B.E Bio signal Processing (BM-451)

LAB MANUAL

Name	**Muhammad Usairim Isani**
Roll No.	**BM-17040**
Year	**Fourth (BE)**
Batch	**2017**
Department	**Biomedical Engineering**

Department of Biomedical Engineering

N.E.D. University of Engineering & Technology,
LEJ Campus Karachi – 75270, Pakistan

Preface

This laboratory work book for course titled “Bio Signal Processing” has been designed in order to be up to date with curriculum changes. The lab manual is per OBE requirement. Following information is provided as per OBE requirement:

1. CLOs

1.	Conduct experiments on their own for recording, processing and analyzing bio-signals	**P3**	**PLO5**

2. LABORATORY EXPERIMENT EVALUATION RUBRIC

CATEGORY	**OUTSTANDING**	**ACCOMPLISHED**	**DEVELOPING**	**BEGINNER**
Implementati on/ Methodology	Effectively follow the provided instruction and complete the experiment procedures independently	Effectively follow the provided instruction and complete the experiment procedures with minimal supervision.	Follow the lab procedure completes experiment procedure with moderate supervision.	Cannot completes tasks and standard procedures
Modern techniques and skills	Demonstrates skilful ability in programming MATLAB tools.	Demonstrates ability in programming MATLAB tools.	Demonstrates some ability in programming MATLAB tools	Demonstrates minimal or no ability in programming MATLAB tools.
Output/Result s	Intelligently displays the output of program. Able to infer and discuss the results obtained. .	Displayed output is acceptable and to some extent conclusion of results is provided.	Displayed output is acceptable but poorly represented.	Unable to display the output

S. No	**Index**	**Sign.**
01	To familiarize the students with MATLAB environment through its some preliminary MATLAB functions will be also covered.	
02	a) To represent the basic signals like impulse, unit step, unit ramp, sinusoidal, cosine using MATLAB software and perform basic operation on signal. b) To represent bio signal i.e. ECG, EEG in MATLAB	
03	To analyze the basic properties including mean, standard deviation and variance of time variant signal under different conditions.	
04	To explain the concepts of ensemble averaging by analyzing multiple time locked epochs	
05	To present the use of auto correlation and cross correlation methods for computing peaks in bio signals.	
06	To explain the principle of orthogonality between bio signals using covariance	
07	To learn transformation of bio signal in frequency domain	
08	To estimate power spectral density of bio signal	
09	To explain concepts of data truncation using different windows	
10	To understand the welch method to estimate power spectral density	
11	To design digital FIR Filter design and implementation on bio signal using filter design tool of MATLAB	
12	To design digital FIR Filter design and implementation on bio signal using filter design tool of MATLAB	
1	To get familiar with the complexity analysis of EEG signal	

3		
1 4	To get familiar with the complexity analysis of ECG signal	
1 5	Processing and Visualization of bio signal in EEGLAB toolbox of MATLAB	
1 6	To learn the recording method of EEG signal	

COMMITMENTS PER WEEK	
Commitme nt/ week	**Go als**
0 1	To familiarize the students with MATLAB environment through it some preliminary MATLAB functions will be also covered
0 2	a) To represent the basic signals like impulse, unit step, unit ramp, sinusoidal, cosine using MATLAB software and perform basic operation on signal. b)To represent bio signal i.e. ECG, EEG in MATLAB
0 3	To analyze the basic properties including mean, standard deviation and variance of time variant signal.
0 4	To compute the ensemble average of bio signal
0 5	To compare bio signal using auto correlation and cross correlation method**.**
0 6	To find the auto covariance and cross covariance
0 7	Representation of bio signal in time frequency domain
0 8	Estimation of power spectral density of bio signal
0 9	Period gram of signal using windowing method
1 0	Power Spectral Density using Welch Method
1 1	Digital FIR Filter design and implementation on bio signal
1 2	Digital IIR Filter design and implementation on bio signal
1	Fractal Dimensionality of EEG signal

3	
1 4	Fractal Dimensionality of ECG signal
1 5	Processing and Visualization of bio signal in EEGLAB tool box of MATLAB
1 6	Recording of EEG signal

Lab Session 1

Objective: To familiarize the students with MATLAB environment through it some preliminary MATLAB functions will be also covered.

Introduction

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming environment. Furthermore, MATLAB is modern programming language environment: it has sophisticated data structures, contains built-in editing and debugging tools, and supports object-oriented programming. These factors make MATLAB an excellent tool for teaching and research.

Basic Functions in MATLAB

Entering a vector

A vector is a special case of a matrix. The elements of vectors in MATLAB are enclosed by square brackets and are separated by spaces or by commas. For example, to enter a row vector,v,

```
 type>> v = [1 4 7 10 13]

         v = 1    4     7     10     13
```

Column vectors are created in a similar way; however, semicolon (;) must separate the components of a column vector.

```
>> w = [1; 4; 7; 10; 13]

       w=14 7 10 13
```

On the other hand, a row vector is converted to a column vector using the transpose operator. The transpose operation is denoted by an apostrophe or a single quote (’).

```
>> w = v’

        w =1
        4
        7
        10
        13
```

To access blocks of elements, we use MATLAB's colon notation (:). For example, to access the first three elements of v, we write

```
>> v(1:3)
ans =1    4    7
```

Or, all elements from the third through the last elements.

```
>>   v(3,end)
ans =7    10
13
```

Defining a matrix

```
>> A = [1 2 3; 4 5 6; 7 8 9]
```

Functions to create matrix

Zeros (m, n) creates an mxn matrix whose elements are equal to zero
Ones (m, n) create an mxn matrix whose elements are equal to one.
eye (m,n) creates an mxn identity matrix
rand(m,n) creates an mxn matrix whose elements are all random number between 0 and 1

To access a particular element A (i, j) in MATLAB refers to the element Aij of matrix A. The first index is the row number the second index is the column number. For example, A (1, 3) is an element of first row and third column.

The colon operator can also be used to pick out a certain row or column. For example, the statement A (m: n, k: l) specifies rows m to n and column k to l.

```
>> A (2, : )
ans =4 5 6 is the second row elements of A
```

```
>> A (2:3, 2:3)
ans =5    6
      8    9
```

The colon operator can also be used to extract a sub-matrix from a matrix A.

```
>> A (: ,2:3)
ans =2 35 68 0
```

The keyword end, used in A (end, :), denotes the last index in the specified dimension.

To delete a row or column of a matrix, use the empty vector operator, [ ].

```
>> A(3,:) =
[] A = 1 2   3
     4   5   6
```

Arithmetic operators in MATLAB

MATLAB utilizes the following operators;

- + Addition
- -Subtraction
- Multiplication
- / Division
- ^ Power Operator
- ‘ transpose
- .*Element-by-element multiplication
- ./Element-by-element division
- .^Element-by-element exponentiation

Relational operators in MATLAB

MATLAB has 6 relational operators to make comparisons between variables. These are

- <is less than
- <=is less than or equal to
- >is greater than
- >=is greater than or equal to
- ==is equal to
- ~=is not equal to

In MATLAB a logical expression has two possible values that are not ‘true’ or ‘false’ but numeric, i.e. 1 if the expression is true and 0 if it is false. For example x= [-1, 3, 9] and y= [-5, 5, 9]. z= (x < y). The i-th element of z is 1 if x(i) < y(i), otherwise it is 0. So, the answer is the vector with elements 0, 1, and 0.

Logical operators in MATLAB

MATLAB uses three logical operators. These are

- And &
- Or |
- Not ~

For & (and) to give a true result both expressions either side of the & must be true.
For | (or) to give a true result only one of the expressions either side of the | needs to be true.
The ~ (not) operator changes a logical expression from 0 to 1 and vice versa.

Control flow in MATLAB

MATLAB has four control flow structures: the if statement, the for loop, the while loop,and the switch statement.The simplest form of the if statement is:

```
if expression
statements
end
```

In the for ... end loop, the execution of a command is repeated at a fixed and predetermined number of times. The syntax is:

```
for variable = expression
statements
end
```

Expression is a vector of the form i: s: j. A simple example of for loop is:

```
for ii=1:5

x=ii*ii
end
```

while loop is used when the number of passes is not specified. The looping continues until a stated condition is satisfied.

```
while expression
statements
end
```

The statements are executed as long as expression is true.

```
x = 1
while x <= 10
x = 3*x
end
```

Switch compares the input expression to each case value. Once the match is found it executes the associated commands.

Creating functions using m-files

There are two kinds of m-files: the script files and the function files. Script files do not take the input arguments or return the output arguments. The function files may take input arguments or return output arguments. Function M-files must always start with the function statement and the name of the function M-file must always be the same as the name of the function.

```
function tc = fahtocel(tf)     % converts function input 'tf' of temperatures in Fahrenheit to
                               % function output 'tc'of temperatures in Celsius
temp = tf-32;
 tc = temp*5/9;
 return;
```

Graphs in MATLAB

MATLAB has a large number of functions associated with graphical output.Use plot command to create plot. For example.

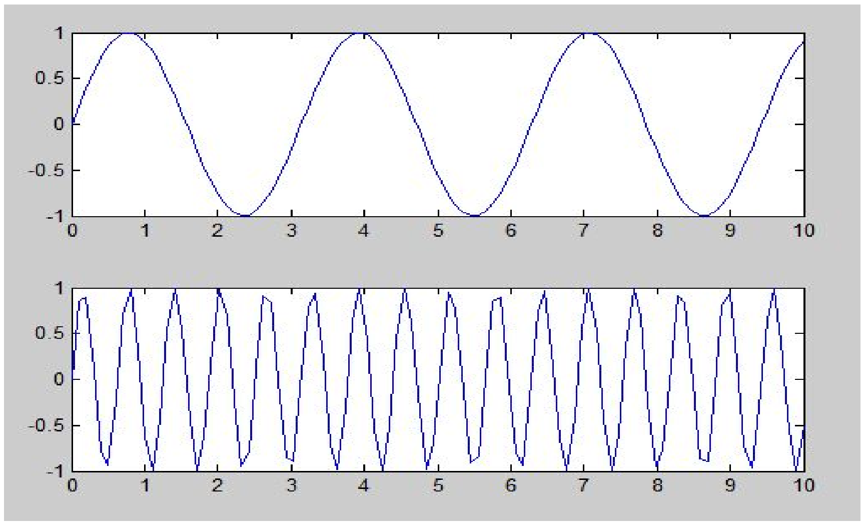
```
plot(date, maxtemp)
```

In MATLAB we can plot two or more graphs in one figure: For this Type plot(date,maxtemp); followed by hold on; Then type plot(date, mintemp); followed by hold off; hold on tells MATLAB to keep the old plot and add the new graph to it. hold off turns the hold-feature off again. We could also have used plot(date, maxtemp, date, mintemp); which tells MATLAB to plot maxtemp against date and then mintemp against date in the same graph. The colors of the graphs are then different.

A title can be created using the MATLAB command title and labels can be added using the xlabel and ylabel commands.

Several plots can be created in the same window using subplot command. Subplot (m,n,p) divides the current figure into an m-by-n grid and creates axes in the position specified by p. MATLAB numbers subplot positions by row. The first subplot is the first column of the first row; the second subplot is the second column of the first row, and so on. If axes exist in the specified position, then this command makes the axes the current axes. For example;

```
subplot(2,1,1);
x = linspace(0,10);
y1 = sin(2*x);
plot(x,y1)

subplot(2,1,2);
y2 = sin(10*x);
plot(x,y2)
```



Post Lab Tasks

1. Create two random vectors x and y, each with 5 elements. Write a for-loop to add x (1) to y(1), x(2) to y(2), etc. Each time, store the computed value in a variable called sum elements.

```
clear all;
close all;
clc;

x = 1:10;
y = 20:30;
z = rand(10,1);

for i = 1:10
   sum_elements(i) = x(i) + y(i) + z(i);
end

disp("Answer: ");
disp(sum_elements)
```

2. Create a function that computes the volume of a sphere with a certain radius. Call the function volsphere.

```
clear all;
close all;
clc;

r = input("Enter the radius of sphere: ");
v = volsphere(r);
fprintf("Volume: %f", v)

function vol = volsphere(r)
   vol = (4/3)*(pi*(r^3));
end
```

Lab Session 2

Objective

1. To represent the basic signals like impulse, unit step, unit ramp, sinusoidal, cosine using MATLAB software and perform basic operation on signal.
2. To represent bio signal i.e. ECG, EEG in MATLAB

Introduction

- **Signal**

Signals are broadly classified into continuous and discrete signals. A continuous signal will be denoted by x (t), in which the variable t can represent any physical quantity. A discrete signal will be denoted x[n], in which the variable n is integer value. In this lab we will learn to represent and operate on signals in MATLAB. The variables t and n are assumed to represent time.

- **Bio signal**

Bio signals, like all signals, must be “carried” by some form of energy Common biological energy sources include chemical, mechanical, and electrical energy Measuring a signal, whether of biological or external origin, usually entails conversion to an electric signal using a transducer Most biological signals exist in the analog domain and must be converted to digital signals for use in a computer. Much of the activity in biomedical engineering, be it clinical or in research, involves the measurement, processing, analysis, display, and/or generation of signals. Signals are variations in energy that carry information. The variable that carries the information (the specific energy fluctuation) depends on the type of energy involved.

Lab Activity 01: Representation of Basic Signals in MATLAB

Continuous Time Signal: Run the following lines. Provide the snapshots of the plots for each step given below.

```
% Sine and cosine signals
fy=1;
%signal frequency in Hz
wy=2*pi*fy;
%signal frequency in rad/s
fs=60;
%sampling frequency in Hz
tiv=1/fs;
%time interval between samples;
t=0:tiv:(3-tiv);
%time intervals set
ys=sin(wy*t);
%signal data set
plot(t,ys,'k'); hold on; %plots figure
axis([0 3 -1.5 1.5]);
xlabel('seconds');
yc=cos(wy*t); %signal data set
plot(t,yc,'--k');
%plots figure
axis([0 3 -1.5 1.5]);
label('seconds');
title('sine (solid) and cosine(dashed)');
```

Discrete Time Signals

For the following: Run the following lines. Provide the snapshots of the plots for each step given below

- Unit Impulse function is denoted by δ(t) and it is defined as δ(t) = {1 t=0
  0 t≠0

```
n1=input('Enter the no of samples');
x1=[-n1:1:n1];
y1=[zeros(1,n1),ones(1,1),zeros(1,n1)];
subplot(2,3,1);
stem(x1,y1);
xlabel('Time Period');
ylabel('Amplitude');
title('Unit Impulse Signal');
```

- Unit step function is denoted by u(t). It is defined as u(t) = {1 t=0
  0 t<0

```
%Unit Step Signal%
n2=input('Enter the no of samples');
x2=[0:1:n2];
y2=ones(1,n2+1); subplot(2,3,2);
stem(x2,y2); xlabel('Time Period'); ylabel('Amplitude'); title('Unit Step
Signal');
```

- Ramp signal is denoted by r(t), and it is defined as r(t) = {t0t⩾0t<0

```
%Unit Step Signal%
n2=input('Enter the no of samples');
x2=[0:1:n2];
y2=ones(1,n2+1);
subplot(2,3,2);
stem(x2,y2); xlabel('Time Period'); ylabel('Amplitude');
title('Unit Step Signal');
```

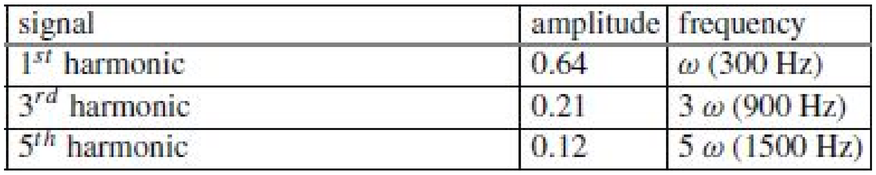
Operations with Signals

Adding Signals

We can add, subtract, multiply, etc. signals. It is interesting to deal, in this section, with some basic examples. Following program add three sinusoidal signals. This is done in the sentence with the following expression:

$$y = 0.64 \sin(\omega y\ t) + 0.21 \sin(3\omega_y t) + 0.12 \sin(5\omega_y t) \quad (1.1)$$

The amplitude and frequency of the three sinusoids are the following:



signal	amplitude	frequency
1^{st} harmonic	0.64	ω (300 Hz)
3^{rd} harmonic	0.21	3 ω (900 Hz)
5^{th} harmonic	0.12	5 ω (1500 Hz)

```
% Sum of sines signal
fy=300; %signal frequency in Hz
wy=2*pi*fy; %signal frequency in rad/s
fs=6000; %sampling frequency in Hz
tiv=1/fs;     %time interval between
samples;
t=0:tiv:(0.01-tiv); %time intervals set (0.01 second)
y=0.64*sin(wy*t)+0.21*sin(3*wy*t)+0.12*sin(5*wy*t);
plot(t,y,'k'); %plots figure
axis([0 0.01 -1.5 1.5]);
xlabel('seconds'); title('sum of sines signal');
```

Multiplication

```
% Multiplication of sines signal
fx=70; %signal frequency in Hz
wx=2*pi*fx; %signal frequency in rad/s
fz=2; %signal frequency in Hz
wz=2*pi*fz; %signal frequency in rad/s
fs=6000; %sampling frequency in Hz
tiv=1/fs; %time interval between samples;
t=0:tiv:(1-tiv); %time intervals set (1 second)
y=sin(wx*t).*sin(wz*t); %signal data set
plot(t,y,'k'); %plots figure
axis([0 1 -1.5 1.5]);
xlabel('seconds'); title('multiplication of sines signal');
```

- **Lab Activity 2: Representation of Bio signal in MATLAB**

Data Bank of Physiological Signals

A number of data banks exist that provide physiological signals such as ECG, EEG, gait, and other common bio signals in digital form. Given the volatility and growth of the Web and the ease with which searches can be made, no attempt will be made to provide a comprehensive list of appropriate Websites. However, a good source of several common bio signals, particularly the ECG, is the Physio Net Data Bank maintained by MIT—http://www.physionet.org. Some data banks are specific to a given set of bio signals or a given signal processing approach. An example of the latter is the ICALAB Data Bank in Japan—http:// www.bsp.brain.riken.go.jp/ICALAB/ which includes data that can be used to evaluate independent component analysis.

PhysioBank

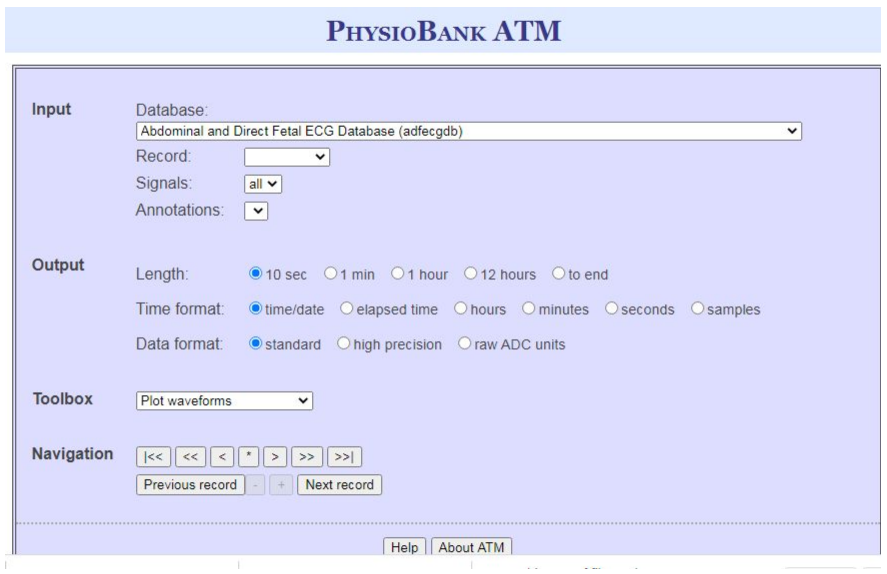
PhysioBank is a large and growing archive of well-characterized digital recordings of physiological signals and related data for use by the biomedical research community. It currently includes databases of multipara meter cardiopulmonary, neural, and other biomedical signals from healthy subjects and from patients with a variety of conditions with major public health implications, including life- threatening arrhythmias, congestive heart failure, sleep apnea, neurological disorders, and aging.

PhysioBank currently contains over 36,000 recordings of annotated, digitized physiologic signals and time series, organized in over 50 databases (collections of recordings). All are freely available from PhysioNet. Currently available databases in the PhysioBank archives are:

Clinical Databases: Data from critical care clinical settings that may include demographics, vital sign measurements made at the bedside, laboratory test results, procedures, medications, caregiver notes, images and imaging reports, and mortality (both in and out of hospital).
Waveform Databases: High resolution continuous recordings of physiological signals. Waveform databases are organized according to their signal and annotation types:
Multi-Parameter Databases: Available signals vary, but may include ECG, continuous invasive blood pressure, respiration, oxygen saturation, and EEG, among others.
ECG Databases: Most of which include ECG signals.
Interbeat (RR) Interval Databases. These contain beat annotations obtained from ECG recordings, but the ECG signals are not available. Also see ECG Databases, most of which include beat annotations in addition to the original ECG signals.
Other Cardiovascular Databases
Gait and Balance Databases
Neuroelectric and Myoelectric Databases. EEG, EHG, and more.
Image Databases
Synthetic Data
Computing in Cardiology Challenge Datasets

- PhysioBank ATM

It is a component of PhysioBank and allows viewing any of the recordings and annotations in PhysioBank within web browser. Choose a few records from several different databases to get a sense of the variety of data available. The PhysioBank ATM is best suited for a quick overview rather than in-depth study.


