Quest for Excellence	Maharashtra Institute of Technology, Aurangabad	LABORATORY MANUAL

Practical Experiment Instruction Sheet

Manual:: MIT(T)/ ETC /_____/Signals & Systems			
Class: SY E&TC	DEPARTMENT: Electronics & Telecommunication Engineering.		
LABORATORY :	Location:-	PART: II	Date:_ _/_ _/20_ _

Electronics & Telecommunication Engineering Department

Vision:

To develop the department into a full fledged centre of learning in the various fields of Electronics and Telecommunication keeping in views the latest development and making student technologically superior and ethically strong.

Mission:

To impart education and training in the field of electronics and telecommunication engineering and its ailed areas by developing competencies of the students to meet social and industrial need.

Program Educational Objectives

PEO1	Graduates will demonstrate professional engineering competencies.
PEO2	Graduates will apply engineering and science knowledge to solve technical problems with high ethical standards.
PEO3	Graduates will have effective communication skills.
PEO4	Graduates will demonstrate leadership, teamwork in their profession and adapt to current trends by engaging lifelong learning.

Program Outcomes:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review and analyze engineering problems reaching substantiated conclusions.
3. **Design/development of solutions**: Design system components or processes as per need and specification.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data in the field of electronics and telecommunication.
5. **Modern tool usage**: Select, and apply appropriate techniques, modern engineering tools, skills and equipment necessary for engineering practices.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal and safety issues.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics, responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively in both verbal and written form.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs)

Engineering Graduates will be able to:

1. Design analog and digital system.
2. Simulate and test communication system.
3. Design and implement embedded system.

COURSE OUTCOMES

Course Name ETC274 Lab VIII: Signals And Systems **Year of Study: 2019 - 2020**

ETC274.1	Classify Continuous time and Discrete time signals and systems.
ETC274.2	Apply time domain analysis techniques to LTI systems. (A)
ETC274.3	Represent different signals in Frequency domain by Fourier series and Fourier Transform. (A)
ETC274.4	Analyze Discrete Time systems by Z Transform. (A)
ETC274.5	Generate different Continuous time and Discrete time signals in MATLAB.
ETC274.6	Implement software simulations of common systems in MATLAB.

Quest for Excellence	Maharashtra Institute of Technology, Aurangabad	LABORATORY MANUAL

Practical Experiment Instruction Sheet			
Manual:: MIT(T)/ ETC /_____/Signals & Systems			
Class: SY E&TC	DEPARTMENT: Electronics & Telecommunication Engineering.		
LABORATORY :	Location:-	PART: II	Date:_ _/_ _/20_ _

MASTER LIST OF EXPERIMENT

Course Name: ETC274 Lab VIII: Signals And Systems Class: SY ETC

EXPT. NO.	EXPERIMENT NAME	PAGE NO.	DATE	GRADE / REMARK	SIGN
01	Study of different MATLAB commands used for Signals and Systems				
02	MATLAB Program to plot various Continuous time signals				
03	MATLAB Program to plot various Discrete time signals				
04	MATLAB Program to Perform addition, Sustraction and Multiplication of Signals.				
05	MATLAB Program to find Even and Odd parts of the Signals.				
06	MATLAB Program to Find convolution of two DT signals using ' conv ' command				
07	Matlab Program to calculate Autocorrelation and Cross-correlation between two DT signals by using 'xcorr ' command.				
08	MATLAB Program to plot poles and zeros of transfer function of system.				
09	MATLAB Program to plot Magnitude and Phase response of second order system.				
10	Generation of Simple GUI				

Course Teacher **Class Teacher** **Program coordinator** **Principal**

Practical Experiment Instruction Sheet

EXPERIMENT TITLE: Study of different MATLAB commands used for Signals and Systems

EXPERIMENT NO. : **MIT(T)/ ETC** / _____/S&S/01

Class: SY E&TC	DEPARTMENT: Electronics & Telecommunication Engineering.		
LABORATORY :	Location:-	PART: II	Date:_ _/_ _/20_ _

Experiment No. 1

Aim: Study of different MATLAB commands used for Signals and Systems

Theory:

What is MATLAB

MATLAB started as an interactive program for doing matrix calculations and has now grown to a high level mathematical language that can solve integrals and differential equations numerically and plot a wide variety of two and three dimensional graphs. In this subject you will mostly use it interactively and also create MATLAB scripts that carry out a sequence of commands. MATLAB also contains a programming language that is rather like Pascal.

