1:6              % Center data (i.e., remove means)
   X(i,:) = X(i,:) - mean(X(i,:));
end

figure;
subplot(1,3,1);
for i = 1:6
     plot(t,X(i,:)+2*(i-1),'k');
      hold on;
end
xlabel('Time (sec)','FontSize',14)
ylabel('Mixed Components','FontSize',14);

%Find principal components
```

```
[U,S,pc]= svd(X,'econ');
eigen = diag(S).^2;
for i = 1:6
   pc(:,i) = pc(:,i) * sqrt(eigen(i));
end

subplot(1,3,2);
plot(eigen,'k');
xlabel('N','FontSize',14);
ylabel('Eigenvalues','FontSize',14);
title('Scree Plot','FontSize',14);

% Calculate Eigenvlaue ratio
total_eigen = sum(eigen);
for i = 1:6
   pct(i) = 100 * sum(eigen(i:6))/total_eigen;
end
disp(pct)

% Print Scaled Eigenvalues and Covariance matrix of Principal components
%       for comparison
%
S = cov(pc)     % Display covariance matrix
% Plot Principal Components and Original Data
subplot(1,3,3);
plot(t,pc(:,1)-2,'k',t,pc(:,2)+2,'k');  %Displace for clarity
xlabel('Time (sec)','FontSize',14)
ylabel('Principal Components','FontSize',14);
```

```
  100.0000   41.6606   12.7685    0.5991    0.0000    0.0000


S =

   10.2462    0.0000   -0.0000   -0.0000   -0.0000    0.0000
    0.0000    5.0744   -0.0000   -0.0000   -0.0000    0.0000
   -0.0000   -0.0000    2.1373   -0.0000   -0.0000    0.0000
   -0.0000   -0.0000   -0.0000    0.1052   -0.0000    0.0000
   -0.0000   -0.0000   -0.0000   -0.0000    0.0000    0.0000
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
```



Scree Plot
Mixed Components
Time (sec)
Eigenvalues
N
Principal Components
Time (sec)