PhysioBank ATM

Input
Database: Abdominal and Direct Fetal ECG Database (adfecgdb)
Record:
Signals: all
Annotations:

Output
Length: [✓ 10 sec] 1 min 1 hour 12 hours to end
Time format: [✓ time/date] elapsed time hours minutes seconds samples
Data format: [✓ standard] high precision raw ADC units

Toolbox: Plot waveforms

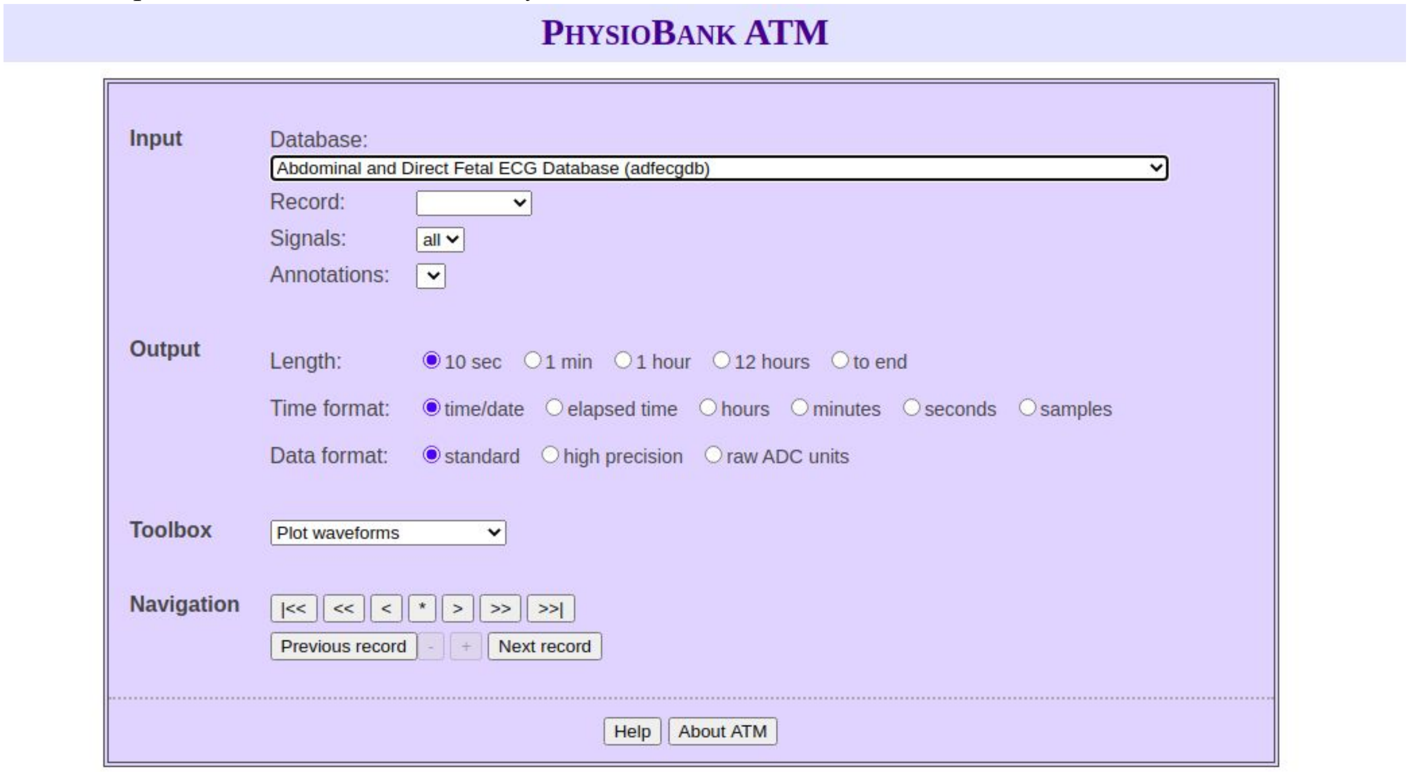
Each database consists of a set of records (recordings), identified by the *record name*. Lists of record names for each database can be found here. In most cases, a record consists of at least three files. The three files are .atr, .dat, and .hea. .dat (signal) files, containing digitized samples of one or more signals; these files can be very large. The .hea (header) file is a short text file that describes the signals (including the name or URL of the signal file, storage format, number and type of signals, sampling frequency, calibration data, digitizer characteristics, record duration and starting time). Most records include one or more binary *annotation* files (in the example, .atr denotes an annotation file). Annotation files contain sets of labels (annotations), each of which describes a feature of one or more signals at a specified time in the record.

Plotting ECG in MATLAB

To plot ECG in MATLAB, first download the recording from PhysioBank ATM and use MATLAB commands and information of the recording. Also attach the plot of signal.

Post Lab Task

1. Explore different databases of PhysioBank ATM.



PHYSIOBANK ATM

Input

Database: Abdominal and Direct Fetal ECG Database (adfecgdb)

Record:

Signals: all

Annotations:

Output

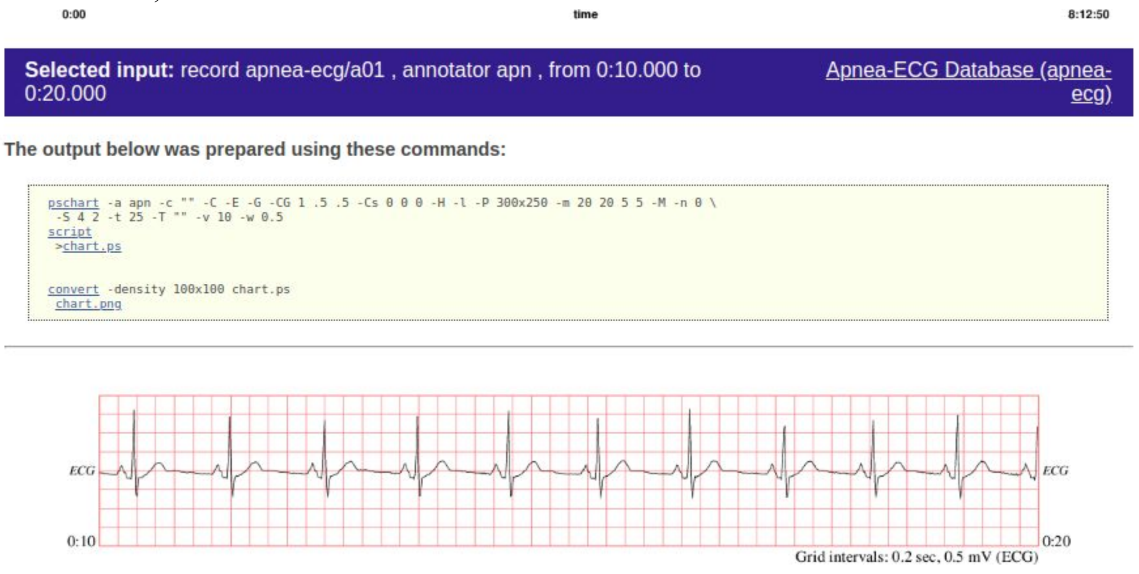
Length: [✓ 10 sec] 1 min 1 hour 12 hours to end

Time format: [✓ time/date] elapsed time hours minutes seconds samples

Data format: [✓ standard] high precision raw ADC units

Toolbox Plot waveforms

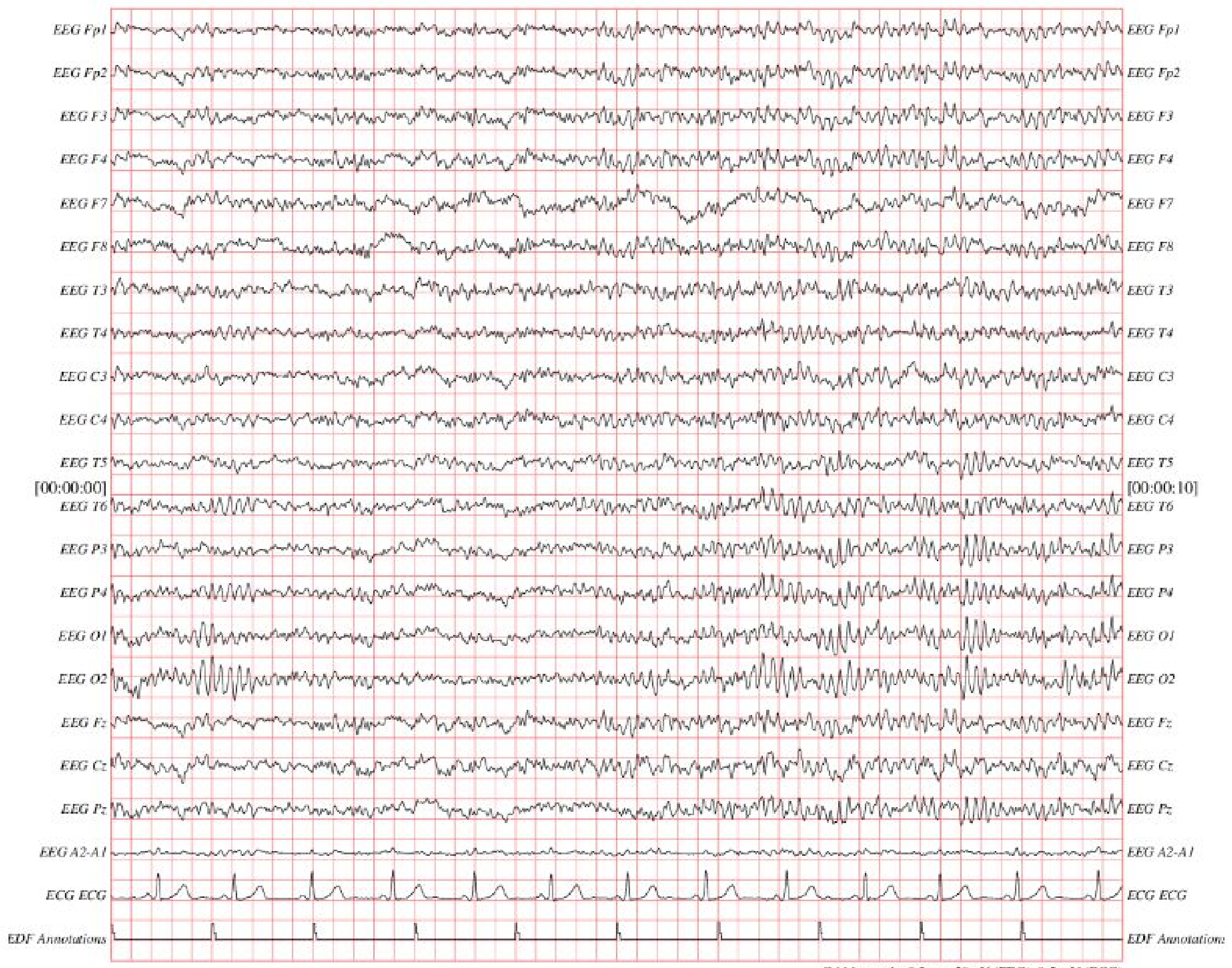
2. Plot two ECG, EEG records available databases.



Selected input: record apnea-ecg/a01 , annotator apn , from 0:10.000 to 0:20.000 — Apnea-ECG Database (apnea-ecg)

The output below was prepared using these commands:

```
pschart -a apn -c "" -C -E -G -CG 1 .5 .5 -Cs 0 0 0 -H -l -P 300x250 -m 20 20 5 5 -M -n 0 \
 -S 4 2 -t 25 -T "" -v 10 -w 0.5
script
 >chart.ps


convert -density 100x100 chart.ps
 chart.png
```



Grid intervals: 0.2 sec, 50 uV (EEG), 0.5 mV (ECG)

Lab Session 3

Objective: To analyze the basic properties including mean, standard deviation and variance of bio signal.

Introduction:

Mean

One of the most straightforward of signal measurements is the assessment of its average value. Averaging is most easily described in the digital domain. To determine the average of a series of numbers, simply add the numbers together and divide by the length of the series (i.e., the number of numbers in the series). Mathematically stated:

$$x_{avg} = \bar{x} = \frac{1}{N}\sum_{n=1}^{N} x_n$$

For a continuous analog signal, x (t), the summation becomes an integration. The average or mean of a continuous signal, the continuous version of above equation is obtained by integrating the signal over time and dividing by the length of the signal, which is now a time length.

$$\bar{x}(t) = \frac{1}{T}\int_0^T x(t)\,dt$$

Variance

The variance is a measure of signal variability irrespective of its average. The calculation of variance for both discrete and continuous signals is given as:

$$\sigma^2 = \frac{1}{T}\int_0^T (x(t) - \bar{x})^2\,dt$$

Standard Deviation

The standard deviation is another measure of a signal's Variability and is simply the square root of the variance.

$$\sigma = \left[\frac{1}{T}\int_0^T (x(t) - \overline{x})^2\, dt\right]^{1/2}$$

$$\sigma = \left[\frac{1}{N-1}\sum_{n=1}^{N} (x_n - \overline{x})^2\right]^{1/2}$$

MATLAB Function

xm=mean(x); % Evaluate mean of x
xvar=var(x); % Variance of x normalizing by N-1
xstd=std(x); % Evaluate the standard deviation of x

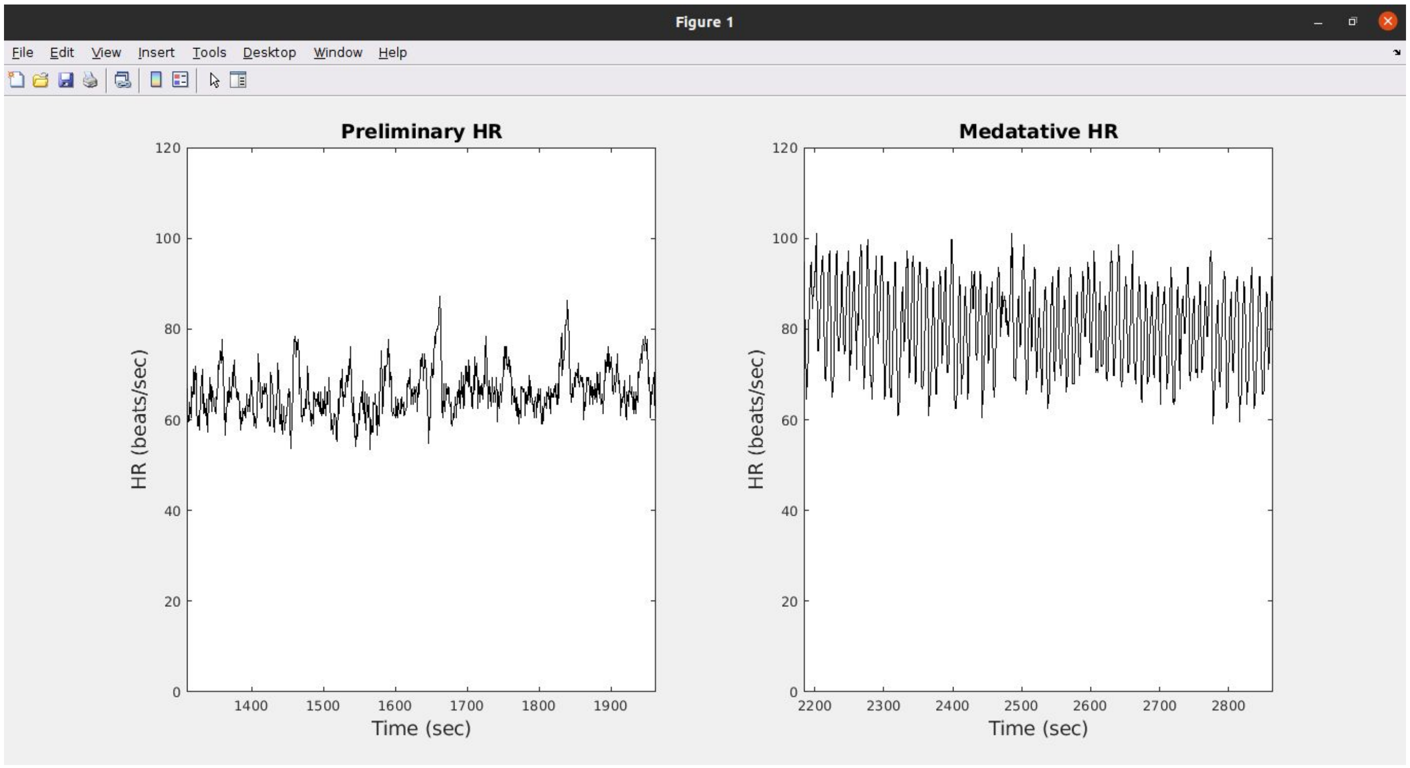
If x is not a vector, but a matrix, then the output is a row vector resulting from application of the calculation (mean, variance, or standard deviation) to each column of the matrix.

Lab Activity 01: Calculate the mean, variance, or standard deviation of ECG signal showing heart rate variability of one subject under normal and meditative state. Download the signal from database of PhysioBankATM. Attach the plot and interpret the results.

```
clear all;
close all;

load Hr_pre % Premeditative HR load Hr_med  % Meditative HR
load  Hr_med

% Calculate the averages and standard deviations
Avg_pre = mean(hr_pre)  % Average pre HR
SD_pre = std(hr_pre)    % Standard deviation pre
Avg_med = mean(hr_med)  % Average and std meditative HR
SD_med = std(hr_med)
subplot(1,2,1);
plot(t_pre,hr_pre,'k'); % Plot heart rate data
xlabel('Time (sec)','FontSize',14);
ylabel('HR (beats/sec)','FontSize',14);
axis([t_pre(1) t_pre(end) 0 120]);
title('Preliminary HR','FontSize',14);
subplot(1,2,2);
plot(t_med,hr_med,'k'); % Plot heart rate data
xlabel('Time (sec)','FontSize',14);
ylabel('HR (beats/sec)','FontSize',14);
axis([t_med(1) t_med(end) 0 120]);

title('Medatative HR','FontSize',14);
```

Output:

```
Avg_pre =   66.5123
SD_pre =    5.3575
Avg_med =   81.3330
SD_med =    9.3507
```



Lab Activity 02: In MATLAB, calculate the mean of 10 sec EEG signal of one subject, recorded during mental arithmetic task. Download from physioBank ATM. Attach the plot and results.

```
clear all;
close all;
clc;

load Subject00_1_edfm

for i = 1:22
   m(i) = mean(val(i,:));
end

figure(1);
stem(1:22, m);
title("Mean of All (22) EEG leads during mental arithmetic task");
xlabel("Lead number");
ylabel("Mean amplitude");
```

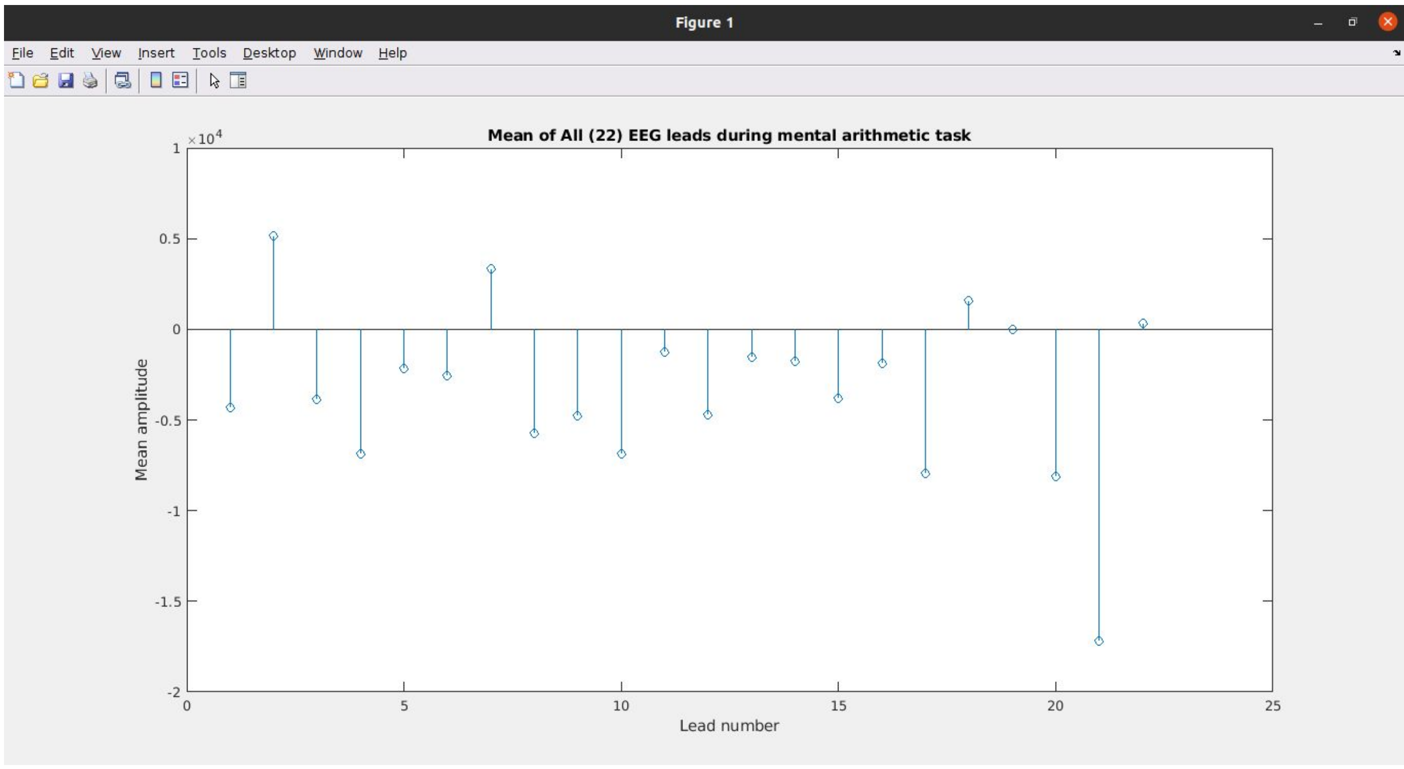


Figure 1
File Edit View Insert Tools Desktop Window Help
Mean of All (22) EEG leads during mental arithmetic task
×10^4
1
0.5
0
-0.5
-1
-1.5
-2
Mean amplitude
0
5
10
15
20
25
Lead number

Lab session 04

Objective: To explain the concepts of ensemble averaging by analyzing multiple time locked epochs

Introduction

Ensemble averaging is a simple yet powerful signal processing technique for reducing noise when multiple observations of the signal are possible. Such multiple observations could come from multiple sensors, but in many biomedical applications the multiple observations come from repeated responses to the same stimulus. In *ensemble averaging*, a number of time responses are averaged together on a point-by-point basis. Ensemble averaging is a simple yet powerful tool for reducing noise, but requires multiple observations of a signal, which are not always possible. Isolating evoked neural responses such as the visual evoked response (VER) requires averaging hundreds of responses to a repetitive stimulus.

Lab Activity 01: Find the average response of a number of individual VER. This is an EEG signal recorded near the visual cortex following a visual stimulus such as a flash of light. Attach the plots and infer the results.

```
clear all;
close all;
clc;

load ver; % load visual evoked signal (ver) data
fs = 1/0.005; % sampling freq
[r, c] = size(ver);
if r>c % do as per dim of vector
    ver = ver';
    t = (1:r)/fs; % time scale
else
    t = (1:c)/fs;
end

ensamble_avg = mean(ver);
figure(1);
plot(t, ensamble_avg);
title("Average Ensemble of VER signal");
xlabel("Time (ms)");

figure(2);
plot(t, ver(1,:)); % plot individual ver
```

```
title("Individual (First) VER signal");
xlabel("Time (ms)");
```

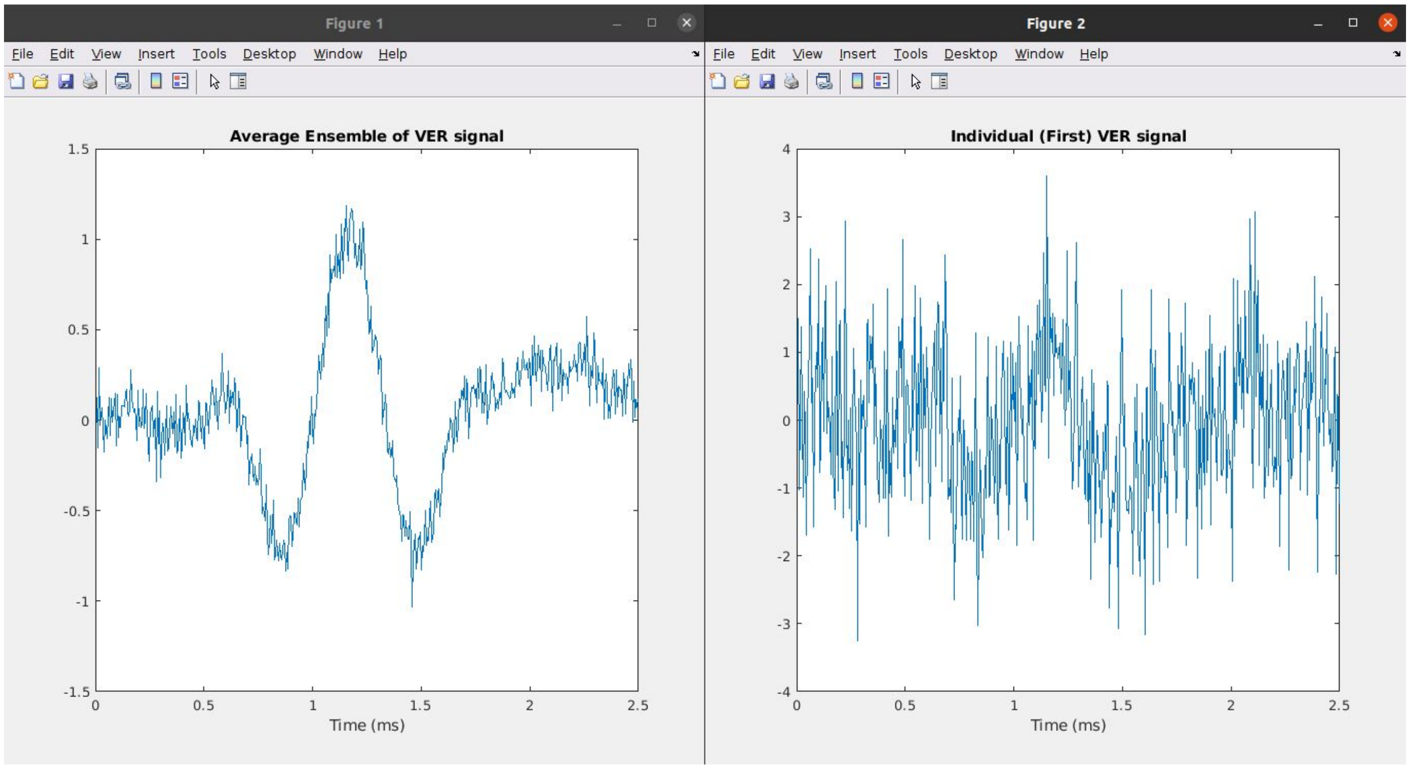


Figure 1
File Edit View Insert Tools Desktop Window Help
Average Ensemble of VER signal
Time (ms)
Figure 2
File Edit View Insert Tools Desktop Window Help
Individual (First) VER signal
Time (ms)

Lab Activity 02: Download the VER (visual evoked responses) EEG recording from MAMEM SSVEP Data base of PysioBank ATM. The data should be stored in matrix which should contain 100 responses. Plot several randomly selected samples of these responses. Construct and plot the ensemble average for this data. Also construct and plot the ensemble average of responses.