The first version of Matlab was produced in the mid 1970s as a teaching tool. The vastly expanded Matlab is now used for mathematical calculations and simulation in companies and government labs ranging from aerospace, car design, signal analysis through to instrument control & financial analysis. Other similar programs are Mathematica and Maple. Mathematica is somewhat better at symbolic calculations but is slower at large numerical calculations.

Recent versions of Matlab also include much of the Maple symbolic calculation program.

Matlab has two different methods for executing commands: *interactive mode* and *batch mode*. In interactive mode, commands are typed (or cut-and-pasted) into the 'command window'. In batch mode, a series of commands are saved in a text file (either using Matlab's built-in editor, or another text editor such as Emacs) with a '.m' extension. The batch commands in a file are then executed by typing the name of the file at the Matlab command prompt. The advantage to using a '.m' file is that you can make small changes to your code (even in different Matlab sessions) without having to remember and retype the entire set of commands. Also, when using Matlab's built-in editor, there are simple debugging tools that can come in handy when your programs start getting large and complicated. More on writing .m files later.

In this lab you will cover the following basic things:

1. Using Matlab as a numerical calculator.
2. Entering row vectors and column vectors.
3. Entering matrices.
4. Forming matrix and vector products.
5. Doing matrix products sums etc.
6. Using Matlab to solve linear equations.
7. Matlab functions that operate on arrays.
8. Plotting basic graphs using Matlab.

Scalar Variables and Arithmetic Operators

Scalar variables are assigned in the obvious way:

```
>>x=7;
x=7
```

The >> is Matlab's command prompt. Notice that Matlab echos back to you the value of the assignment you have made. This can be helpful occasionally, but will generally be unwieldy when working with vectors. By placing a semicolon at the end of a statement, you can suppress this verbose behavior.

```
>> x = 7;
```

Variables in Matlab follow the usual naming conventions. Any combination of letters, numbers and underscore symbols ('_') can be used, as long as the first character is a letter. Note also that variable names are case sensitive ('X' is different from 'x').

All of the expected scalar arithmetic operators are available:

```
>> 2*x
ans =
   14

>> x^2
ans =
   49
```

Notice that the multiply operation is not implied as it is in some other computational environments and the '*' operator has to be specified (typing '>>2x' will result in an error). This is also a good time to point out that Matlab remembers the commands that you are entering and you can use the up-arrow button to scroll through them and edit them for re-entry.

This is very handy, especially when you are repeatedly making minor changes to a long command line. There are a few predefined variables in Matlab that you will use often: pi, i, and j. Both i and j are defined to be the square-root of -1 and pi is defined as the usual 3.1416... . These variables can be assigned other values, so the statement would only remove the variable named '`x`' from Matlab's memory.

```
>> pi = 4;
```

Care is needed to avoid changing a predefined variable and then forgetting about that change later. That may seem unreasonable with `pi`, but `i` and `j` are both natural variables to use as indices and they are often changed. A useful command is the '`clear`', or '`clear all`' command. Typing either of these at the command prompt will remove all current variables from Matlab's memory and reset predefined variables to their original values. The `clear` command can also remove one specific variable from memory. The command

```
>> clear x
```

would only remove the variable named '`x`' from Matlab's memory

Matrix Variables and Arithmetic Operators

Another useful Matlab command is 'whos', which will report the names and dimensions of all variables currently in Matlab's memory. After typing the command at the prompt we see:

```
>> whos

Name      Size        Bytes   Class
x         1x1             8   double array

Grand total is 1 elements using 8 bytes
```

The important thing to note right now is that the size is given as '1x1'. Matlab is really designed to work with vectors and matrices, and a scalar variable is just a special case of a 1x1 dimensional vector. To

assign a vector containing the first 5 integers to the variable x, we could type this command:

```
>> x = [1 2 3 2^2 2*3-1]
x =
   1    2    3    4    5
```

```
4    5    6
```

We won't have much occasion to operate on matrices that are higher dimension, but if you wanted to create a 2-D matrix you could use a command something like:

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

To create larger vectors than the toy examples above (say, the integers up to 100), we would need to type a lot of numbers. Not surprisingly, there are easier ways built in to create vectors. To create the same 5-element vector we did above, we could also type:

```
>> x = [1:5]
x =
     1     2     3     4     5
```