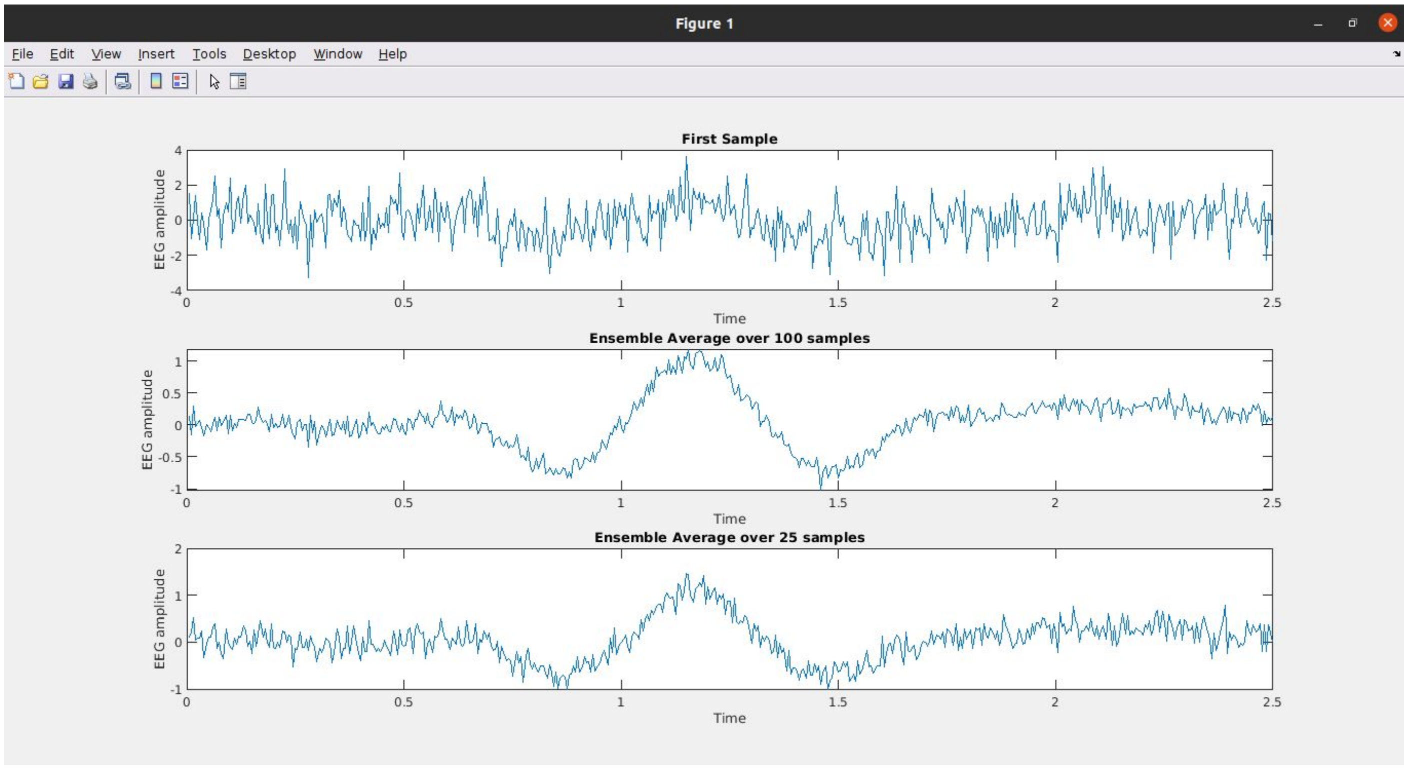
```
clear all;
close all;
clc;

load ver; % load visual evoked signal (ver) data
fs = 1/0.005; % sampling freq
[r, c] = size(ver);
if r>c % do as per dim of vector
    ver = ver';
    t = (1:r)/fs; % time scale
else
    t = (1:c)/fs;
end
```

```
avg100 = mean(ver); % ensemble average of 100 samples
avg25 = mean(ver(1:25,:)); % ensemble avg of 25 samples

% plot differently avg and non-avg signals
subplot(3,1,1);
plot(t,ver(1,:)); % first sample
title("First Sample");
xlabel("Time");
ylabel("EEG amplitude")

subplot(3,1,2);
plot(t,avg100); % 100 avg ensemble
title("Ensemble Average over 100 samples");
xlabel("Time");
ylabel("EEG amplitude")

subplot(3,1,3);
plot(t,avg25); % avg 25 samples
title("Ensemble Average over 25 samples");
xlabel("Time");
ylabel("EEG amplitude")

% find how much noise is present in a signal
ver_noise = ver(1,:) - actual_ver; % actual_ver is in file ver.m
avg100_noise = avg100 - actual_ver; % noise in 100 avg signal
avg25_noise = avg25 - actual_ver;

% some statistics of noise
ver_noise_std = std(ver_noise);
avg100_noise_std = std(avg100_noise);
avg25_noise_std = std(avg25_noise);

% display
fprintf("Std of average ensemble data %f",ver_noise_std);
disp(" ");
fprintf("Std of 100 average samples %f",avg100_noise_std);
disp(" ");
fprintf("Std of 25 average samples %f",avg25_noise_std);

%disp("Std of average ensemble data "ver_noise_std);
```

Output:

```
Std of average ensemble data 1.040470
```

Std of 100 average samples 0.100550
Std of 25 average samples 0.198620



Post Lab Tasks

1. What is the advantage of ensemble average applied on biological signal?

   Ensemble is a technique that allows noise reduction by taking the average of multiple recordings of a signal. This technique is proved to be useful for reducing noise by a factor of root(n), where *n* is the number of averaged samples.

2. Biological signals are affected by noise of various types. How can we reduce the electrical noise?

   One technique to remove electrical or line noise can be to use a band stop filter for 50 or 60 Hz frequency, as these are two frequencies of AC supply around the globe.

3. What is the difference between ensemble average and mean of signal?

   The ensemble average is the average of multiple signal recordings, i.e we average two recordings element wise. Whereas, the mean is the average of a single signal or sample, i.e adding the entire signal and then dividing it by its length.

Lab Session 05

Objective: To present the use of auto correlation and cross correlation methods for computing peaks in bio signals.

Introduction

Correlation

One of the most common ways of quantitatively comparing two functions is through correlation. Correlation seeks to quantify how much one function is like another. Correlation use multiplication to compare the linear relationship between two variables, but in correlation the coefficients are normalized to fall between zero and one. The linear correlation between two functions or signals can be obtained using the given formula.

$$r_{xy} = \frac{1}{T}\int_{0}^{T} x(t)y(t)dt \text{....(i)}$$

$$r_{xy} = \frac{1}{N}\sum_{k=1}^{N} x(k)y(k) \text{...(ii)}$$

where rxy is the un scaled correlation between x and y. If continuous functions are involved, the summation becomes an integral and the discrete functions, x[n] and y[n], become continuous functions, x(t) and y(t)

- Cross Correlation

Cross correlation shows the similarity between two waveforms at all possible relative positions of one waveform with respect to the other, and it is useful in identifying segments of similarity.

- Auto Correlation

A special case of the correlation function occurs when the comparison is between two waveforms that are one in the same; that is, a function is correlated with different shifts of itself. This is termed the autocorrelation function and it provides a description of how similar a waveform is to itself at various time shifts, or time lags.

MATLAB Function

MATLAB has specific functions for performing cross correlation/ autocorrelation. The cross correlation and autocorrelation operations are both performed with the same MATLAB routine, with autocorrelation being treated as a special case: The xcorr function produces an output argument, c, that is a vector of length 2. It returns the cross-correlation of two discrete-time sequences.

[c, lags] =xcorr (x,y,maxlags,‘options’)

Correlation or covariance matrices are calculated using the **corrcoef.**

Rxx = corrcoef(x)

where X is a matrix that contains the various signals to be compared in columns. Some options are available as explained in the associated MATLAB help file. The output, Rxx, of the corrcoef routine will be an *n*-by-*n* matrix where *n* is the number of signals (i.e., columns). The diagonals of this matrix represent the correlation of the signals with themselves and therefore will be 1. The off-diagonals represent the correlation coefficients of the various combinations. For example, *r*12 is the correlation between signals 1 and 2. Since the correlation of signal 1 with signal 2 is the same as signal 2 with signal 1, *r*12=*r*21, and in general *r*m,n=*r*n,m, the matrix will be symmetrical about the diagonals.

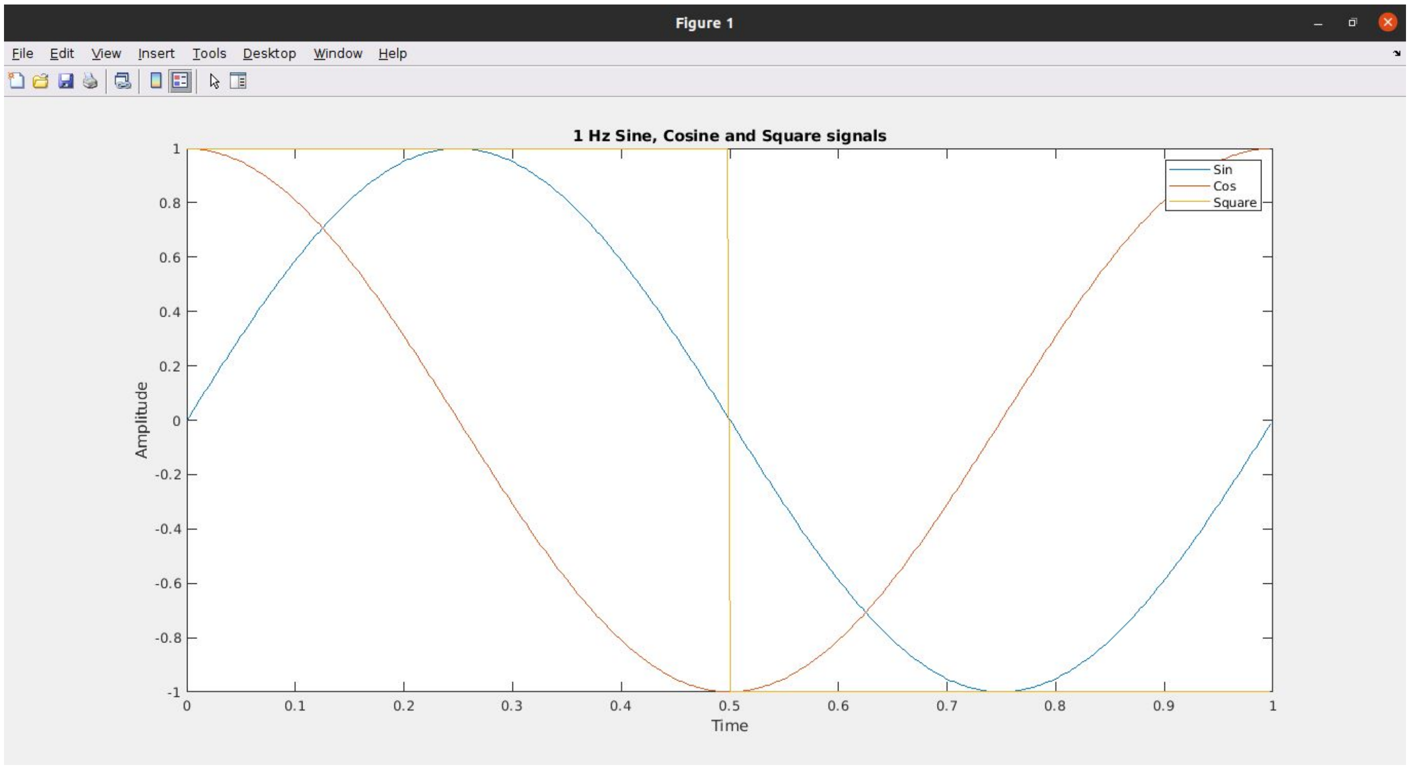
Lab Activity 01: Use MATLAB function to find the correlation between a sine wave, a cosine wave and square wave.

```
clear all;
close all;
clc;

N=500; % no of data points
T = 1; % time period of signals
fs= N/T; % sampling freq
t = [0:N-1]/fs; % Time vector

x=sin(2*pi*t); % 1 Hz sine wave
y=cos(2*pi*t); % 1 Hz cosine wave
z = square(2*pi*t); % 1 Hz square wave

% plotting all 3 signals
figure(1);
plot(t,x);
hold on;
plot(t,y);
plot(t,z);
title("1 Hz Sine, Cosine and Square signals")
xlabel("Time");
ylabel("Amplitude");
legend(["Sin", "Cos", "Square"])
```

```
hold off;

% Calculate corr among all signals
D = zeros(N,3); % matrix having all signals
D(:,1) = x; % signals in columns
D(:,2) = y;
D(:,3) = z;
r = corrcoef(D); % corr using this func
disp(r)
```

Output:

```
    1.0000    0.0000    0.9003
    0.0000    1.0000    0.0057
    0.9003    0.0057    1.0000
```



Lab Activity 02: Cross correlate EEG signal with the range of sinusoidal frequencies (0.5 to 50Hz). Frequency increment should be 0.5 Hz. attach the plot and explain the working of code and results obtained.

```matlab
clear all;
close all;
clc;

load eeg_data;
N = length(eeg);
fs = 50;
t = [0:N-1]/fs;

figure(1);
plot(t,eeg);
title("EEG signal");
xlabel("Time");
ylabel("Amplitude");

% create a set of sin waves (1 to 25 Hz)
for i = 1:25
   f(i) = i; % used for plotting
   x = sin(2*pi*i*t);
   rxy = xcorr(x, eeg,"coeff");
   rmax(i) = max(rxy); % max corr with current sine wave
end

figure(2);
stem(f,rmax); % plot max corr with each sin wave
title("Max corr with Sine waves");
xlabel("Frequency (Hz)")
ylabel("Corr coef");
```

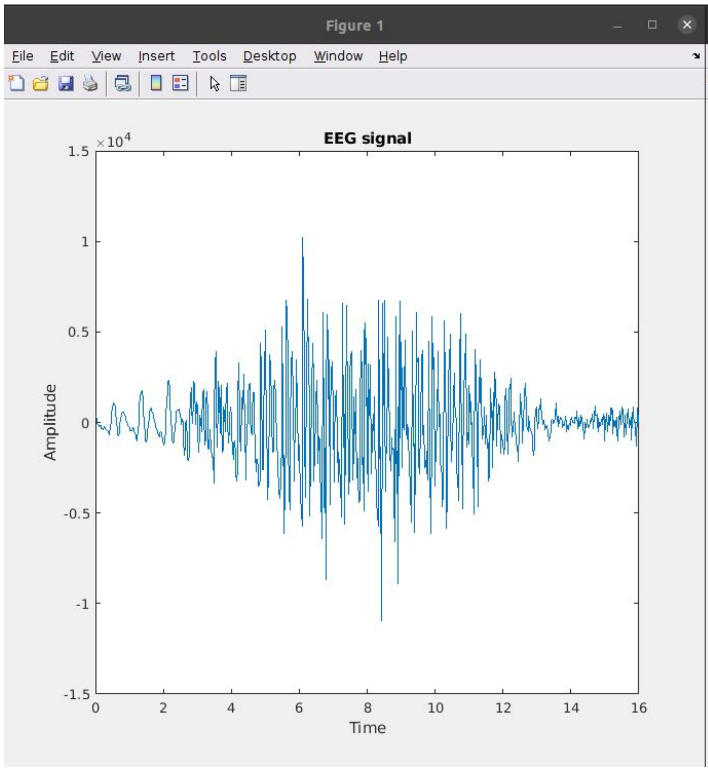


Figure 1
File Edit View Insert Tools Desktop Window Help
EEG signal
×10^4
1.5
1
0.5
0
-0.5
-1
-1.5
Amplitude
0 2 4 6 8 10 12 14 16
Time

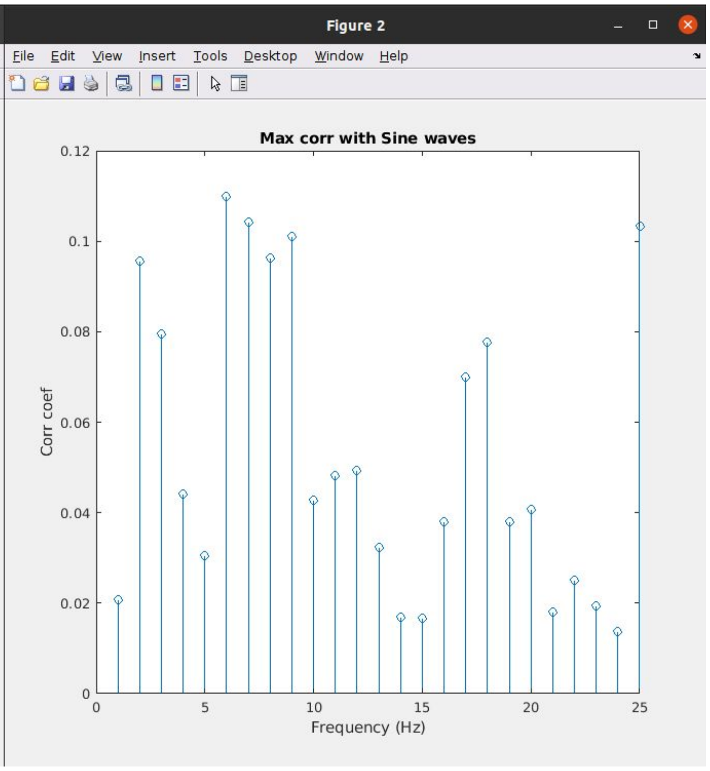


Figure 2
File Edit View Insert Tools Desktop Window Help
Max corr with Sine waves
0.12
0.1
0.08
0.06
0.04
0.02
0
Corr coef
0 5 10 15 20 25
Frequency (Hz)

Post Lab Task

1. Find the correlation between two subjects EEG signals recoded during mental arithmetic task. Download the signal from physioBank ATM. Attach the code and interpret the results.

```
clear all;
close all;
clc;

load Subject00_1_edfm

D = zeros(2,5000);
val = val';
D = val(:, 1:3); % correlation between 3 EEG leads
r = corrcoef(D); % corr using this func
disp(r)
```

Output:

```
1.0000   0.9442   0.9277
0.9442   1.0000   0.7885
0.9277   0.7885   1.0000
```

Interpretation:

- The results show that the lead 1 is highly correlated with lead 2 and 3.
- The lead 2 is less correlated with lead 3 than lead 2.
- Hence, lead 1 and 2 are very similar to each other, and lead 3 is different than lead 1 and 2.

Lab Session 06

Objective: To explain the principle of orthogonality between bio signals using covariance

Introduction

The relationship between correlation and covariance functions is similar to the relationship between standard correlation, Covariance and correlation functions are the same except that in covariance the means have been subtracted from the input signals, *x*(*t*) and *y*(*t*) (or just *x*(*t*) in the case of auto covariance). The auto covariance function can be thought of as measuring the memory or self-similarity of the deviation of a signal about its mean level. Similarly, the cross covariance function is a measure of the similarity of the deviation of two signals about their respective means.