The colon operator creates vectors of equally spaced elements given a beginning point, and maximum ending point and the step size in between elements. Specifically, `[b:s:e]` creates the vector `[b b+s b+2*s ... e]`. If no step size is specified (as in the example above), a step of 1 is assumed. So, the command `[1:2:10]` would create the vector of odd integers less than 10, `[1 3 5 7 9]` and the command `[1:3:10]` would create the vector of elements `[1 4 7 10]`. Let's create the vector of odd elements mentioned above:

```
>> x_odd = [1:2:10]
x_odd =
     1     3     5     7     9
```

You can access any element by indexing the vector name with parenthesis (indexing starts from one, not from zero as in many other programming languages). For instance, to access the 3rd element, we would type:

```
>> x_odd(3)
ans =
     5
```

We can also access a range of elements (a subset of the original vector) by using the colon operator and giving a starting and ending index.

```
>> x_odd(2:4)
ans =
     3     5     7
```

If we want to do simple arithmetic on a vector and a scalar, the expected things happen,

```
>> 3+[1 2 3]
ans =
     4     5     6

>> 3*[1 2 3]
ans =
     3     6     9
```

and the addition or subtraction of matrices is possible as long as they are the same size:

```
>> [1 2 3]+[4 5 6]
ans =
     5     7     9
```

The operators '*' and '/' actually represent matrix multiplication and division which is not typically what we will need in this course. However, a common task will be to form a new vector by multiplying (or

dividing) the elements of two vectors together, and the special operators '.*' and './' serve that purpose.

```
>> [1 2 3].*[4 5 6]
ans =
     4    10    18

>> [1 2 3]./[4 5 6]
ans =
    0.2500    0.4000    0.5000
```

Beware that the operator '^' is a shortcut for repeated *matrix* multiplications ('*'), whereas the operator '.^' is the shortcut for repeated *element-by-element* multiplications ('.*'). So, to square all of the elements in a vector, we would use

```
>> [1 2 3].^2
ans =
     1     4     9
```

Built-in Commands

Matlab has many built-in commands to do both elementary mathematical operations and also complex scientific calculations. The 'help' command will be useful in learning about built-in commands. By typing help at the command prompt, you are given a list of the different categories that Matlab commands fall into (e.g., general, elementary matrix operations, elementary math functions, graphics, etc.). Notice that there are probably even specific toolboxes included with your Matlab package for performing computations for disciplines such as signal processing. To see all of the commands under a certain topic, type 'help topic'. To get a description of a specific command, type 'help command'.

We'll look at a couple of example commands. First, the square-root command, sqrt. To calculate the square root of a number, type:

```
>> sqrt(2)
ans =
    1.4142
```

Again, to illustrate that Matlab understands complex numbers, we can have it calculate the root of a negative number:

```
>> sqrt(-9)
ans =
        0 + 3.0000i
```

Many Matlab commands that operate on scalars will also operate element-by-element if you give it a vector. For instance: represented by a vector. Because Matlab allows functions like `sin` and `sqrt` (as well as many others) to operate on vectors, many of the signal definitions and calculation we would like to perform are very straight-forward. We will illustrate this a little more explicitly in the next section.

```
>> sqrt([1 2 4 6])
ans =
    1.0000    1.4142    2.0000    2.4495

>> sqrt([1:8])
ans =
    1.0000    1.4142    1.7321    2.0000    2.2361    2.4495    2.6458    2.8284
```

Another useful command is `sin` function, which also operates on a vector.

```
>> sin([pi/4 pi/2 pi])
ans =
  0.7071    1.0000    0.0000
```

It is important to realize that digital signals are really just a collection of points and can be

It is important to realize that digital signals are really just a collection of points and can be represented by a vector. Because Matlab allows functions like sin and sqrt (as well as many others) to operate on vectors, many of the signal definitions and calculation we would like to perform are very straight-forward. We will illustrate this a little more explicitly in the next section.

Signals, Plotting and Batch Operation

There are a few commands necessary to do basic plotting of these signals. The `figure` command will bring up a new figure window and the `close` command closes a specific window (`close all` closes all open windows). The `subplot(r, c, p)` command allows many plots to be 'tiled' into `r` rows and `c` columns on one figure window. After this command is issued, any succeeding commands will affect the p^{th} tile. The `plot(x,y)` command will then plot the equal-length vectors `x` and `y` on the appropriate axis and in the obvious way. The two sinewaves created above are then plotted along with the sum of the two signals in separate tiles with the commands:

MATLAB Program:

```
%==========================START=============================================
```


```
%================================End of program============================
```

Conclusion:

Questions:

1.

2.

3.

Rubrics for Practical Assessment:

Cognitive (3)	Affective (3)	Psychomotor (3)	Total (9)

Sign of Course Teacher with Date

Practical Experiment Instruction Sheet

EXPERIMENT TITLE: MATLAB Program to plot various Continuous time signals

EXPERIMENT NO. : **MIT(T)/ ETC** / _____/S&S/02

Class: SY E&TC	DEPARTMENT: Electronics & Telecommunication Engineering.		
LABORATORY :	Location:-	PART: II	Date:_ _/_ _/20_

Experiment No. 2

Aim: Write a MATLAB program to plot various continuous time signals.

Theory: If the signal is defined over continuous-time, then the signal is a continuous-time signal

Different types of basic Continuous time signals are as follows:

1] Unit Impulse signal

It is denoted as δ(t) and defined as

x (t) = 1 for t = 0

= 0 otherwise

Also known as unit impulse signal. The amplitude of signal is zero everywhere, Except at t =0 where its value is unity.

2] Unit Step signal

It is denoted by u(t) and defined as

x (t) = 1 for t >=0

= 0 otherwise

3] Unit Ramp signal

It is denoted as r(n) and defined as

x (t) = t for t >= 0

=0 otherwise

4] The Exponential signal

It is a sequence of form

x (t) = a ^ t

Further if 0 < a <1 exponentially decaying function

a > 1 exponentially increasing function

5] The Sinusoidal signal:

x [t] =A * sin (2* pi* F *t)

where

A is Peak Amplitude

F is Frequency of Signal

6] The Signum Function

It is denoted as Sgn (t) and defined as

$$x(t) = 1 \quad \text{for } t > 0$$
$$= 0 \quad \text{for } t = 0$$
$$= -1 \quad \text{for } t < 0$$

MATLAB Program: (to plot continuous time signal)

```
%=========================START=============================================
```

OUTPUT:

Conclusion:

Questions:

1.

2.

3.

Rubrics for Practical Assessment:

Cognitive (3)	Affective (3)	Psychomotor (3)	Total (9)

Sign of Course Teacher with Date

Practical Experiment Instruction Sheet

EXPERIMENT TITLE: MATLAB Program to plot various Discrete time signals

EXPERIMENT NO. : **MIT(T)/ ETC** / _____/S&S/03

Class: SY E&TC	DEPARTMENT: Electronics & Telecommunication Engineering.		
LABORATORY :	Location:-	PART: II	Date:_ _/_ _/20_ _

Experiment No. 3

Aim:MATLAB Program to plot various Discrete time signals

.

Theory: Different types of basic discrete time signals are as follows:

1] Unit Impulse signal

It is denoted as d(n) and defined as

d(n) = 1 for n=0

= 0 otherwise

Also known as unit impulse signal. The amplitude of signal is zero everywhere, Except at n=0 where its value is unity.

2] Unit Step signal

It is denoted by u(n) and defined as

U(n) = 1 for n>_o

=0 for n<0

3] Unit Ramp signal

It is denoted as Ur(n) and defined as

Ur(n) = n for n>_o

=0 for n<0

4] The Exponential signal

It is a sequence of form

X(n) = a^n

Further if 0<a<1 exponentially decaying function

a>1 exponentially increasing function

5] The Sinusoidal signal:

x [t] =A * sin (2* pi* F *t)

where

A is Peak Amplitude

F is Frequency of Signal

Program for Discrete Time Signals:

```
% ==================== START ========================
%% Aim:- To write a MATLAB program for plotting basic discrete time sequence.
```

OUTPUT:

Conclusion:

Questions:

1.

2.

3.

Rubrics for Practical Assessment:

Cognitive (3)	Affective (3)	Psychomotor (3)	Total (9)

Sign of Course Teacher with Date

Practical Experiment Instruction Sheet

EXPERIMENT TITLE: MATLAB Program to Perform addition, Sustraction and Multiplication of Signals.

EXPERIMENT NO. : **MIT(T)/ ETC** / _____/S&S/04

Class: SY E&TC	DEPARTMENT: Electronics & Telecommunication Engineering.		
LABORATORY :	Location:-	PART: II	Date:_ _/_ _/20_ _

Experiment No.4

Aim: Write a MATLAB program to perform addition , subtraction and multiplication on signals

Theory:

Addition, subtraction, multiplication, differentiation, and integration fall under the category of basic signal operations acting on the dependent variable

1] Signal addition:

Let x{n] & y[n] be two sequences. The addition of two signal is defined as term by term z[n] as