MATLAB Function

Auto covariance or cross covariance is obtained using the **xcov** function:

[c,lags] = xcov(x, y, maxlags ,‘options’)

covariance matrices are calculated using the **cov** functions

S = cov(x), or S = cov(x,1);

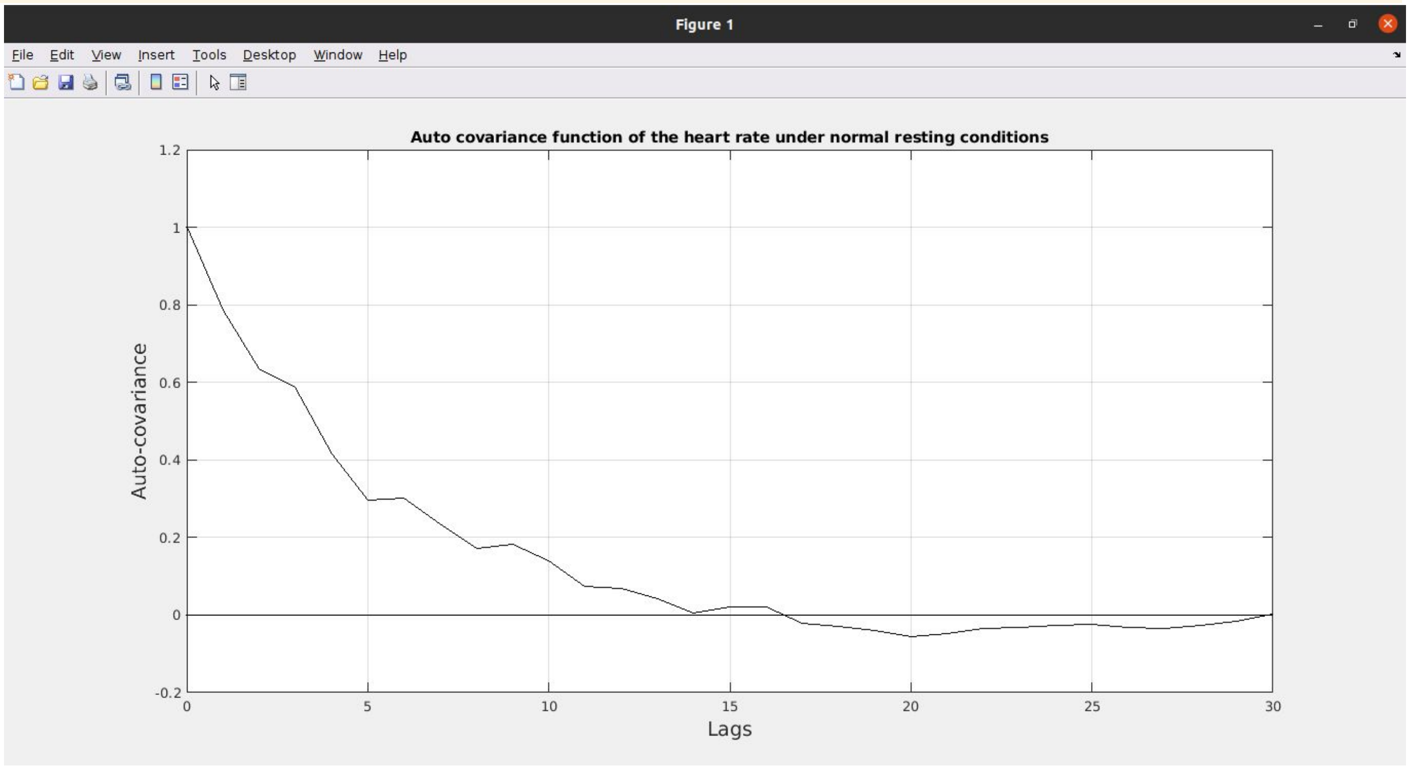
The covariance matrix assume that the multivariate data are arranged in a matrix, The covariance matrix gives the variance of the columns of the data matrix in the diagonals while the covariance between columns is given by the off diagonals:

$$S = \begin{bmatrix} \sigma_{1,1} & \sigma_{1,2} & \cdots & \sigma_{1,N} \\ \sigma_{2,1} & \sigma_{2,2} & \cdots & \sigma_{2,N} \\ \ldots & \ldots & O & \ldots \\ \sigma_{N,1} & \sigma_{N,2} & \cdots & \sigma_{N,N} \end{bmatrix}$$

Lab Activity: Determine if there is any correlation in the variation between the timing of successive heartbeats under normal resting conditions. Explain the working of code and also infer the results obtained.

```
clear all;
close all;
clc;
```

```
load Hr_pre; %Load normal HR data
[c,lags] = xcov(hr_pre,'coeff');
x=c';

% Plot and label data.
plot(lags,x,'k');
hold on
plot([lags(1), lags(end)], [0 0],'k'); % Plot a zero line
xlabel('Lags','FontSize',14);
ylabel('Auto-covariance','FontSize',14);
axis([0 30 -.2 1.2]);
title('Auto covariance function of the heart rate under normal resting
conditions');
grid on;
```



Lab Activity 02: Write a MATLAB code to determine 1Hz sine, 1 Hz cosine, 2 Hz cosine and 1 Hz sawtooth waveforms are orthogonal or not.

```
N = 100;
fs = 50;

t = [1:N-1]/fs;
x1 = sin(2*pi*t); % 1 hz sin wave
x2 = cos(2*pi*t); % 1 hz cos wave
x3 = cos(2*pi*2*t); % 2 hz cos wave
```

```
x4 = sawtooth(2*pi*t); % 1 hz sawtooth wave

figure(1);
subplot(2,2,1);
plot(t, x1);
xlabel("Time");
ylabel("Amplitude");
title("1 Hz Sin wave");

subplot(2,2,2);
plot(t, x2);
xlabel("Time");
ylabel("Amplitude");
title("1 Hz Cos wave");

subplot(2,2,3);
plot(t, x3);
xlabel("Time");
ylabel("Amplitude");
title("2 Hz Cos wave");

subplot(2,2,4);
plot(t, x4);
xlabel("Time");
ylabel("Amplitude");
title("1 Hz Sawtooth wave");

X = [x1; x2; x3; x4];
S = cov(X);
```

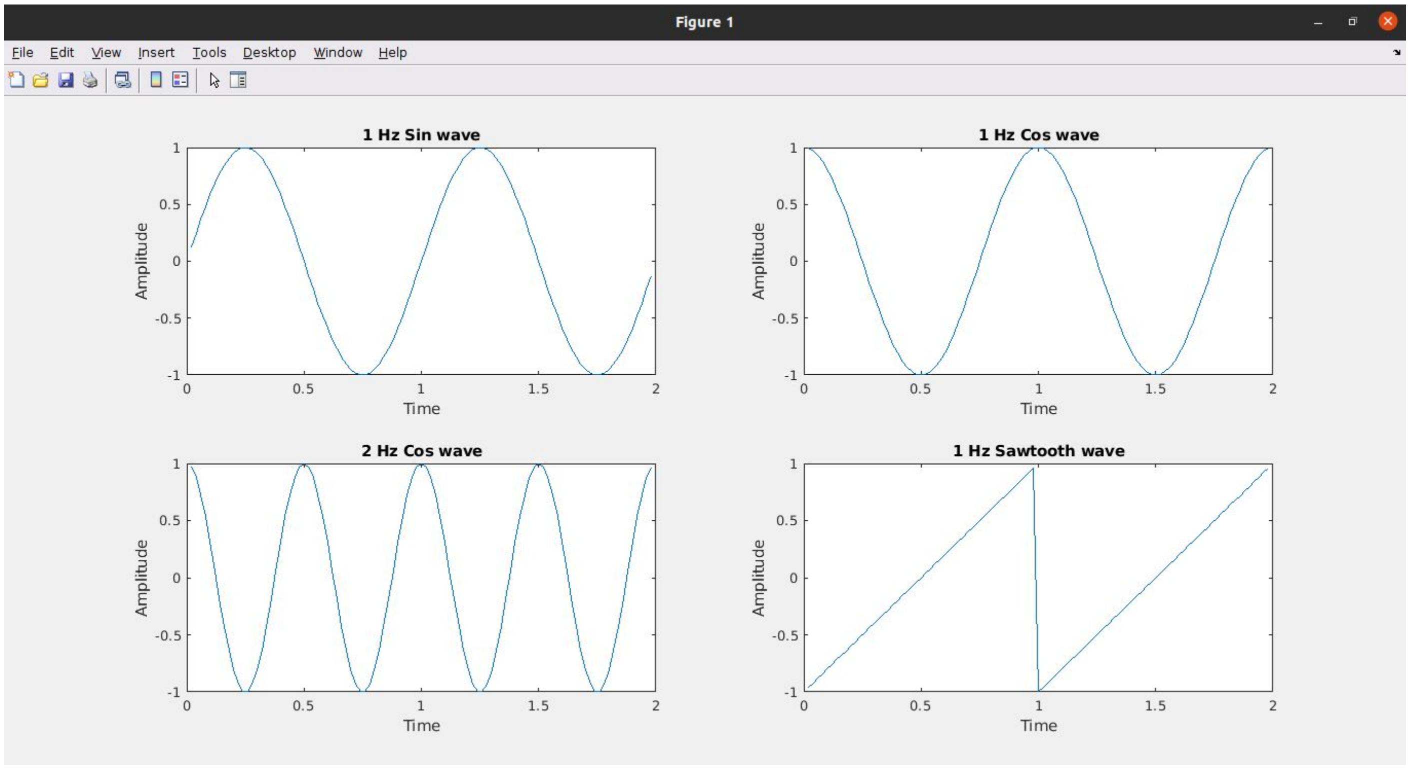
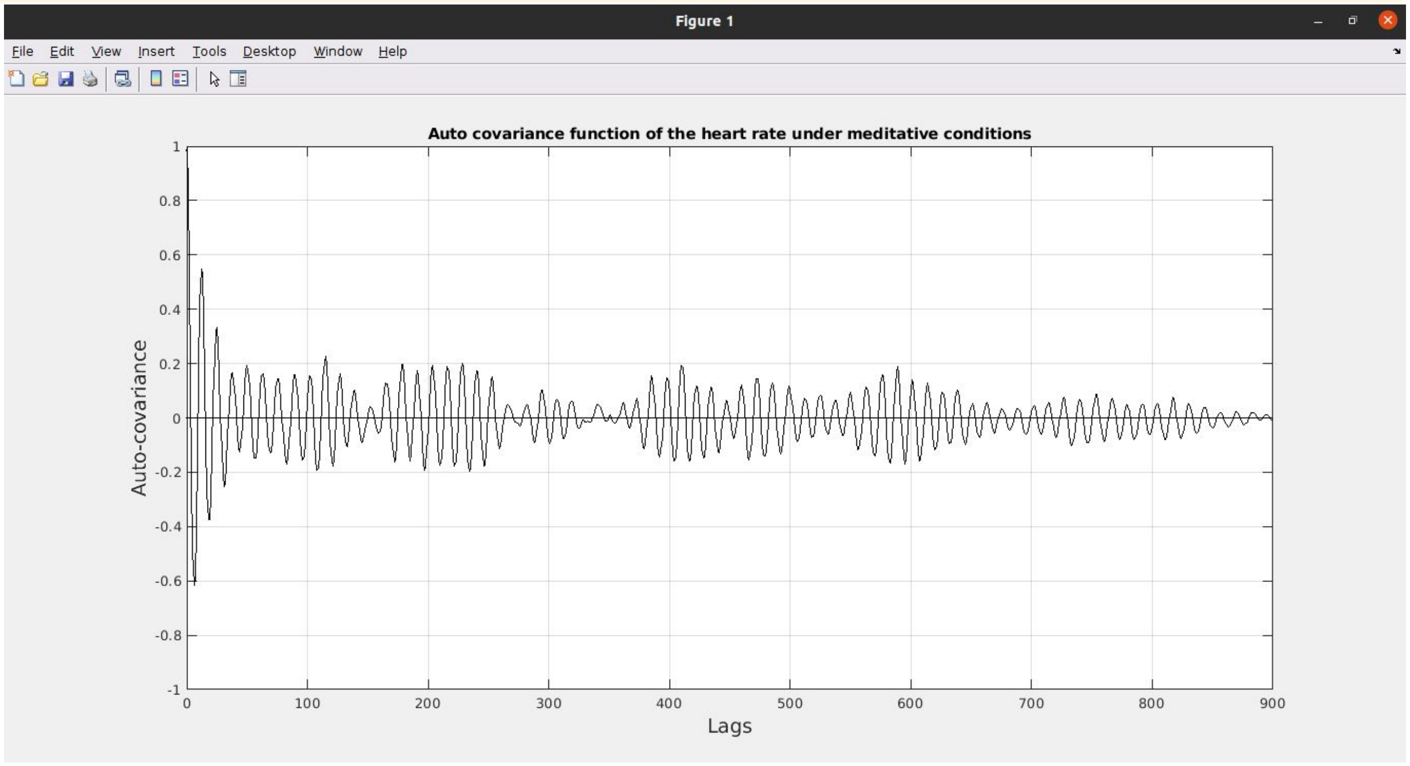


Figure 1
File Edit View Insert Tools Desktop Window Help
1 Hz Sin wave
1 Hz Cos wave
2 Hz Cos wave
1 Hz Sawtooth wave
Amplitude
Time
1
0.5
0
-0.5
-1
0
0.5
1
1.5
2

Post Lab Task

1. Determine if there is any correlation in the variation between the timing of successive heartbeats under meditative condition. Compare your results with the results obtained in lab activity 01.

```
load Hr_med; %Load normal HR data
[c,lags] = xcov(hr_med,'coeff');
x=c';

% Plot and label data.
plot(lags,x,'k');
hold on
plot([lags(1), lags(end)], [0 0],'k'); % Plot a zero line
xlabel('Lags','FontSize',14);
ylabel('Auto-covariance','FontSize',14);
axis([0 900 -1 1]);
title('Auto covariance function of the heart rate under meditative conditions');
grid on;
```



Lab Session 07

Objective: Study and implement methods for frequency-domain analysis of biomedical signals.

Introduction

Both analog and digital signals can be represented in either the time or the frequency domain. The frequency-domain representation is an equally valid representation of a signal and consists of two components: magnitude and phase. The components show how the signal energy is distributed over a range of frequencies. Sometimes the spectral representation of a signal is more understandable than the time-domain representation. Magnitude and phase plots are necessary to represent the signal completely and to convert the frequency representation back into a time representation. To represent the signal, in frequency domain, the classical Fourier transform (FT) method is the most straightforward.

MATLAB Implementation

MATLAB provides a variety of methods for calculating spectra, particularly if the Signal Processing Toolbox is available. The basic Fourier transform routine is implemented as:

Xf = fft(x,n)   % Calculate the Fourier Transform

where x is the input waveform and Xf is a complex vector providing the sinusoidal coefficients.

The magnitude of the frequency spectra is obtained by applying the absolute value function, abs, to the complex output Xf:

Magnitude = abs(Xf);   % Take the magnitude of Xf

The phase angle of the spectra can be obtained by application of the MATLAB angle function: Phase =

angle(Xf) % Find the angle of Xf.

The angle function takes the arctangent of the imaginary part divided by the real part of Xf.

Lab Activity 01: Use fft to find the magnitude spectrum of a waveform consisting of a single 250- Hz sine wave and white noise with an SNR of −14 db. Calculate the Fourier transform of this waveform and plot the magnitude spectrum. Explain the working of sig_noise function. Attach the plot and write your observation.

```
clear all;
close all;
clc;

N = 1024; % no. of data points
```

```
f = 50;
fs = 1000;
t = [0:N-1]/fs;
x = sin(2*pi*t*f) + rand(size(t));
% [x,t] = sig_noise(250, -14, N); % this file is not available; add 250 db
noise to a data of length N, the SNR = -14

figure(1);
plot(t,x);
title("Noisy signal in time domain");
xlabel("Time");
ylabel("Amplitude");

Xf = fft(x); % transform to freq domain
mag = abs(Xf);
f = (1:N)*fs/N;
figure(2);
plot(f,mag);
title("Noisy signal in Frequency domain");
xlabel("Frequency (Hz)");
ylabel("Amplitude");
```

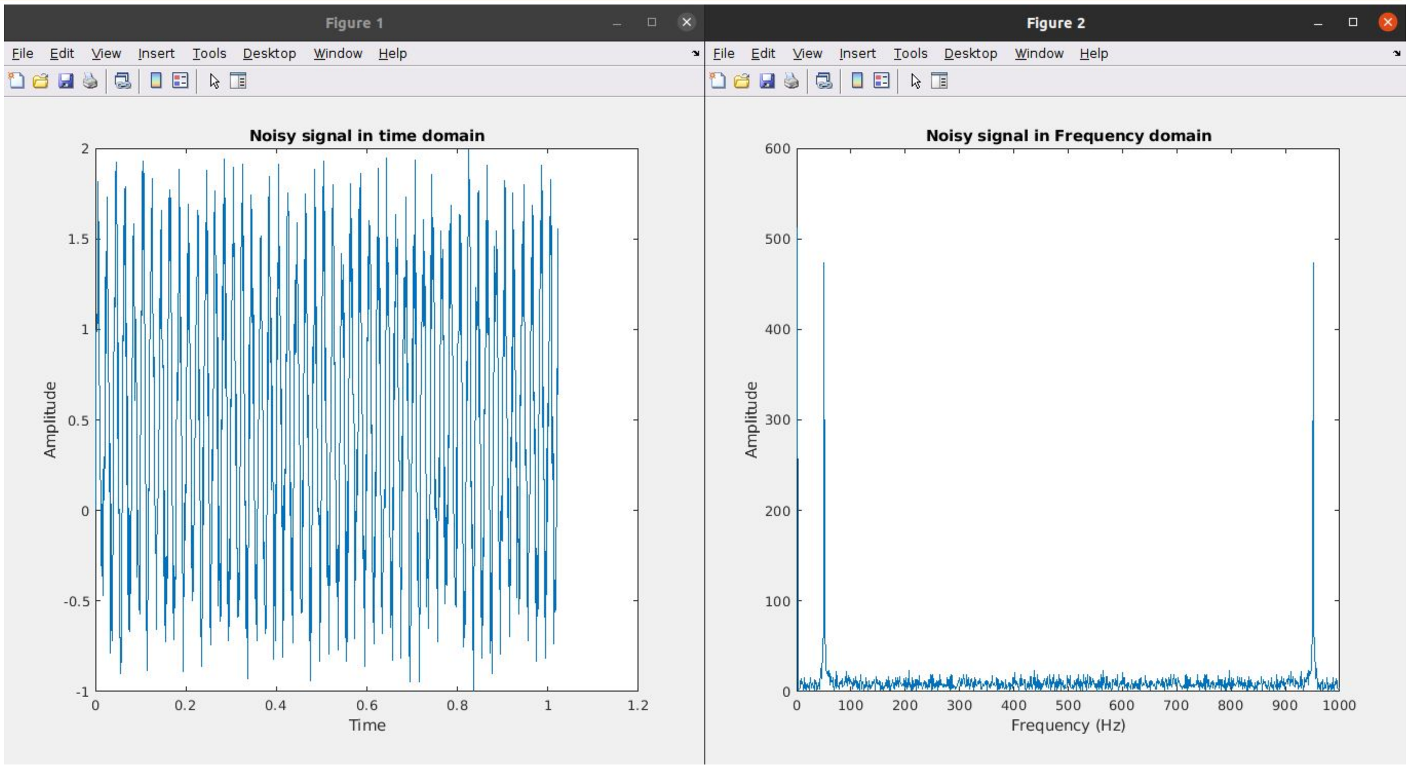
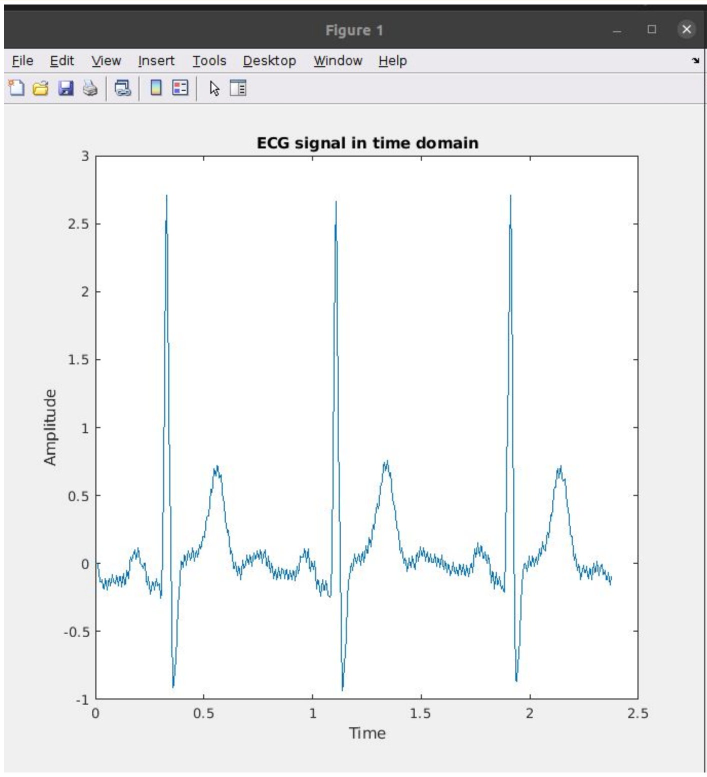
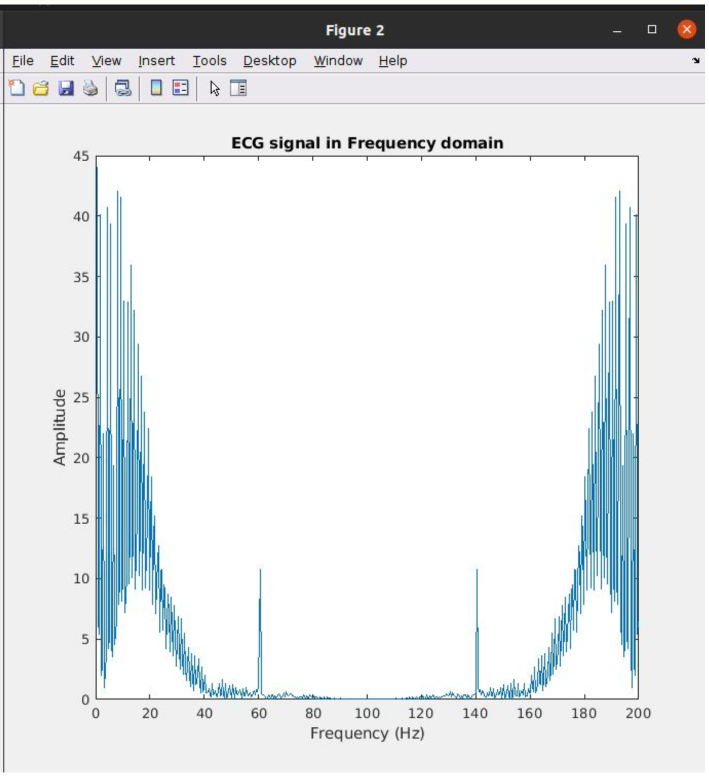


Figure 1
File Edit View Insert Tools Desktop Window Help
Noisy signal in time domain
Amplitude
Time
Figure 2
File Edit View Insert Tools Desktop Window Help
Noisy signal in Frequency domain
Amplitude
Frequency (Hz)

Lab Activity 02: Download ECG signals from MIT-BIH Arrhythmia Database (physionet database) Compute the magnitude spectrum of EEG signal. Also attach the results with explanation**.**

```
clear all;
close all;
clc;

load("ecg.dat");
fs = 200;
N = length(ecg);
ts = 1/fs; % sampling interval
t = 0:ts:(N/fs)-ts;

figure(1);
plot(t, ecg);
title("ECG signal in time domain");
xlabel("Time");
ylabel("Amplitude");

Xf = fft(ecg);
mag = abs(Xf);
freq = (1:N)*fs/N;
figure(2);
plot(freq, mag);
title("ECG signal in Frequency domain");
xlabel("Frequency (Hz)");
ylabel("Amplitude");
```





Lab Session 08

Objective: Estimation of Power spectral Density of bio signal

Introduction:

The power spectrum is commonly defined as the Fourier transform of the auto-correlation function. A more popular method for evaluating the power spectrum is the *direct approach*. In the "direct approach," the power spectrum is calculated as the magnitude squared of the Fourier transform (or Fourier series) of the waveform of interest. Unlike the Fourier transform, the power spectrum does not contain phase information, so the power spectrum is not an invertible transformation: It is not possible to reconstruct the signal from the power spectrum. However, the power spectrum has a wider range of applicability and can be defined for some signals that do not have a meaningful Fourier transform (such as those resulting from random processes). Since the power spectrum does not contain phase information, it is applied in situations where phase is not considered useful or to data that contain a lot of noise, since phase information is easily corrupted by noise.

In order to obtain an estimate of the PSD of a signal, or a segment of interest of the signal, follow the procedure given below.

- Load the Signal of interest
- Obtain the Fourier transform (FT) of the segment. The Mat lab command for the computation of the FT is fft (representing the fast Fourier transform or FFT algorithm).
- Obtain the squared magnitude of the FT.
- Normalize the squared magnitude spectrum by dividing by its maximum.
- Convert the result to dB: take the logarithm to base ten of the result above and multiply by ten. This is an estimate of the PSD of the original signal in dB.
- Select the first half of the PSD as above. Prepare the corresponding frequency axis in Hz, spanning the range [0, $f_s/2$], where f_s is the sampling frequency of the signal.
- Plot and analyze the result. Ensure that the axes are labeled in Hz and dB.