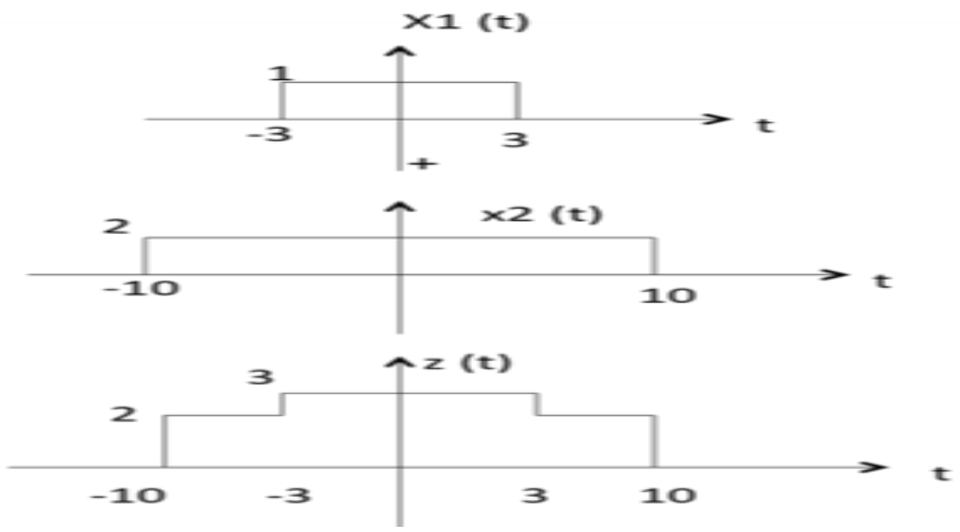
$$z[n] = x[n] + y[n]$$

e.g. if x [n] ={ 1 , ↑2 , 3 , 4} and y [n] = {↑1 ,1 ,1 ,1,}

then z [n] = [1, ↑3, 4, 5, 1]

Addition of two signals is nothing but addition of their corresponding amplitudes.

Amplitude addition:



As seen from the diagram above,

-10 < t < -3 amplitude of z(t) = x1(t) + x2(t) = 0 + 2 = 2

-3 < t < 3 amplitude of z(t) = x1(t) + x2(t) = 1 + 2 = 3

3 < t < 10 amplitude of z(t) = x1(t) + x2(t) = 0 + 2 = 2

2] Signal Substraction :

Let x{n] & y[n] be two sequences. The substraction of two signal is defined as term by term z[n] as

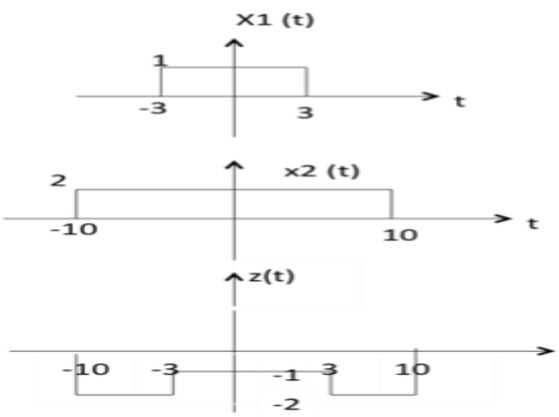
z [n] = x [n] - y[n]

e.g. if x [n] ={ 1 , ↑2 , 3 , 4} and y [n] = {↑1 ,1 ,1 ,1,}

then z [n] = [1, ↑1, 2, 3, -1]

subtraction of two signals is nothing but subtraction of their corresponding amplitudes.

This can be best explained by the following example:



As seen from the diagram above,

-10 < t < -3 amplitude of z (t) = x1(t) - x2(t) = 0 - 2 = -2

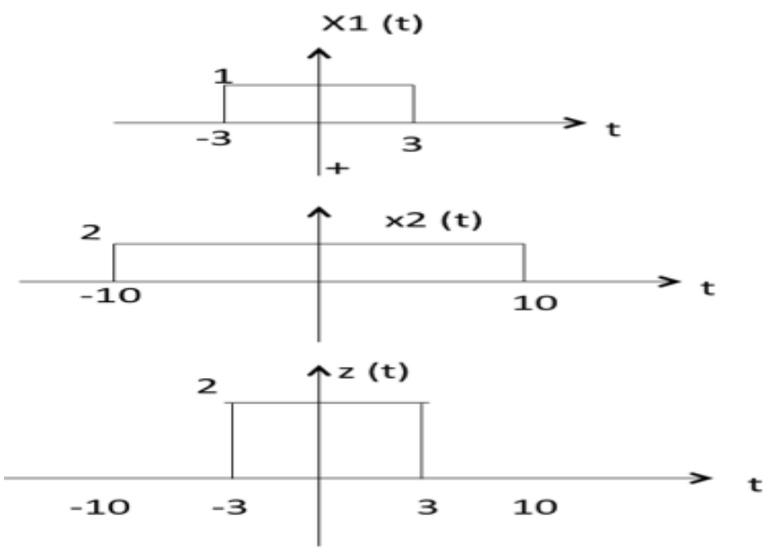
-3 < t < 3 amplitude of z (t) = x1(t) - x2(t) = 1 - 2 = -1