Lab Activity 01: Generate sum of two signals of equal magnitude in MATLAB. The frequency of signals should be 30 Hz and 50 Hz. The sampling frequency should be 1000Hz. Plot each of the signal and the resultant sum. Then compute the power spectrum of the sum of signal using above steps. Attach the plots and write your observation.

```
clear all;
close all;
clc;

f1 = 30;
f2 = 50;
fs = 1000;
N = 200;
```

```
t = [0:N-1]/fs;

sig1 = sin(2*pi*f1*t);
sig2 = sin(2*pi*f2*t);

figure(1)
subplot(2,1,1);
plot(t, sig1);
title("30 Hz Sine wave")
xlabel("Time");
ylabel("Amplitude");
subplot(2,1,2);
plot(t, sig2);
title("50 Hz Sine wave")
xlabel("Time");
ylabel("Amplitude");

summed_sig = sig1 + sig2;
figure(2);
subplot(2,1,1);
plot(t,summed_sig);
title("Summed Sine wave (30Hz + 50) Hz");
xlabel("Time");
ylabel("Amplitude");
hold on

freq = fft(summed_sig);
freq_pwr = abs(freq(1:N/2)).^2;
freq_pwr_normalized = freq_pwr/max(freq_pwr);
freq_db = log10(freq_pwr_normalized)*10;
freq_scale = (1:N/2)*fs/N;
subplot(2,1,2);
plot(freq_scale,freq_db);
title("Normalized frequency spectrum of summed signal");
xlabel("Frequency");
ylabel("dB");
```

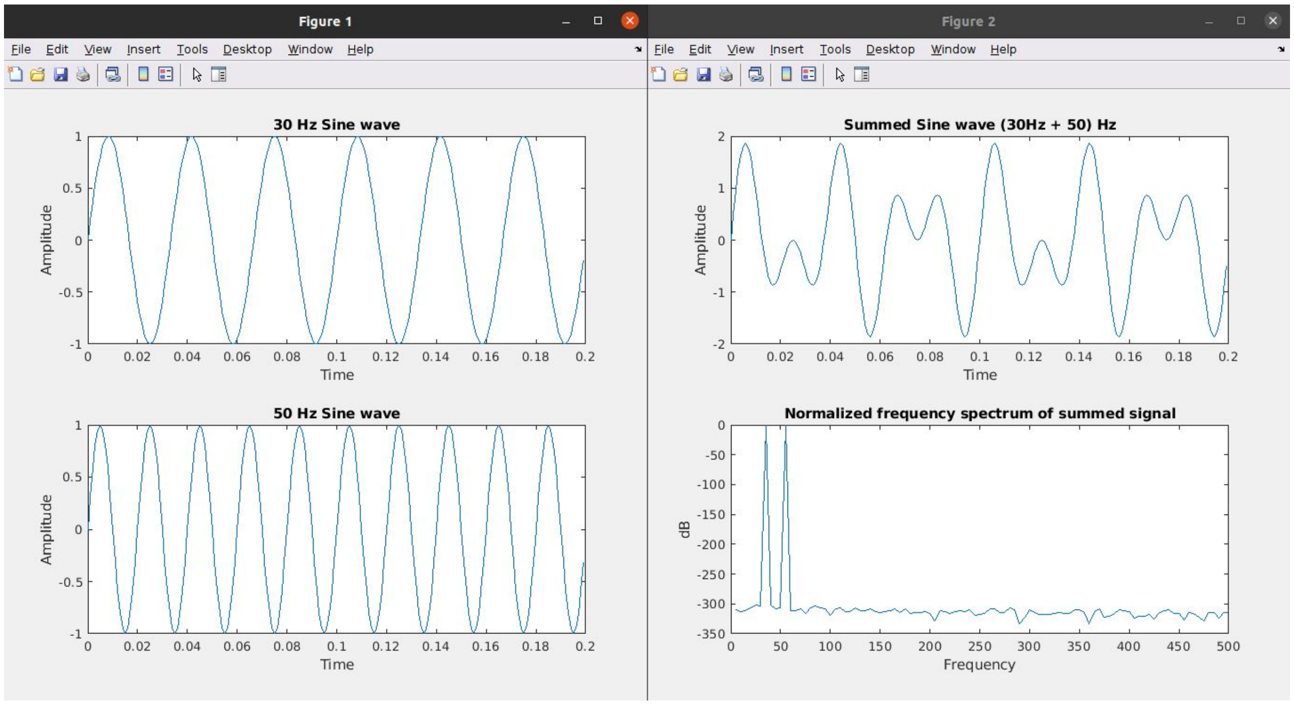


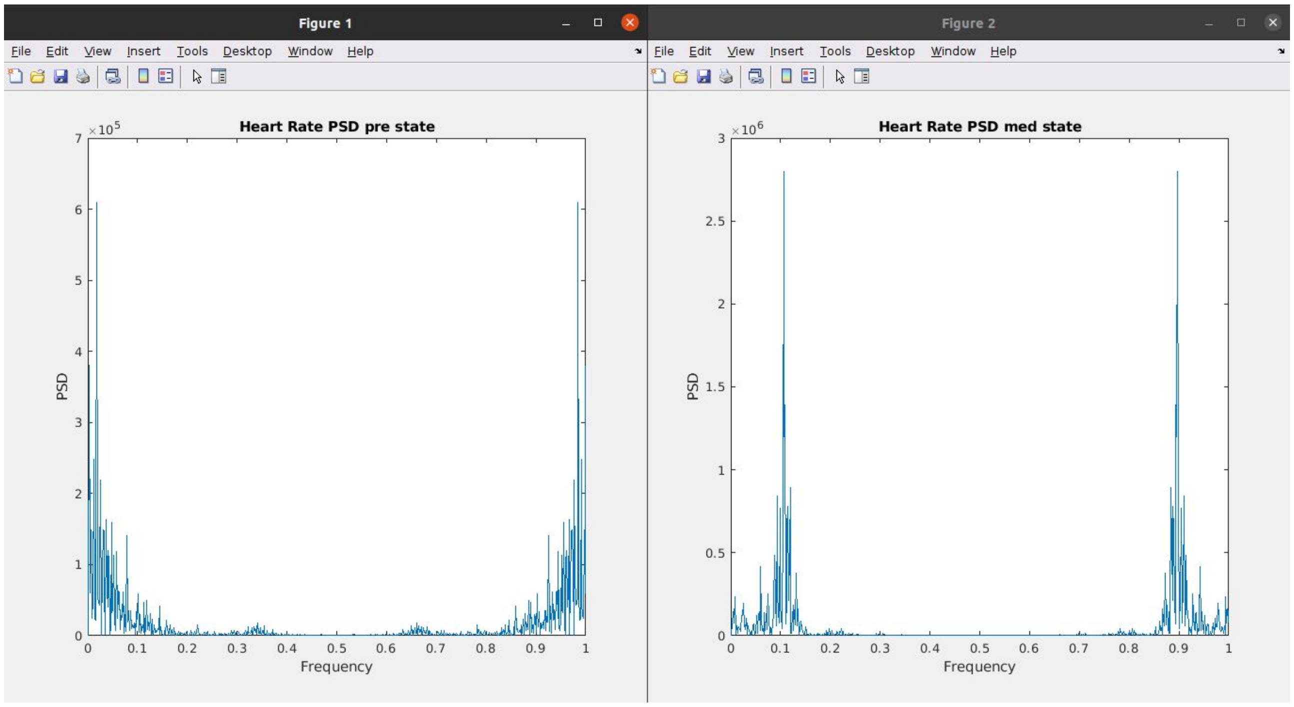
Figure 1
File Edit View Insert Tools Desktop Window Help
30 Hz Sine wave
50 Hz Sine wave
Amplitude
Time
Figure 2
Summed Sine wave (30Hz + 50) Hz
Normalized frequency spectrum of summed signal
dB
Frequency

Lab Activity 02: Determine and plot the power spectra of heart rate variability data recorded during both normal states. The heart rate data should first be converted to evenly sampled time data. Time interval of data needs to be rearranged into evenly spaced time positions. This process, known as resampling, will be done through interpolation using MATLAB's interp1 routine.

```
clear all;
close all;
clc;

load("Hr_pre.mat"); % load datasets
load("Hr_med.mat");

fs = 1; % 1 samples/sec

% pre med recording
xi_pre = ceil(t_pre(1):fs:floor(t_pre(end))); % evenly spaced vector of
desired datapoints
% resampling using interp1
yi_pre = interp1(t_pre,hr_pre, xi_pre); % linear interpolation (using evenly
spaced vectors), t_pre and hr_pre used for interpolation, and xi_pre = desired
data points.
YI_pre = yi_pre - mean(yi_pre); % heart rate - avg_HR
Y_freq_pre = fft(YI_pre); % step 1 for PSD
Y_mag_pre = abs(Y_freq_pre);
PSD_pre = (Y_mag_pre.^2); % step 2 for PSD
N_pre = length(PSD_pre); % used for plotting
freq_pre = (1:N_pre)*fs/N_pre;
figure(1);
plot(freq_pre, PSD_pre);
```

```
title("Heart Rate PSD pre state")
xlabel("Frequency");
ylabel("PSD");

% post med recording
xi_med = ceil(t_med(1):fs:floor(t_med(end))); % evenly spaced vector of
desired datapoints
yi_med = interp1(t_med,hr_med, xi_med); % linear interpolation (using evenly
spaced vectors), t_pre and % hr_pre used for interpolation, and xi_pre =
desired data points.
YI_med = yi_med - mean(yi_med); % heart rate - avg_HR
Y_freq_med = fft(YI_med); % step 1 for PSD
Y_mag_med = abs(Y_freq_med);
PSD_med = (Y_mag_med.^2); % step 2 for PSD
N_med = length(PSD_med); % used for plotting
freq_med = (1:N_med)*fs/N_med;
figure(2);
plot(freq_med, PSD_med);
title("Heart Rate PSD med state")
xlabel("Frequency");
ylabel("PSD");
```

Post Lab Tasks



1. What are the advantages of power spectral density estimation of bio signal? Support your answer by giving an example of PSD of bio signal.
   Power spectral density function is a very useful tool if you want to identify oscillatory signals in

your time series data and want to know their amplitude. Power spectral density tells us at which frequency ranges variations are strong and that might be quite useful for further analysis.

2. Plot the power spectrum of heart rate variability during meditative state. Analyze both plots and interpret the results. What is the difference at 0.1 Hz frequency in both plots?

The plot is shown above. The pre-meditative state shows a very low of insignificant PSD at 0.1 Hz frequency, whereas the meditative state shows peak at 0.1 Hz frequency.

Lab Session 9

Objective: To learn the method of windowing a signal and application of windowing in signal processing

Introduction:

When bio signal's segment is simply cut out from the overall waveform; that is, a portion of the waveform is truncated and stored, without modification, in the computer. This is equivalent to the application of a rectangular *window* to the overall waveform, and the analysis is restricted to the *windowed* portion of the waveform. Window shapes other than rectangular are possible simply by multiplying the waveform by the desired shape (sometimes these shapes are referred to as tapering functions). Again, points outside the window are assumed to be zero even if it is not true.

Window Functions in MATLAB

MATLAB has a number of data windows. The relevant MATLAB routine generates an n-point vector array containing the appropriate window shape. All have the same form:

```
S= window_name(N);
```

Where N is the number of points in the output vector and window_name is the name, or an abbreviation of the name, of the desired window. A few of the more popular windows are: bartlett, blackman, gausswin, hamming (a common MATLAB default window), hann, kaiser, and triang. To apply a window to the Fourier series analysis simply point-by-point multiply the digitized waveform by the output of the MATLAB window_name routine before calling the FFT routine. For example:

```
w=triang(N);
x=   x.*w';        %Multiply(point-by-point)data by window
X = fft(x);        %CalculateFFT
```

Graphical User Interface Tools For Windows

Two graphical user interface tools are provided for working with windows in the Signal Processing toolbox:

- Window Design and Analysis Tool (wintool)
- Window Visualization Tool (wvtool)

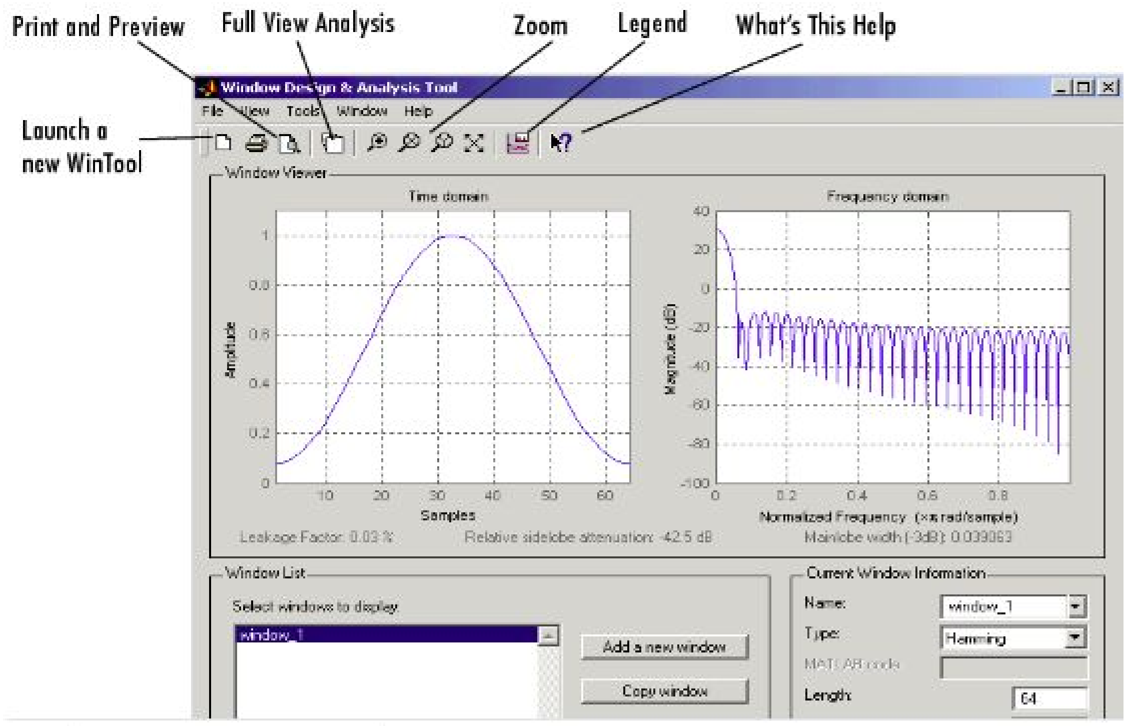
<u>Wintool :</u> Window Design and Analysis Tool (WinTool)

Syntax

- wintool
- wintool(obj1,obj2,...)

Description

wintool opens WinTool, a graphical user interface (GUI) for designing and analyzing spectral windows. It opens with a default 64-point Hamming window.



Leakage Factor: 0.03 %
Relative sidelobe attenuation: -42.5 dB
Mainlobe width (-3dB): 0.039063

Name: window_1
Type: Hamming
Length: 64

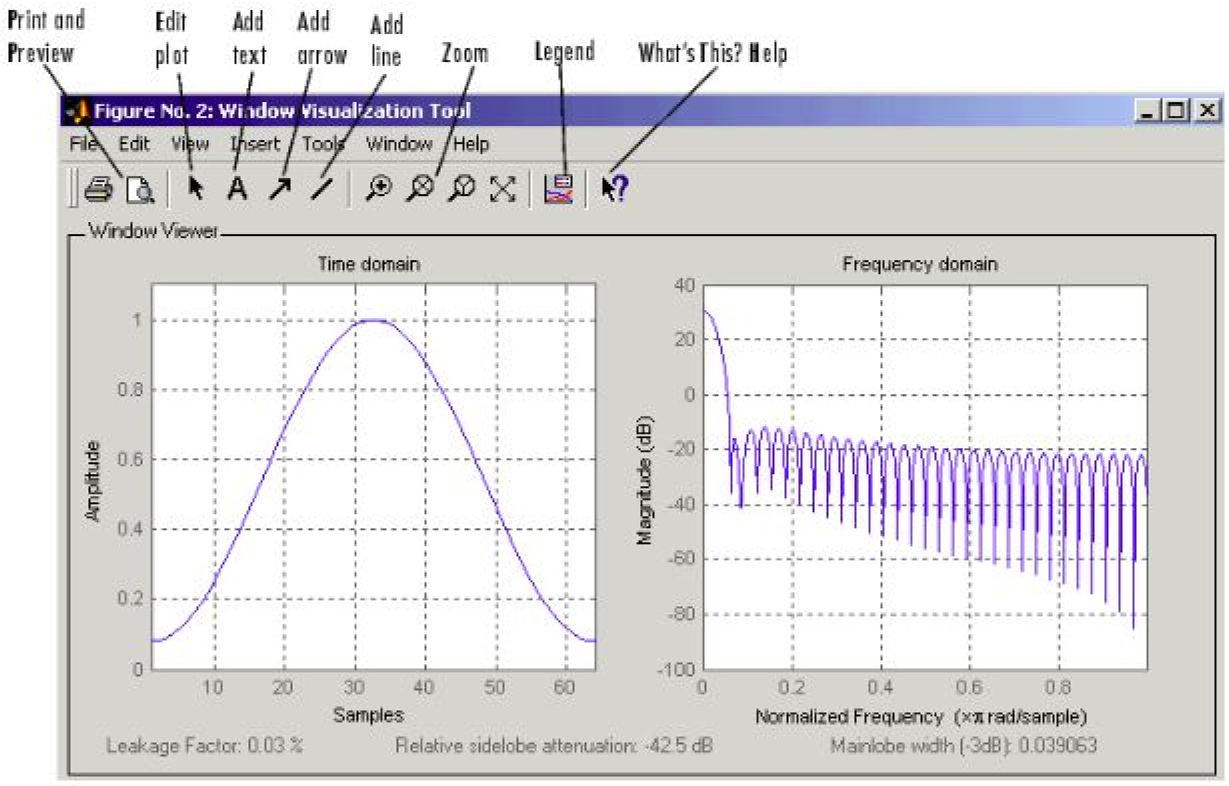
<u>Wvtool:</u> Window Visualization Tool

Syntax

- wvtool(winname(n))
- wvtool(winname$_1$(n),winname$_2$(n),...winname$_m$(n))
- wvtool(w)
- h = wvtool(...)

Description

wvtool(winname(n)) opens WVTool with the time and frequency domain plots of the n-length window specified in winname, which can be any window in the Signal Processing Toolbox.

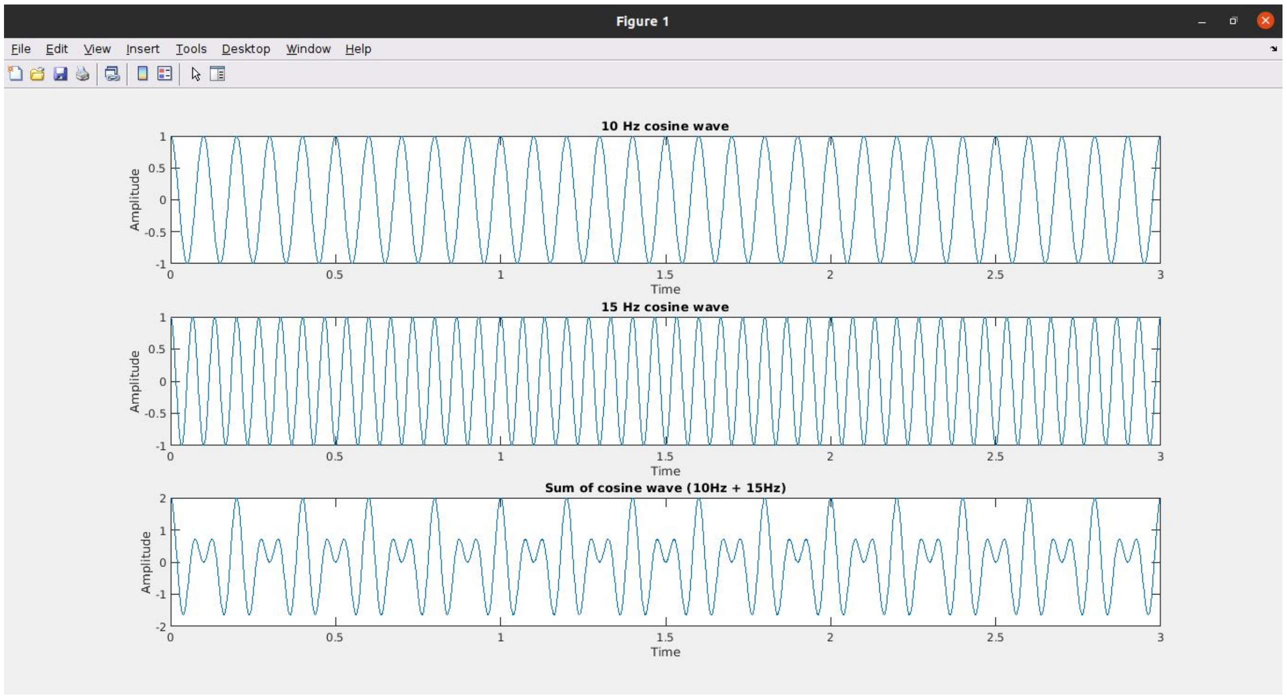


Leakage Factor: 0.03 %
Relative sidelobe attenuation: -42.5 dB
Mainlobe width (-3dB): 0.039063

Lab Activity 01: Write a MATLAB program to generate a sum of cosine waves of frequency 10Hz and 15Hz and number of samples should be 1000. Apply a rectangular and hamming window on the sum of signals. Plot actual signal and windowed signal. Observe the effect of changing the input N to the window function. Compute the power spectrum of windowed signals. Attach the plots and write your observation.

```
close all;
clear all;
clc;

f1 = 10;
f2 = 15;
fs = 1000;
t = 0:1/fs:3;
sig1 = cos(2*pi*f1*t);
sig2 = cos(2*pi*f2*t);
sig_summed = sig1 + sig2;

figure(1);
subplot(311);
plot(t, sig1);
xlabel("Time");
```

```
ylabel("Amplitude");
title("10 Hz cosine wave");

subplot(312);
plot(t, sig2);
xlabel("Time");
ylabel("Amplitude");
title("15 Hz cosine wave");

subplot(313);
plot(t, sig_summed);
xlabel("Time");
ylabel("Amplitude");
title("Sum of cosine wave (10Hz + 15Hz)");
```



```
% create window
N = 250;
rect_win = rectwin(N);
rect_win = ([zeros(1,25), rect_win', zeros(1,25)]); % zeros padding (outside
window)

hann_win = hann(N);
hann_win = ([zeros(1,25), hann_win', zeros(1,25)]);

figure(2);
plot(hann_win, "r");
hold on;
```

```
plot(rect_win, "g");
legend(["Hanning Window", "Rectangular Window"]);

% freq domain
hann_fft = fftshift(fft(hann_win)); % fftshift shifts scale to -fs/2 to fs/2
rect_fft = fftshift(fft(rect_win));

freq = (-150:150-1)*fs/300; % samples =  25 + 250 + 25 = 300
hann_ss = 10*log10(abs(hann_fft));
rect_ss = 10*log10(abs(rect_fft));

figure(3);
plot(freq, hann_ss,"b");
hold on;
plot(freq, rect_ss, "r");
legend(["FFT of hanning window", "FFT of rectangular window"])
xlabel("Frequency");
ylabel("Amplitude");
```

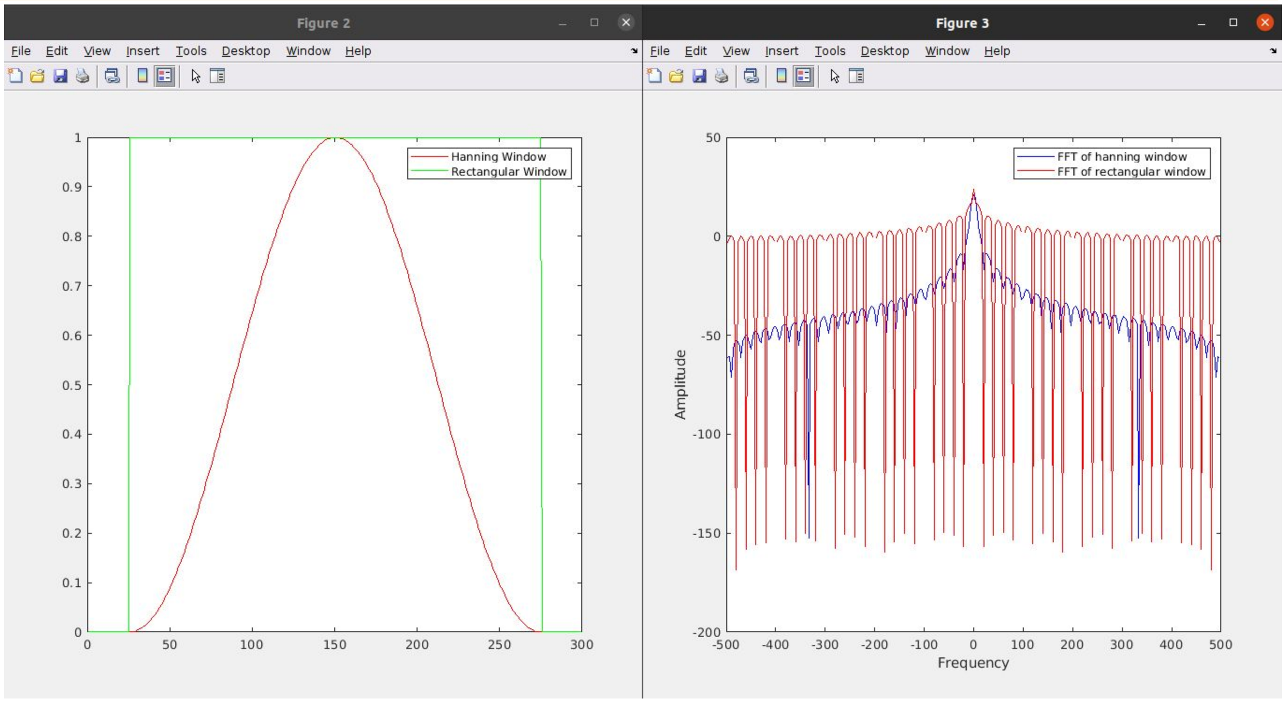
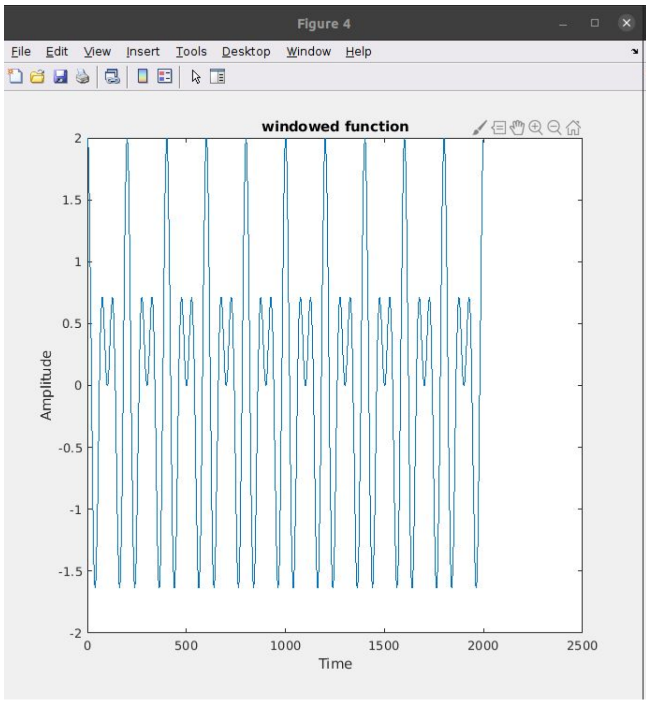


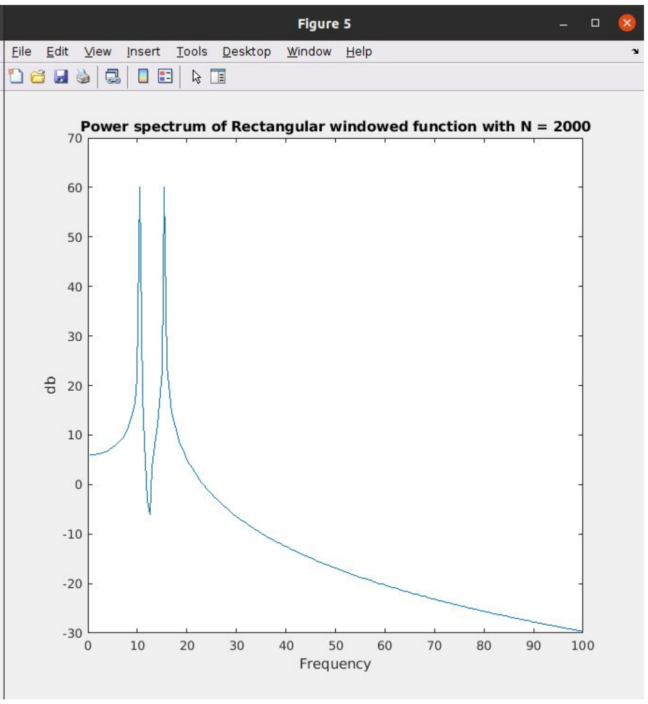
Figure 2
File Edit View Insert Tools Desktop Window Help
Hanning Window
Rectangular Window
Figure 3
File Edit View Insert Tools Desktop Window Help
FFT of hanning window
FFT of rectangular window
Amplitude
Frequency

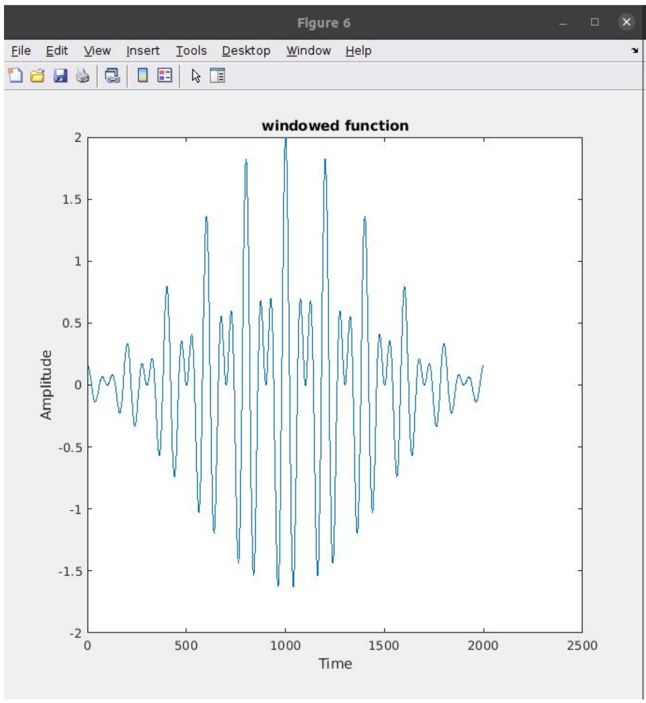
Window size = 3000

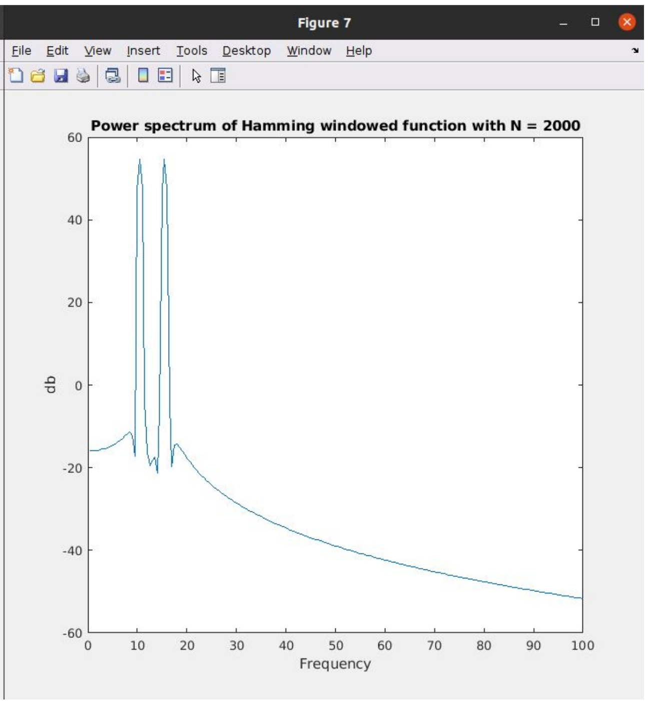
```
% apply rect window of size 2000
M = 2001; % window size
Rect_win = rectwin(M);
signal_rect = sig_summed(1:M).*Rect_win'; % dot product
figure(4);
plot(signal_rect);
title("windowed function");
xlabel("Time");
```

```matlab
ylabel("Amplitude");
fft1_mag = abs(fft(signal_rect));
ps1 = (fft1_mag).^2;
freq_axis = (1:M)*fs/M;
figure(5);
plot(freq_axis, 10*log10(ps1));
title("Power spectrum of Rectangular windowed function with N = 2000");
xlabel("Frequency");
ylabel("db");
xlim([0 100]);


% apply hamm window of size 2000
M = 2001; % window size
hamm_win = hamming(M);
signal_hamm = sig_summed(1:M).*hamm_win'; % dot product
figure(6);
plot(signal_hamm);
title("windowed function");
xlabel("Time");
ylabel("Amplitude");
fft1_mag = abs(fft(signal_hamm));
ps1 = (fft1_mag).^2;
freq_axis = (1:M)*fs/M;
figure(7);
plot(freq_axis, 10*log10(ps1));
title("Power spectrum of Hamming windowed function with N = 2000");
xlabel("Frequency");
ylabel("db");
xlim([0 100]);
```





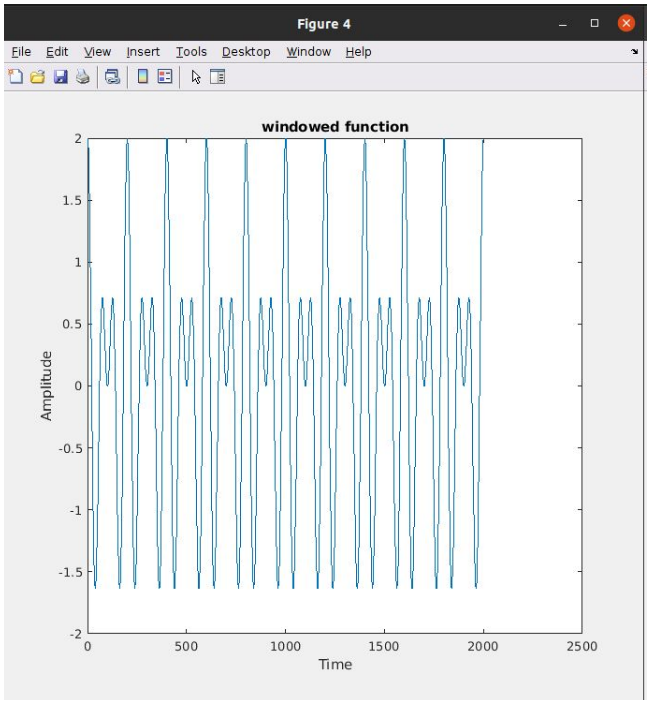


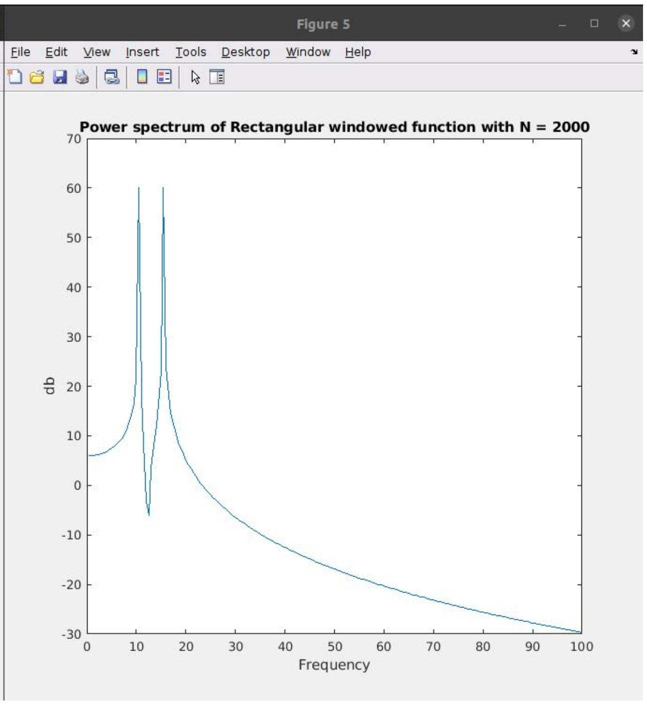


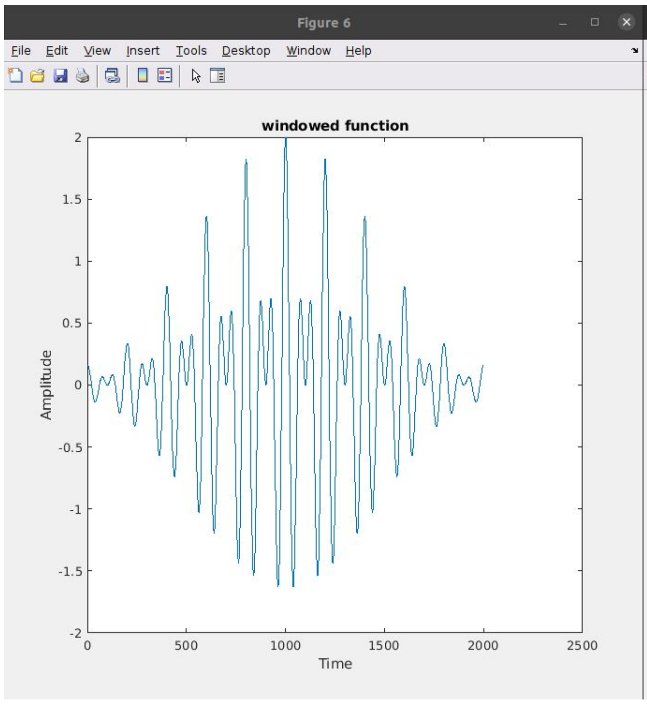
Window size = 3000

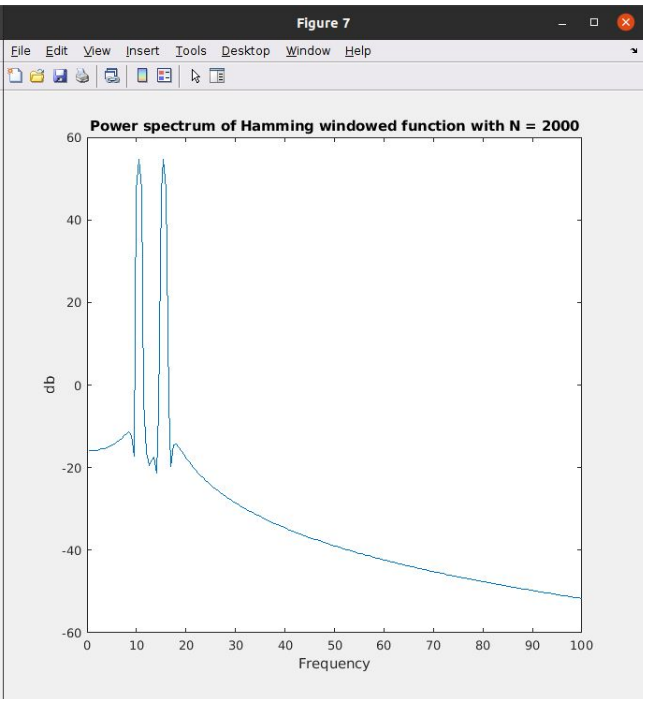
```
% apply rect window of size 3000
M = 3000; % window size
Rect_win = rectwin(M);
signal_rect = sig_summed(1:M).*Rect_win'; % dot product
figure(8);
plot(signal_rect);
title("windowed function");
xlabel("Time");
ylabel("Amplitude");
```

```
fft1_mag = abs(fft(signal_rect));
ps1 = (fft1_mag).^2;
freq_axis = (1:M)*fs/M;
figure(9);
plot(freq_axis, 10*log10(ps1));
title("Power spectrum of Rectangular windowed function with N = 2000");
xlabel("Frequency");
ylabel("db");
xlim([0 100]);


% apply hamm window of size 3000
M = 3000; % window size
hamm_win = hamming(M);
signal_hamm = sig_summed(1:M).*hamm_win'; % dot product
figure(10);
plot(signal_hamm);
title("windowed function");
xlabel("Time");
ylabel("Amplitude");
fft1_mag = abs(fft(signal_hamm));
ps1 = (fft1_mag).^2;
freq_axis = (1:M)*fs/M;
figure(11);
plot(freq_axis, 10*log10(ps1));
title("Power spectrum of Hamming windowed function with N = 2000");
xlabel("Frequency");
ylabel("db");
xlim([0 100]);
```









Observations:

- Larger window provides better frequency resolution.
- Ideally, the power spectrum should have two sharp peaks only at 10Hz and 15Hz. However, the side lobes are visible due to spectral leakage.
- The main lobe for the rectangular window is narrower than for the hamming window.

Post Lab Task

1. What is side lobe attenuation in window frequency response?
2. Write a MATLAB program to apply rectangular window and hamming window on the segment of EEG signal. Attach the plots.

```
load ecg.dat

t = 0:length(ecg)-1;
figure(1);
plot(t, ecg);
xlabel("Samples");
ylabel("Amplitude");
title("ECG waveform");

M = 200; % window size
hamm_win = hamming(M);
ecg_hamm = ecg(1:M).*hamm_win'; % dot product
figure(2);
plot(ecg_hamm);
title("Hamming window applied on ECG");
xlabel("Time");
ylabel("Amplitude");

M = 200; % window size
rect_win = rectwin(M);
ecg_rect = ecg(1:M).*rect_win'; % dot product
figure(3);
plot(ecg_rect);
title("Rectangular window applied on ECG");
xlabel("Time");
```

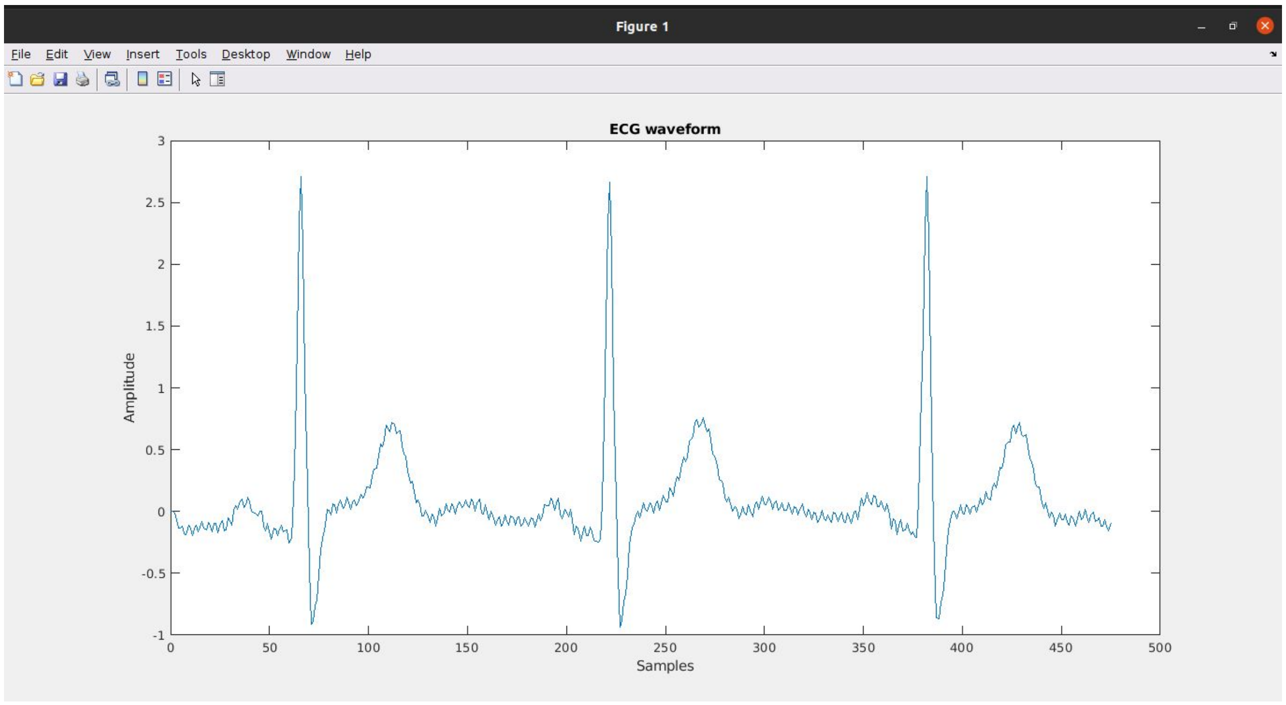


Figure 1
File Edit View Insert Tools Desktop Window Help
ECG waveform
Amplitude
Samples
3
2.5
2
1.5
1
0.5
0
-0.5
-1
0 50 100 150 200 250 300 350 400 450 500

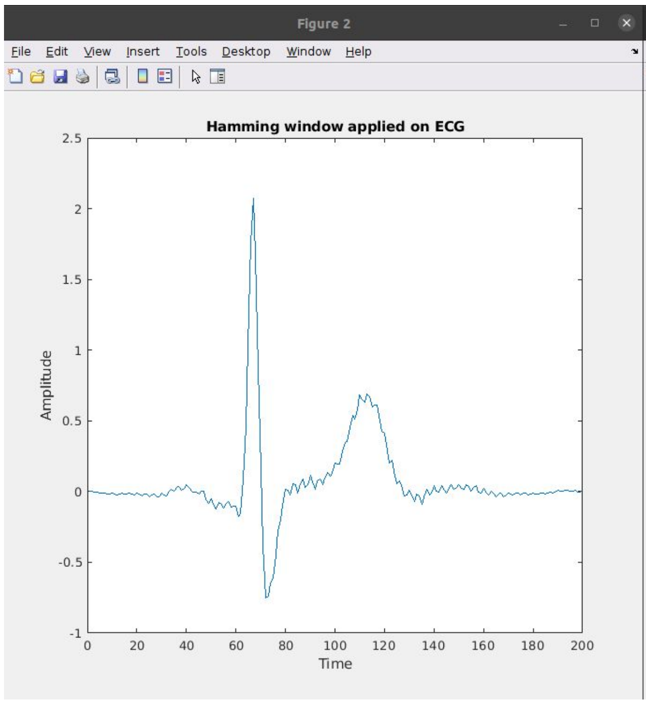


Figure 2
File Edit View Insert Tools Desktop Window Help
Hamming window applied on ECG
Amplitude
Time
2.5
2
1.5
1
0.5
0
-0.5
-1
0 20 40 60 80 100 120 140 160 180 200

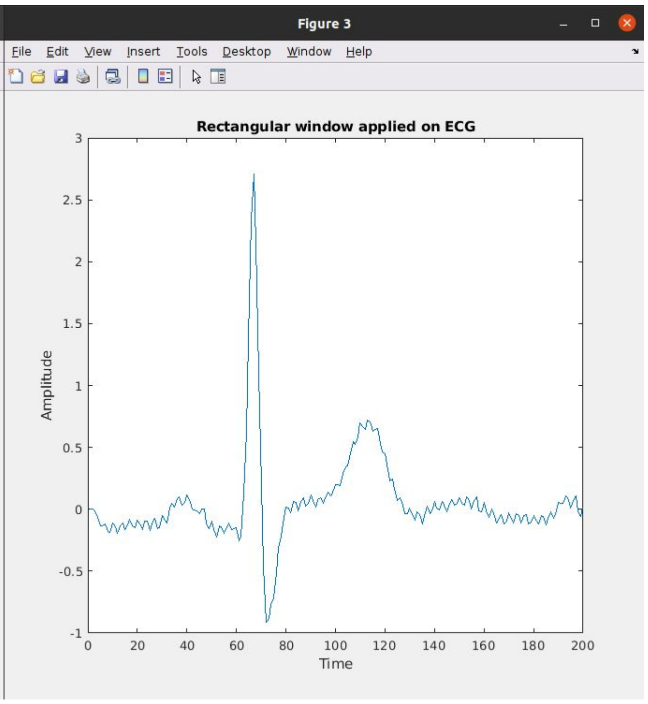


Figure 3
File Edit View Insert Tools Desktop Window Help
Rectangular window applied on ECG
Amplitude
Time
3
2.5
2
1.5
1
0.5
0
-0.5
-1
0 20 40 60 80 100 120 140 160 180 200

Lab Session 10

Objective: To get familiar with the method of designing and implementation of FIR filter in MATLAB

Introduction:

A basic filter can be viewed as a linear process in which the input signal's spectrum is reshaped in some well-defined (and, one hopes, beneficial) manner. Filters differ in the way they achieve this spectral reshaping, and can be classified into two groups based on their approach. These two groups are termed finite impulse response (FIR) filters and infinite impulse response (IIR) filters. The goal of filtering is to perform frequency-dependent alteration of a signal. A simple design specification for a filter might be to remove noise above a certain cutoff frequency.