3 < t < 10 amplitude of z (t) = x1(t) + x2(t) = 0 - 2 = -2

3] Signal Multiplication:

Multiplication of two signals is nothing but multiplication of their corresponding amplitudes.

This can be best explained by the following example:



As seen from the diagram above,

-10 < t < -3 amplitude of z (t) = x1(t) ×x2(t) = 0 ×2 = 0

-3 < t < 3 amplitude of z (t) = x1(t) ×x2(t) = 1 ×2 = 2

3 < t < 10 amplitude of z (t) = x1(t) × x2(t) = 0 × 2 = 0

Program:

OUTPUT:

Conclusion:

Questions:

1.

2.

3.

Rubrics for Practical Assessment:

Cognitive (3)	Affective (3)	Psychomotor (3)	Total (9)

Sign of Course Teacher with Date

Practical Experiment Instruction Sheet

EXPERIMENT TITLE: Decomposition of Real signal into Even & Odd Components.

EXPERIMENT NO. : **MIT(T)/ ETC** / _____/S&S/05

Class: SY E&TC	DEPARTMENT: Electronics & Telecommunication Engineering.		
LABORATORY :	Location:-	PART: II	Date:_ _/_ _/20_ _

Experiment No.5

Aim: Write a MATLAB program for the Decomposition of real signal into even & odd components

Theory:

Even and odd signals bear some important symmetry properties. Under reversal of independent variable, these signals either remain the same (even signal) or get reflected or flipped (odd signal) about the horizontal axis. Equations or definitions (1) and (2) mathematically express these properties for both continuous and discrete time cases.

Even Signals: $x(t) = x(-t)$, $x[n] = x[-n]$ ---------- (1)

Odd Signals: $x(t) = -x(-t)$, $x[n] = -x[-n]$ ---------- (2)

Even and Odd Functions

A function, f , is even (or symmetric) when
f (x) = f (-x)
A function, f , is odd (or antisymmetric) when
f (x) = -f (-x)

Program:

OUTPUT:

Conclusion:

_____
_____
_____
_____

Questions:

1.

_____
_____
_____
_____
_____
_____

2.

_____
_____
_____
_____
_____
_____

3.

_____
_____
_____
_____
_____
_____

Rubrics for Practical Assessment:

Cognitive (3)	Affective (3)	Psychomotor (3)	Total (9)

Sign of Course Teacher with Date

Practical Experiment Instruction Sheet

EXPERIMENT TITLE: To generate program for Convolution.

EXPERIMENT NO. : **MIT(T)/ ETC** / _____/S&S/06

Class: SY E&TC	DEPARTMENT: Electronics & Telecommunication Engineering.		
LABORATORY :	Location:-	PART: II	Date:_ _/_ _/20_ _

Experiment No. 6

Aim: Write a MATLAB program for calculating 1-D linear convolution.

.

Theory: There are two types of Convolution:-

1) LinearConvolution
2) CircularConvolution

- The output y[n] of a LTI (linear time invariant) system can be obtained by convolving theinput x[n] with the system‘s impulse responseh[n].
- The convolutionsumis$y[n]=x[n]*h[n]=\sum_{k=-\infty}^{+\infty}x[k]h[n-k]=\sum_{k=-\infty}^{+\infty}x[n-k]h[k]$
- x[n] and h[n] can be both finite or infinite durationsequences.
- Ifboththesequencesareoffiniteduration,thenwecanusetheMATLABfunction_***conv***‘to evaluate the convolution sum to obtain the outputy[n].
- The length of y[n] = xlength + hlength-1.

Algorithm:

1. Input the two sequences as xn, hn
2. Convolve both to get output yn.
3. Plot the sequences.

1) Program for Linear convolution

Program(Method1)

```
% ============ START ====================
```

OUTPUT:

Convolution:

Program(Method2):

% program for linear convolution

OUTPUT:

Enter the first sequence

Enter the second sequence

The resulted signal is

Conclusion:

_____
_____
_____
_____

Questions:

1.