Within the MATLAB environment, filter design and application occur in either two or three stages, each stage executed by separate, but related routines. In the three-stage protocol, the user supplies information regarding the filter type and desired attenuation characteristics, but not the filter order. The first-stage routines determine the appropriate order as well as other parameters required by the second-stage routines. The second stage routines generate the filter coefficients, b(n), based the arguments produced by the first-stage routines including the filter order. A two-stage design process would start with this stage, in which case the user would supply the necessary input arguments including the filter order.

When the designer known the filter order, i.e., the number of coefficients in b(n), The basic rectangular filter is implemented with the routine fir1 as:

```
b = fir1(n,wn,'ftype' window);
```

where n is the filter order, wn the cutoff frequency, ftype the filter type, and window specifies the window function (i.e., Blackman, Hamming, triangular, etc.). The output, b, is a vector containing the filter coefficients. The last two input arguments are optional

Filter Design And Application Using The MATLAB Signal Processing Toolbox

The MATLAB Signal Processing Toolbox includes routines that can be used to apply both FIR and IIR filters. Recent versions of MATLAB's Signal Processing Toolbox provide a filter design package called FDATool (for filter design and analysis tool) which performs the same operations de- scribed below, but utilizing a user-friendly graphical user interface (GUI). An- other Signal Processing Toolbox package, the SPTool (signal processing tool) is useful for analyzing filters and generating spectra of both signals and filters.

All the filter design functions in the Signal Processing Toolbox operate with normalized frequencies, so that they do not require the system sampling rate as an extra input argument. The normalized frequency is always in the interval $0 \leq f \leq 1$. For example, with a 1000 Hz sampling frequency, 300 Hz is 300/500 = 0.6. To frequency convert normalized frequency to angular around the unit circle, multiply by π. To convert normalized frequency back to Hertz, multiply by half the sample frequency.

Filter Specifications in Matlab

- Wp - Passband cutoff frequencies (normalized)
- Ws - Stopband cutoff frequencies (normalized)
- Rp - Passband ripple: deviation from maximum gain (dB) in the passband
- Rs - Stopband attenuation: deviation from 0 gain (dB) in the stopband

The Filter Design and Analysis Tool (FDATool) shows these specifications graphically: fdatool

Lab activity 01: Apply a band pass filter to the EEG data. Use a lower cutoff frequency of 6 Hz and an upper cutoff frequency of 12 Hz. Plot the data before and after band pass filtering. Also plot the spectrum of filtered data.

```
clear all;
close all;
clc;

fs = 100;
order = 128; % higher order --> greater roll of rate
wn = [6*2/fs 12*2/fs];
[b, a] = fir1(order, wn, "bandpass",hamming(order+1));
[h, freq] = freqz(b,a, 512, fs);

figure(1);
plot(freq, abs(h), "k");
xlabel("Frequency (Hz");
ylabel("H(f)")
title("Frequency Response of Bandpass filter 6Hz to 12Hz")

% apply on eeg data
load EEG_data
N = length(eeg);
t = (1:N)/fs;
filtered_eeg = filter(b,a, eeg); % apply designed filter to eeg
figure(2);
subplot(211);
plot(t, eeg);
title("Original EEG signal");
xlabel("Time");
```

```
ylabel("Amplitude");
subplot(212);
plot(t, filtered_eeg);
title("Filtered EEG signal");
xlabel("Time");
ylabel("Amplitude");

% freq response of filter
figure(3);
fscale = (1:N)*fs/N;
plot(fscale, abs(fft(filtered_eeg).^2));
xlim([0 20]);
xlabel("Frequency (Hz)");
ylabel("Amplitude");
title("Frequency response of filtered EEG signal");
```

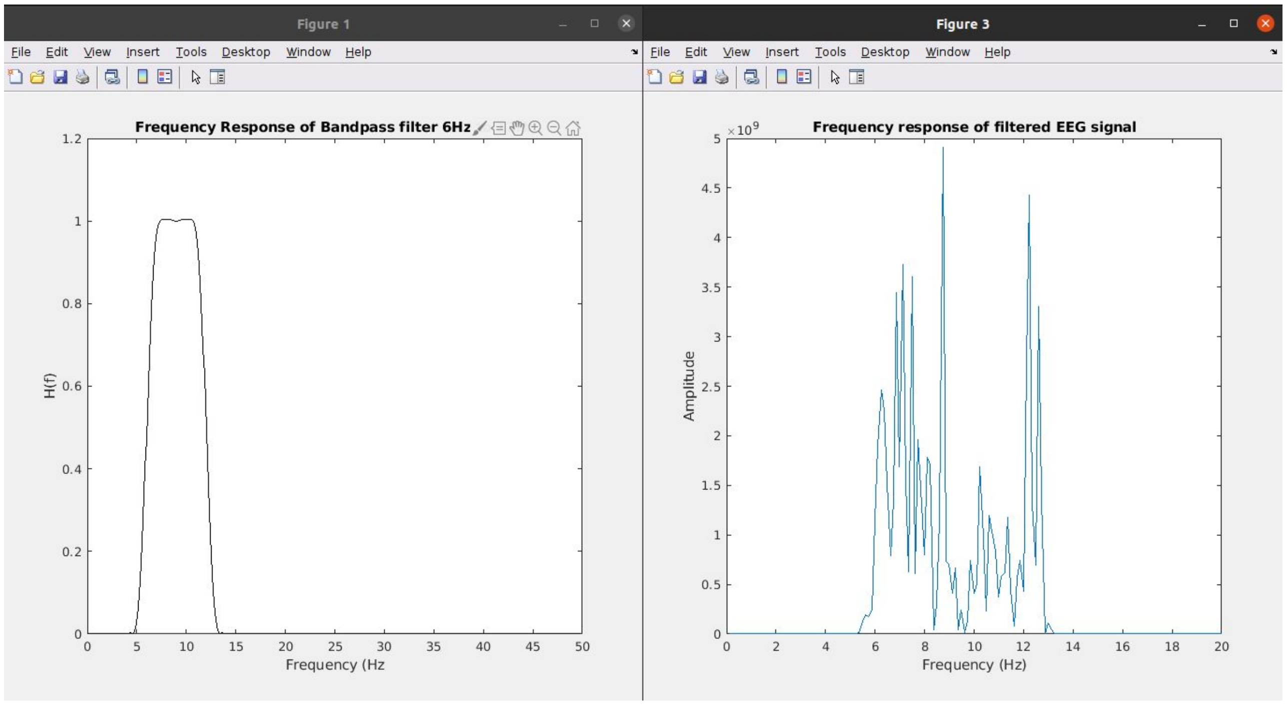
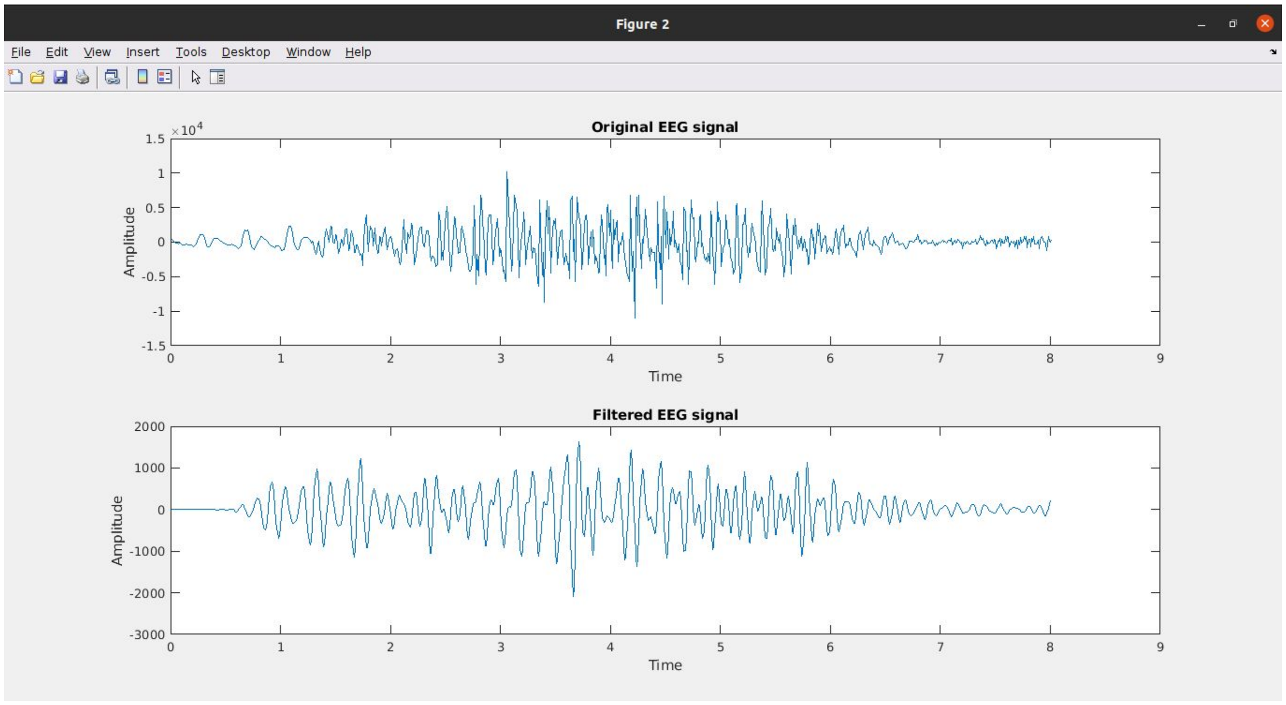


Figure 1
File Edit View Insert Tools Desktop Window Help
Frequency Response of Bandpass filter 6Hz
H(f)
Frequency (Hz
Figure 3
File Edit View Insert Tools Desktop Window Help
Frequency response of filtered EEG signal
$\times 10^9$
Amplitude
Frequency (Hz)



Lab Activity 02: Design a window-based bandpass filter with cutoff frequencies of 5 and 15 Hz. Assume a sampling frequency of 100 Hz. Filter order = 128

```
clear all;
close all;
clc;

% filter design
fs = 100;
order = 128;
wn1 = 5*2/fs; % normalized bandwidth (omega 1)
wn2 = 15*2/fs;
wn = [wn1 wn2]; % cut-off freqs for band pass/stop filters
b = fir1(order, wn, "bandpass"); % default window = hamming (input: N = order
+1)

[h, freq] = freqz(b, 1, 512, fs); % check mag and freq response of filter
plot(freq, abs(h));
title("Frequency response of bandpass filter");
xlabel("Frequency (Hz");
ylabel("H(f)")
```

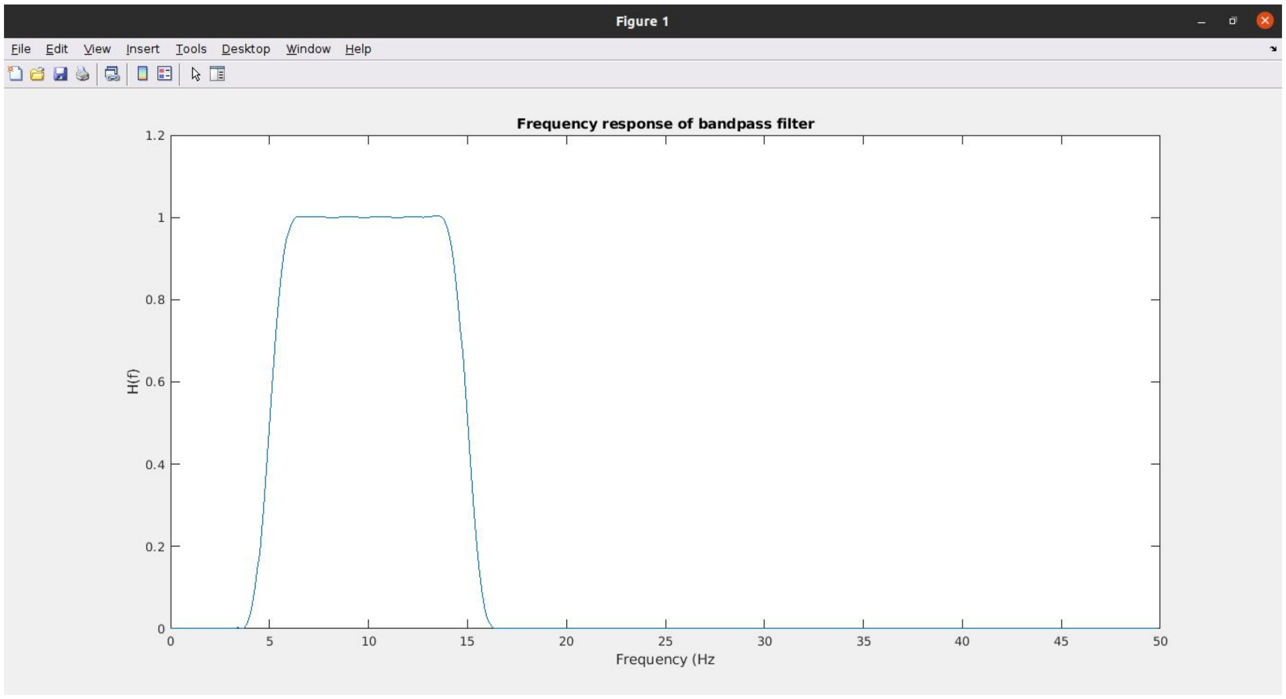


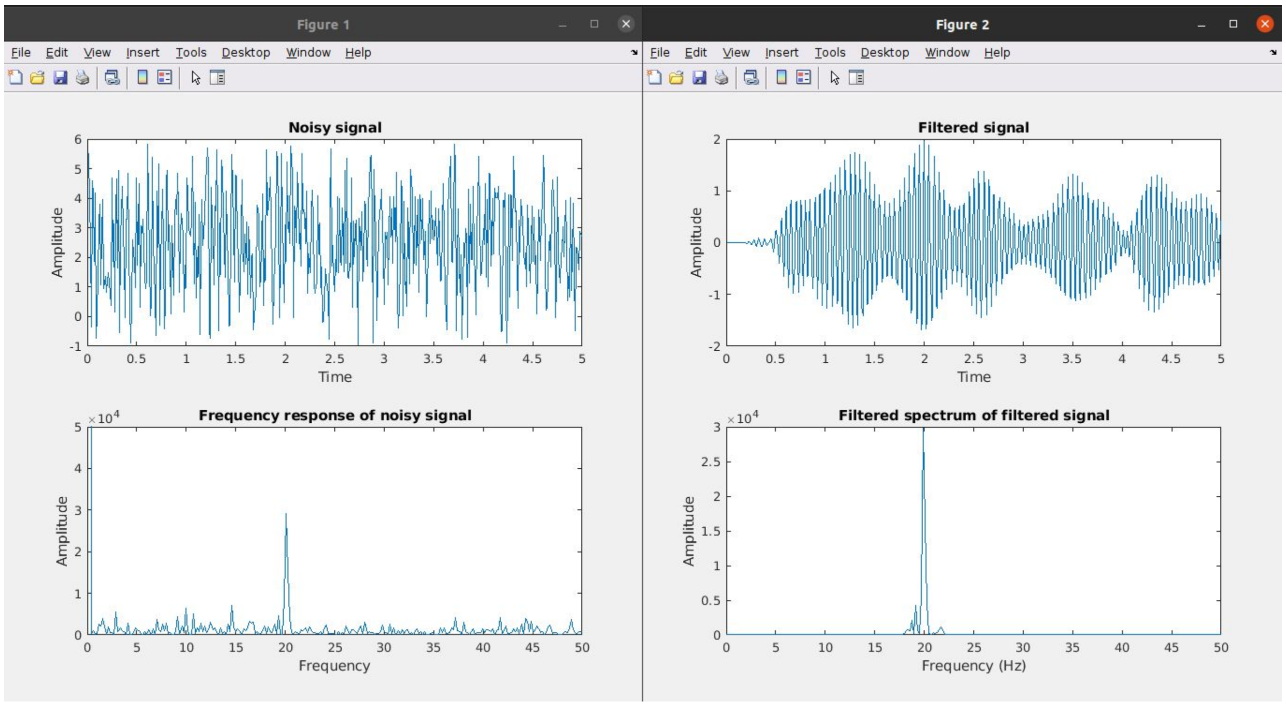
Figure 1
File Edit View Insert Tools Desktop Window Help
Frequency response of bandpass filter
H(f)
Frequency (Hz
0
0.2
0.4
0.6
0.8
1
1.2
0
5
10
15
20
25
30
35
40
45
50

Post Lab Task

1. Generate a sine wave with some random noise. Design a suitable filter to remove the noise.(use awgn function to add noise to the sine wave). Plot signal before and after filtration.

```
clear all;
close all;
clc;

f = 20;
fs = 100;
N = 513;
t = [1:N-1]/fs;
sig = sin(2*pi*t*f); % 20 hz sin wave
sig = sig + rand(size(sig)).*5; % add noise

figure(1);
subplot(211);
plot(t, sig);
title("Noisy signal (20 Hz Sine wave + Random noise)");
xlabel("Time");
ylabel("Amplitude");
xlim([0 5]);

freq = (1:N-1)*fs/N;
subplot(212);
plot(freq, abs(fft(sig).^2));
xlim([0 fs/2]);
title("Frequency response of noisy signal");
xlabel("Frequency");
ylabel("Amplitude");
ylim([0 50000])

order = 128; % higher order --> greater roll of rate
f_low = 18;
f_high = 22;
wn = [f_low*2/fs f_high*2/fs];
[b, a] = fir1(order, wn, "bandpass",hamming(order+1));
[h, freq] = freqz(b,a, N-1, fs*2);

filtered_sig = filter(b,a, sig); % apply designed filter to signal
figure(2);
subplot(211);
plot(t, filtered_sig)
title("Filtered signal");
```

```
xlabel("Time");
ylabel("Amplitude");
xlim([0 5]);

subplot(212);
plot(freq, abs(fft(filtered_sig).^2))
title("Filtered spectrum of filtered signal");
xlabel("Frequency (Hz)");
ylabel("Amplitude");
xlim([0 fs/2]);
```



2. Use SPTool and FDA tool of MATLAB to design a window-based bandpass filter with cutoff frequencies of 5 and 15 Hz. Assume a sampling frequency of 100 Hz. Filter order
= 128

Lab Session 11

Objective: Design and implementation of IIR filter in MATLAB

Introduction:

IIR filters are digital filters with infinite impulse response. IIR filters have much better frequency response than FIR filters of the same order. Unlike FIR filters, their phase characteristic is not linear which can cause a problem to the systems which need phase linearity. For this reason, it is not preferable to use IIR filters in digital signal processing when the phase is of the essence. well- known analog filter types can be duplicated as IIR filters. Specifically, analog filters termed Butterworth, Chebyshev Type I and II, and Elliptic (or Cauer) designs can be implemented as IIR digital filters and are supported in the MATLAB Signal Processing Toolbox.

Implementation in

MATLAB Butterworth IIR

filter

The filter coefficients for a Butterworth IIR filter can be determined using the MATLAB

routine: [b,a]=butter(order,wn,'ftype')

Where order and wn are the order and cutoff frequencies, respectively. if wn is scalar or a band pass filter if wn is a two-element vector. In the latter case, wn=[w1 w2], where w1 is the low cutoff frequency and w2 is the high cutoff frequency. If a high-pass filter is desired, then wn should be a scalar and 'ftype' should be high. For a stop band filter, wn would be a two-element vector indicating the frequency ranges of the stop band and 'ftype' should be stop. The outputs of butter are the b and coefficients.

Chebyshev Type I and II filters

The Chebyshev Type I and II filters are designed with similar routines except an additional parameter is needed to specify the allowable ripple, rp, which specifies the maximum desired pass band ripple in dB.

[b,a]=cheby1 (order, rp,wn,'ftype'); % Design Chebyshev Type I

The Type II Chebyshev filter is designed using: rs specifies the stopband ripple again in dB.