_____
_____
_____
_____
_____
_____

2.

_____
_____
_____
_____
_____
_____

3.

_____
_____
_____
_____
_____
_____

Rubrics for Practical Assessment:

Cognitive (3)	Affective (3)	Psychomotor (3)	Total (9)

Sign of Course Teacher with Date

Quest for Excellence	Maharashtra Institute of Technology, Aurangabad	LABORATORY MANUAL

Practical Experiment Instruction Sheet

EXPERIMENT TITLE: To generate program for cross correlation & Auto correlation.

EXPERIMENT NO. : **MIT(T)/ ETC** / _____/S&S/07

Class: SY E&TC	DEPARTMENT: Electronics & Telecommunication Engineering.		
LABORATORY :	Location:-	PART: II	Date:_ _/_ _/20_ _

Experiment No. 7

Aim:- Write a MATLAB program to calculate cross correlation & Auto correlation.

Theory:

Auto correlation

- Correlation is mathematical technique which indicates whether 2 signals are related and in aprecise quantitative way how much they are related. A measure of similarity between a pair of energy signals x[n] and y[n] is given by the cross correlation sequence $r_{xy}[l]$ defined by

  $r_{xy}[l] = \sum_{n=-\infty}^{\infty} x[n]y[n-l]; l = 0,\pm1,\pm2,....$

- The parameter _l' called _lag' indicates the time shift between the pair.
- Autocorrelation sequence of x[n] is given by $r_{xx}[l] = \sum_{n=-\infty}^{\infty} x[n]x[n-l]; l = 0,\pm1,\pm2,...$
- Some of the properties of autocorrelation are enumerated below
  - The autocorrelation sequence is an even function i.e., $r_{xx}[l] = r_{xx}[-l]$
  - Atzerolag,i.e.,atl=0,thesamplevalueoftheautocorrelationsequencehasitsmaximumvalue (equal to the total energy of the signal ε_x) i.e., $r_{xx}[l] \le r_{xx}[0] = \varepsilon = \sum_{n=-\infty}^{\infty} x^2[n]$.

    This is verified in Fig. 5.1, where the autocorrelation of the rectangular pulse (square) has a maximum value at l=0. All other samples are of lower value. Also the maximum value = 11 = energy of the pulse $[1^2+1^2+1^2..]$.
  - A time shift of a signal does not change its autocorrelation sequence. For example, let y[n]=x[n-k]; then $r_{yy}[l] = r_{xx}[l]$ i.e., the autocorrelation of x[n] and y[n] are the same regardless of the value of the time shift k. This can be verified with a sine and cosine sequences of same amplitude and frequency will have identical autocorrelationfunctions.
  - For power signals the autocorrelation sequence is givenby

    $r_{xx}[l] = \lim_{k\to\infty} \frac{1}{2k+1} \sum_{n=-k}^{k} x[n]x[n-l]; l = 0,\pm1,\pm2,.....$ and for periodic signals with period N itis

    $r_{xx}[l] = \frac{1}{N} \sum_{n=0}^{N-1} x[n]x[n-l]; l = 0,\pm1,\pm2,.....$ and this **r_{xx}[l] is also periodic** with N. This is verifiedin Fig. 5.3 where we use the periodicity property of the autocorrelation sequence to determine the period of the periodic signal y[n] which is x[n] (=cos(0.25*pi*n)) corrupted by an additive uniformly distributed random noise of amplitude in the range [-0.5 0.5].

Algorithm:

1. Input the sequence asx.
2. fliplr function x and use the conv to get Auto correlated outputrxx.
3. Plot the outputsequence.

Cross correlation:

Cross Correlation has been introduced in the last experiment. Comparing the equations forthe linear convolution and cross correlation we find that

$r_{xy}[l] =$

∞

$\sum_{n=-\infty}^{\infty} x[n]y[n-l]=$

$\sum x[n]y[-(l-n)] = x[l] * y[-l]$. i.e. convolving the reference signal with a folded version of sequence to be shifted (y[n]) results in cross correlation output. (Use _fliplr' function for folding the sequence for correlation).

The cross correlation of two sequences x[n] and y[n]=x[n-k] shows a peak at the value of k. Hence cross correlation is employed to compute the exact value of the delay k between the 2 signals. Used in radar and sonar applications, where the received signal reflected from the target is the delayed version of the transmitted signal (measure delay to determine the distance of the target).