[b,a]=cheby2(n,rs, wn,'ftype'); % Design Chebyshev Type II

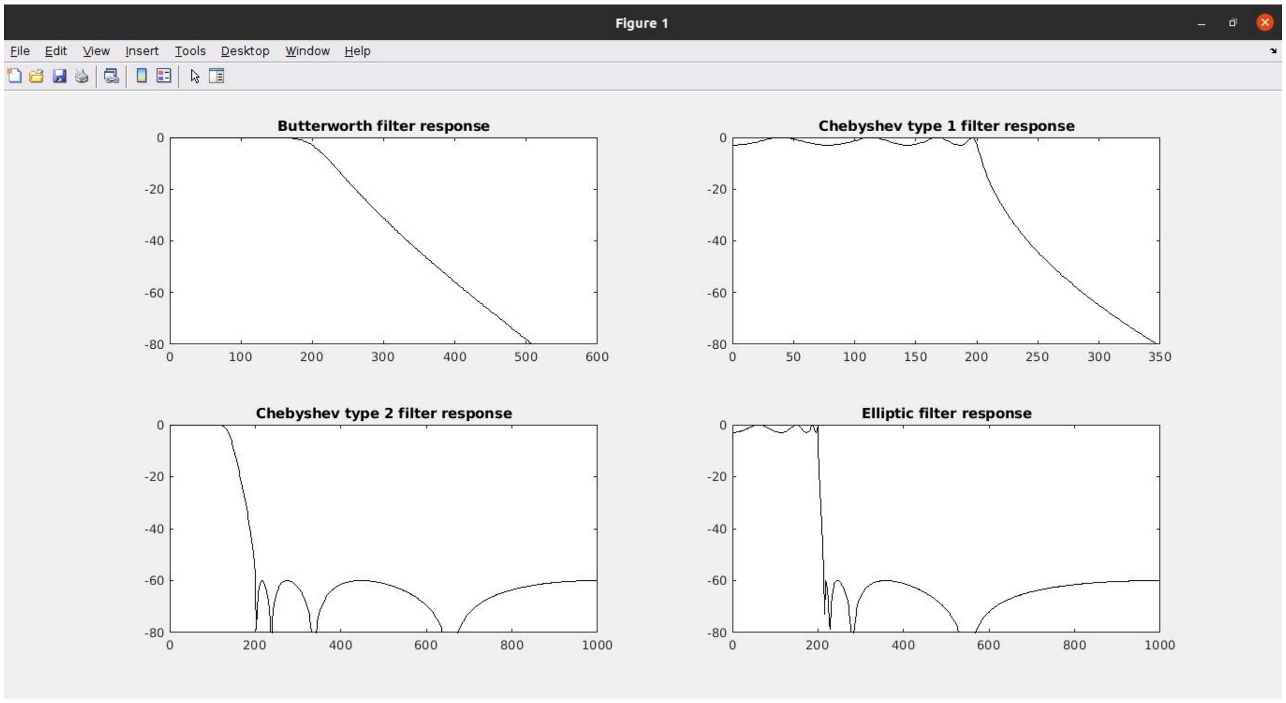
Elliptic filter

The Elliptic filter includes both stopband and passband ripple values:

[b,a]=ellip(n,rp,rs,wn,'ftype'); % Design Elliptic filter

Lab Activity 01: Plot the frequency response curves (in dB) obtained from an 8th-order low-pass filter using the Butterworth, Chebyshev Type I and II, and Elliptic filters. Use a cutoff frequency of 200 Hz and assume a sampling frequency of 2 kHz. For all filters, the ripple or maximum attenuation should be less than 3 dB in the pass band, and the stop band attenuation should be at least 60 dB. Use routine functions and fdatool of signal processing toolbox. Compare the frequency response of filters also attached the plots.

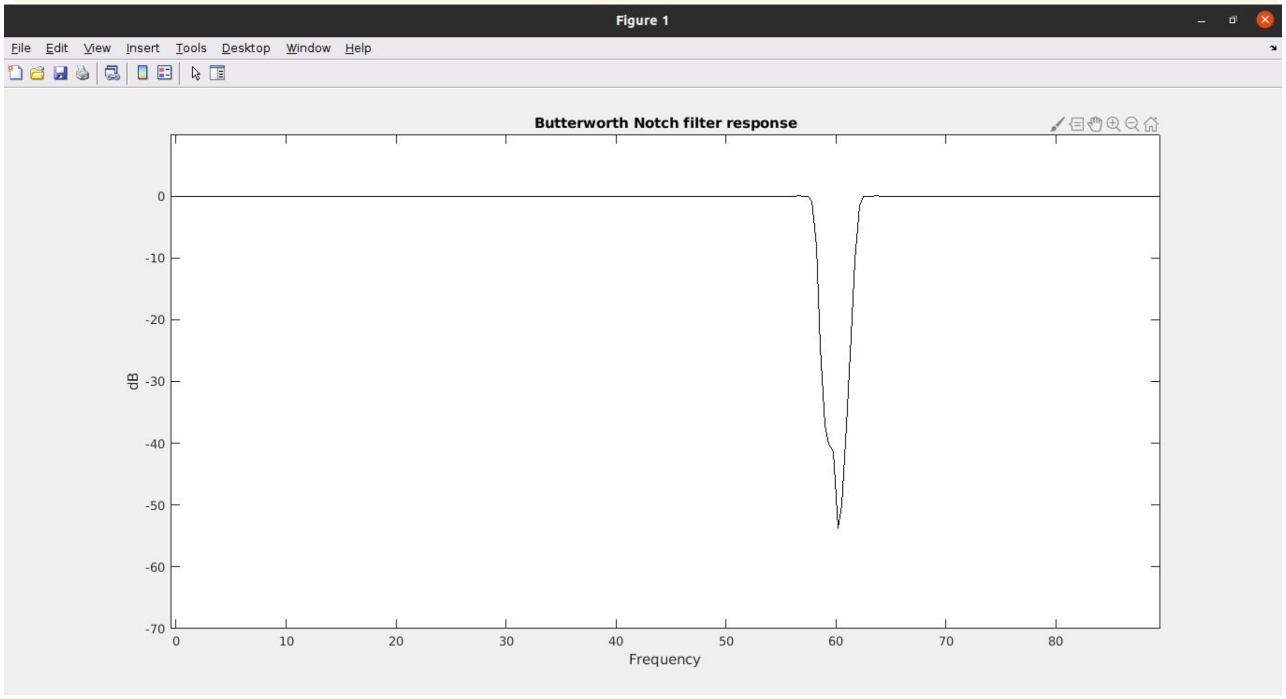
```
clear all;
close all;
clc;

fs = 2000;
n = 8; % order of filter
wn = 200*2/fs; % normalized cutoff freq (200)
rp = 3; % passband ripples
rs = 60; % stopband ripples

% butterworth filter has no ripple in pass/stop band
[b, a] = butter(n,wn, "low"); % 2nd stage IIR filter: Since we know n, wn
% b and a are coefs of filter

[H, f1] = freqz(b, a, 512, fs); % freq response of filter,
% no. of points in output spectrum  = 512

H = 20*log10(abs(H)); % plot freq response ROR
subplot(221);
plot(f1,H, "k");
ylim([-80 0]);
title("Butterworth filter response");

% chebyshev type 1 filter has ripple in passband
[b,a] = cheby1(n,rp,wn);
[H, f2] = freqz(b,a,512, fs);
H = 20*log10(abs(H));
subplot(222);
plot(f2, H,"k");
title("Chebyshev type 1 filter response");
ylim([-80 0]);

% chebyshev type 2 filter has ripples in stopband
% also called equiripple filter
[b,a] = cheby2(n,rs,wn);
[H, f3] = freqz(b,a,512, fs);
H = 20*log10(abs(H));
subplot(223);
plot(f3, H,"k");
title("Chebyshev type 2 filter response");
ylim([-80 0]);

% Elliptic filter has ripples in both pass and stopband
% fastest ROR (transition from pass to stop band)
[b,a] = ellip(n,rp, rs,wn);
[H, f4] = freqz(b,a,512, fs);
H = 20*log10(abs(H));
subplot(224);
plot(f4, H,"k");
title("Elliptic filter response");
ylim([-80 0]);
```

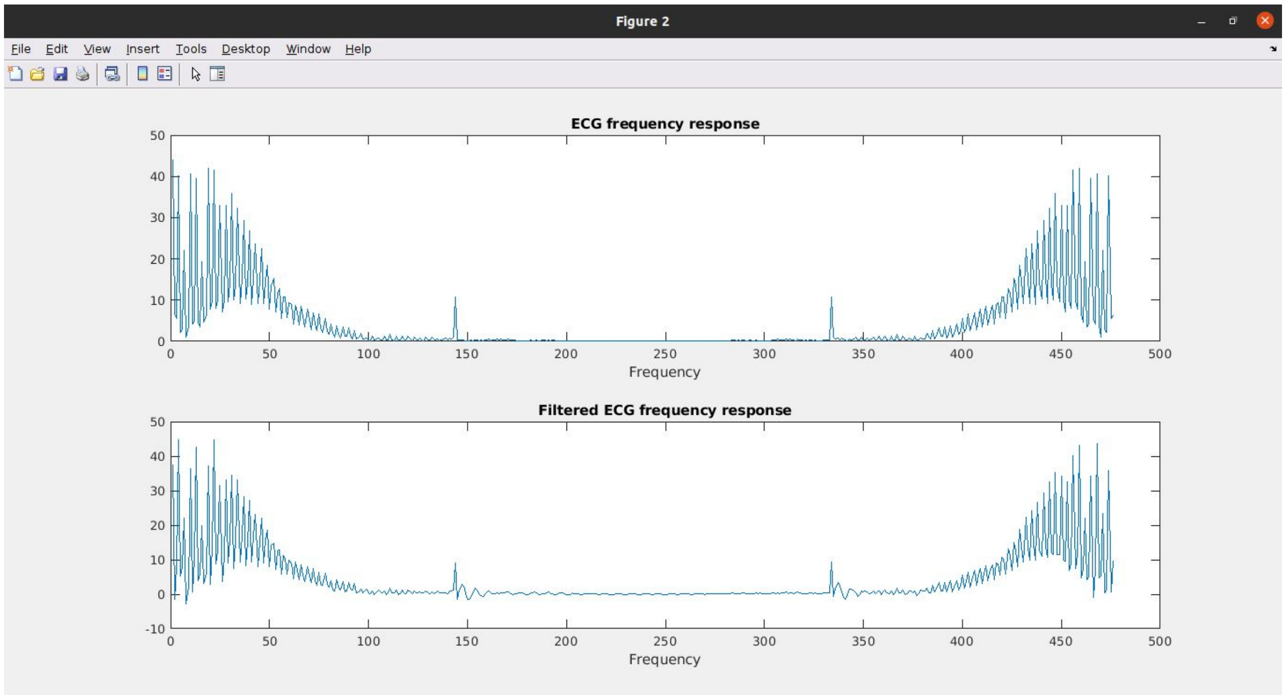


Lab Activity 02: Download unfiltered ECG signal from ECG-ID database of physioBank ATM**.** Design a Butterworth notch filter to remove line noise. Plot both ECG signal before and after filtration. Also compare your filtered ECG signal from downloaded filtered signal.

```
load ecg.dat;


fs = 400;
n = 8;
```

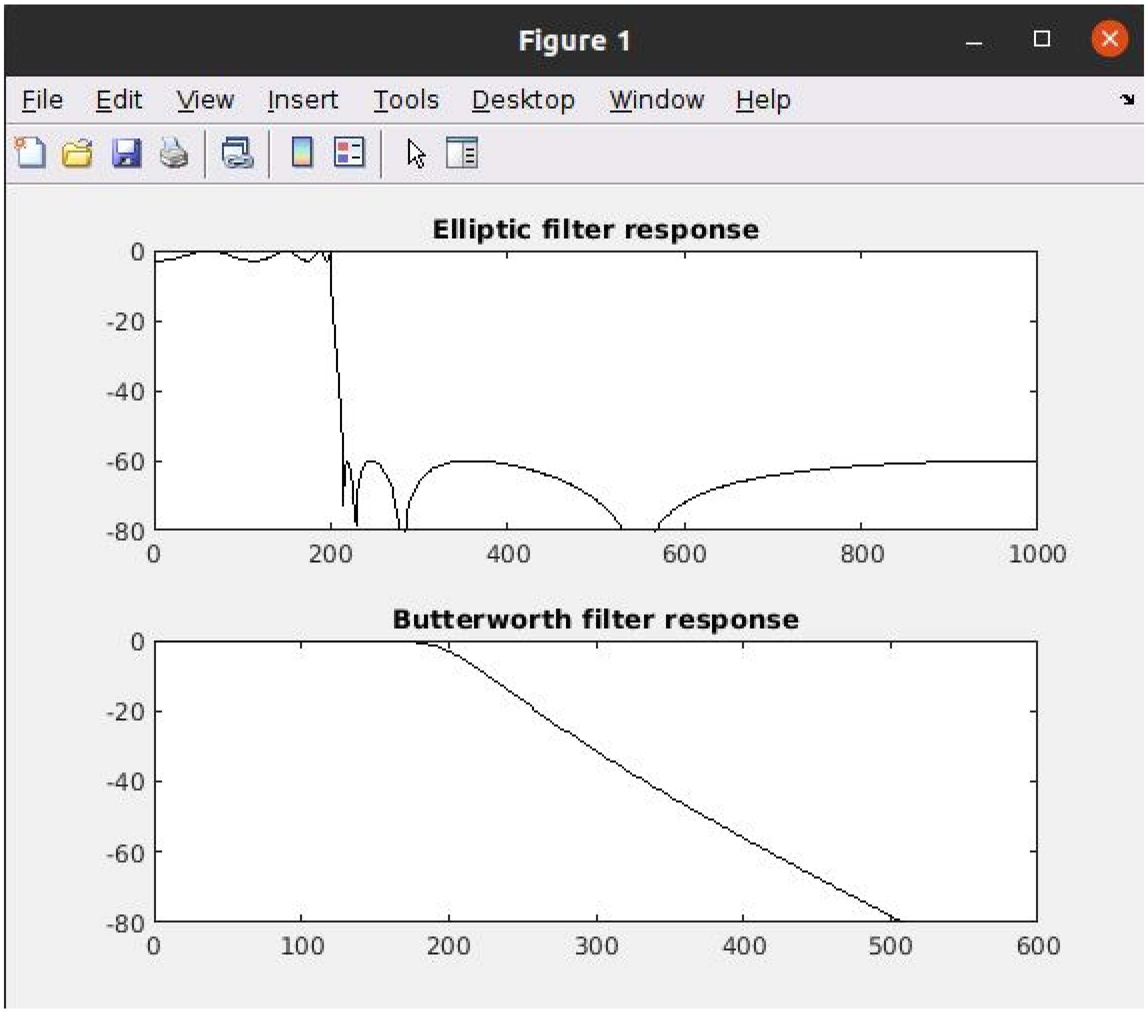
```
wp = 58*2/fs;
wn = 62*2/fs;

[b,a] = butter(n, [wp, wn],'stop'); % notch filter for 60Hz noise
[H, f1] = freqz(b,a,512, fs);
H = 20*log10(abs(H));
figure(1);
plot(f1,H, "k");
ylim([-80 0]);
xlabel("Frequency")
title("Butterworth Notch filter response");

ecg_freq = abs(fft(ecg));
ecg_filtered = filter(b,a,(ecg_freq));

figure(2);
subplot(211);
plot(1:length(ecg_filtered), ecg_freq);
xlabel("Frequency")
title("ECG frequency response");
subplot(212);
plot(1:length(ecg_filtered), ecg_filtered);
xlabel("Frequency")
title("Filtered ECG frequency response");
```





Post Lab Task

1. Compare frequency response of Butterworth and elliptic IIR filter.
   - The Elliptic IIR filters have sharper frequency shift between pass and stop bands, i.e greater roll of rate than bandpass filter
   - However, the elliptic IIR filter has ripples in both pass and stop bands. Therefore, butterworth filter has smoother response, i.e free of ripples, and hence recommended for biomedical signal filtering.



2. Compare IIR and FIR filters in terms of frequency response and application.
   - FIR filters are more powerful than IIR filters, but also require more processing power and more work to set up the filters.
   - The IIR filter's response can become infinite very rapidly, however, it has better response given same order of filter as compared to FIR.

Lab Session 12

Objective: Spectral Analysis of Signal Using Welch Method

Introduction:

Welch method is used for evaluating the power spectrum divides the data in several segments, possibly overlapping, performs an FFT on each segment, computes the magnitude squared (i.e., power spectrum), then averages these spectra. Averaging is usually achieved by dividing the waveform into a number of segments and evaluating the power spectrum on each of these segments the final spectrum is constructed from the *ensemble average* of the power spectra obtained from each segment.

MATLAB Function:

The **pwelch*** function of signal processing toolbox is used to perform
spectral averaging. In general it is called as:

[PS,f] = pwelch(x,window,noverlap,nfft,fs)

Only the first input argument, the name of the data vector, is required as the other arguments have default values. By default, **x** is divided into eight sections with 50% overlap, each section is windowed with a Hamming window and eight period grams are computed and averaged. If window is an integer, it specifies the segment length, and a Hamming window of that length is applied to each segment. The sampling frequency is specified by the optional argument fs and is used to fill the frequency vector, f, in the output with appropriate values.

Lab Activity 01: Generate a noisy sine wave in MATLAB. Use pwelch function to compute the spectral power of the signal. Plot both, noisy signal and PSD of the signal. Write your observation.

```
N = 1024;                                   % Number of data points
fs = 1000;                                  % The sample frequency of data is 1
kHz.
freq = (0:N-1)*fs/(N-1);                    % Frequency vector for plotting

t = [1:N]/fs;
x = sin(2*pi*t*5)+ cos(2*pi*t*6)+ 0.8*rand(1,N);
% [x,t] = sig_noise (250,-14,N);
Xf = fft(x);
Mf = abs(Xf).^2;
figure(1)
plot(freq,10*log10(Mf),'k');
hold on;
```

```
xlim([0 500])
title('Spectrum','FontSize',14);
xlabel('Frequency (Hz)','FontSize',14);
ylabel('Magnitude','FontSize',14);

[pyy1,f1]=pwelch(x ,[],[],4096,fs);
figure(2)
plot(f1,10*log10(pyy1));
```

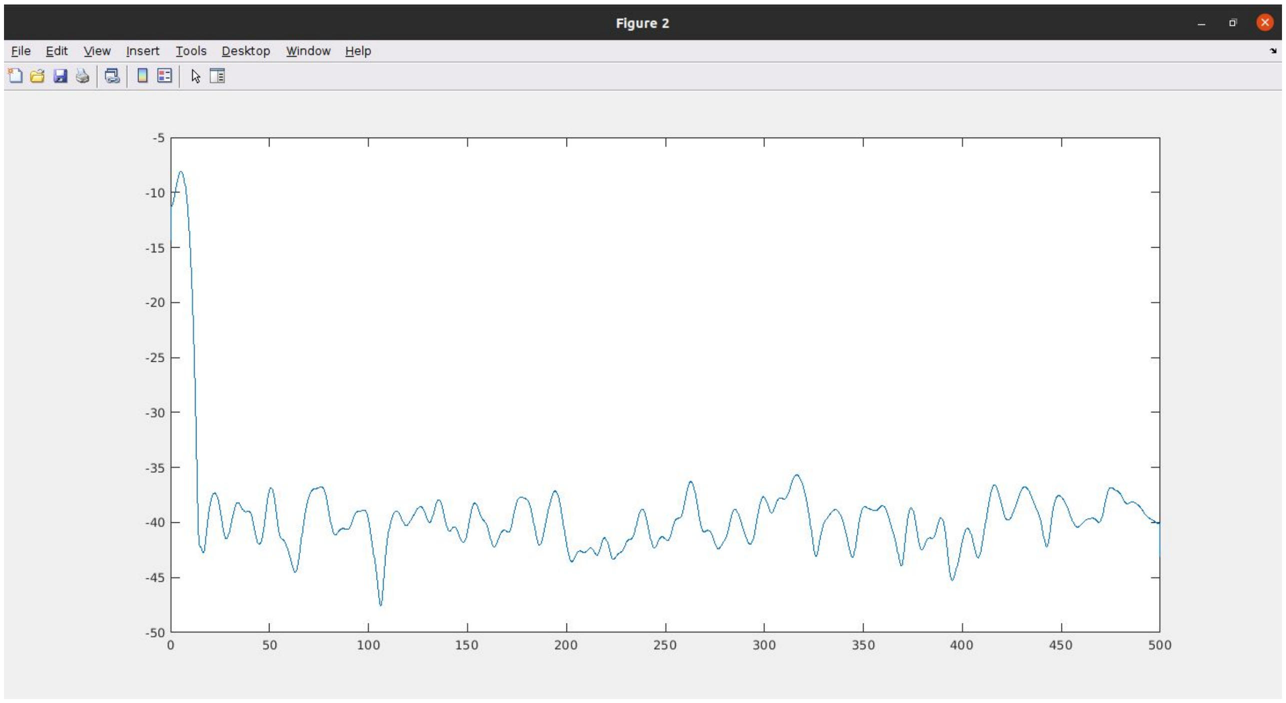


Figure 2
File Edit View Insert Tools Desktop Window Help

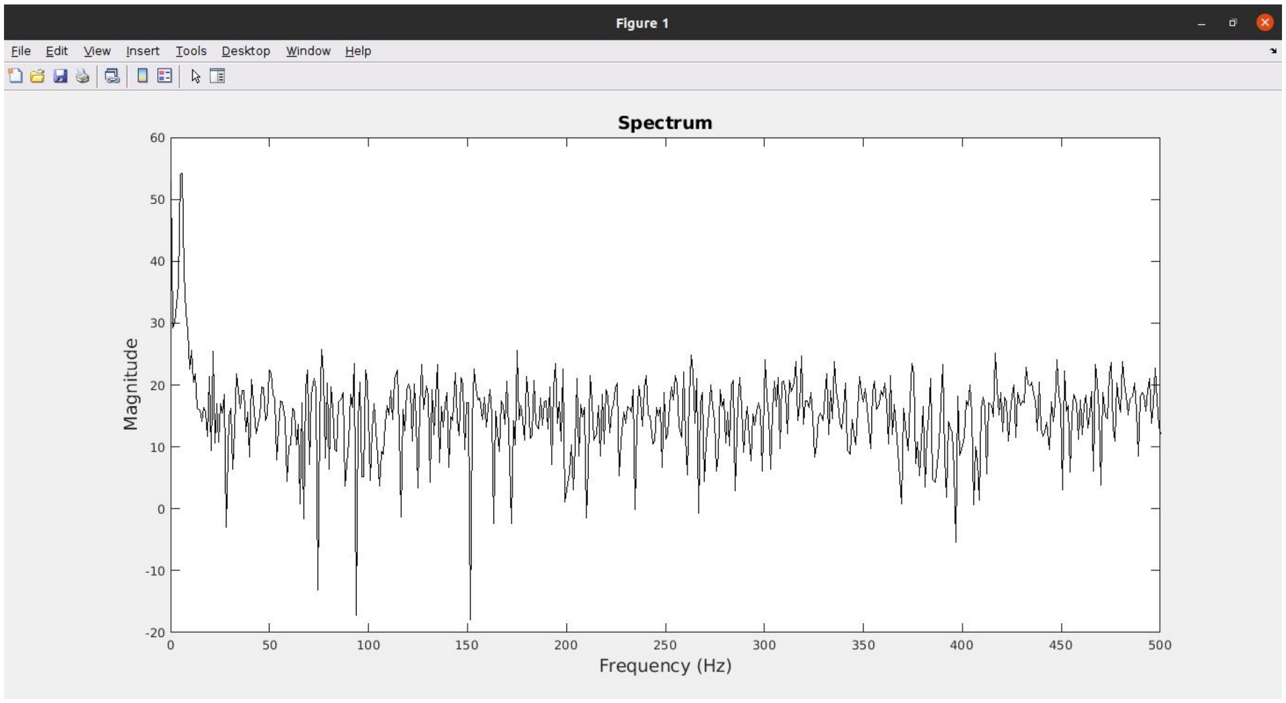


Figure 1
File Edit View Insert Tools Desktop Window Help
Spectrum
Magnitude
Frequency (Hz)