The ordering of the subscripts xy specifies that x[n] is the reference sequence that remains fixed in time, whereas the sequence y[n] is shifted w. r .t x[n]. If y[n] is the reference sequence then $r_{yx}[l]=r_{xy}[-l]$.Hence $r_{yx}[l]$is obtained by time reversing the sequence $r_{xy}[l]$.

Algorithm:

1. Input the sequence as x and y.
2. fliplr function y and use the conv to get cross correlated output rxy.
3. Plot the output sequence

MATLAB Program for Auto correlation:

```
%====================START=====================
```

```
%====================End=====================
```

Result:

Enter sequence x(n)=

rxx=

To obtain cross correlation of the given sequence
MATLAB Program

Result:

Enter sequence x(n)=

Enter the second sequence y(n)=

rxy=

Conclusion:

Questions:

1.

2.

3.

Rubrics for Practical Assessment:

Cognitive (3)	Affective (3)	Psychomotor (3)	Total (9)

Sign of Course Teacher with Date

Quest for Excellence	Maharashtra Institute of Technology, Aurangabad	LABORATORY MANUAL

Practical Experiment Instruction Sheet

EXPERIMENT TITLE: Program to find the poles and zeros of transfer function.

EXPERIMENT NO. : **MIT(T)/ ETC** / _____/S&S/08

Class: SY E&TC	DEPARTMENT: Electronics & Telecommunication Engineering.		
LABORATORY :	Location:-	PART: II	Date:_ _/_ _/20_ _

Experiment No. 8

Aim: Write a MATLAB program to find the poles and zeros of transfer function

Theory:

Poles and **Zeros** of a transfer function are the frequencies for which the value of the denominator and numerator of transfer function becomes zero respectively. The values of the poles and the zeros of a system determine whether the system is stable, and how well the system performs.

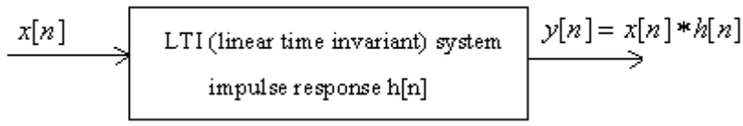


Fig.6.1 A LTI system

- A discrete time LTI system (also called digital filters) as shown in Fig.6.1 is representedby
  - A linear constant coefficient difference equation, for example,
    $y[n] + a_1 y[n-1] - a_2 y[n-2] = b_0 x[n] + b_1 x[n-1] + b_2 x[n-2]$;
  - A system function H (z) (obtained by applying Z transform to the differenceequation).
    $H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a z_1^{-1} + a z_2^{-2}}$
- GiventhedifferenceequationorH(z),theimpulseresponseoftheLTIsystemisfoundusing**filter or impz** MATLAB functions.

MATLAB Program:

Result:

[Given y(n)-y(n-1)+0.9y(n-2)= x(n)]

Difference Equation of a digital system

Desired Impulse response length =

Coefficients of x[n] terms =

Coefficients of y[n] terms =

Conclusion:

_____
_____
_____
_____

Questions:

1.

_____
_____
_____
_____
_____
_____

2.

_____
_____
_____
_____
_____
_____

3.

_____
_____
_____
_____
_____
_____

Rubrics for Practical Assessment:

Cognitive (3)	Affective (3)	Psychomotor (3)	Total (9)

Sign of Course Teacher with Date

Practical Experiment Instruction Sheet

EXPERIMENT TITLE: Program for Magnitude and Phase response of second order system.

EXPERIMENT NO. : **MIT(T)/ ETC** / _____/S&S/09

Class: SY E&TC	DEPARTMENT: Electronics & Telecommunication Engineering.		
LABORATORY :	Location:-	PART: II	Date:_ _/_ _/20_ _

Experiment No. 9

Aim: Write a MATLAB program to plot Magnitude and Phase response of second order system.

Theory:

The order of a system is determined by the power of Z in the denominator of its transfer function. If the power of Z in the denominator of transfer function of a system is 2, then the system is said to be second-order system. The general expression of transfer function of a second order system is given as

$$H(z) = \frac{Z+1}{Z^2+2Z+3}$$

The frequency response of a system is the major way of characterizing how a system behaves in the frequency domain. It is important to understand the frequency characteristics of a given system rather than time domain characteristics alone for many practical applications like filter design.

The frequency response of a system, $H(\omega)$ is just the transfer function, $H(z)$evaluated at z = e ^ jω . Frequency response is usually a complex valued function, so it can be written as $H(\omega) = |H(\omega)|\angle H(\omega)$, where $|H(\omega)|$ is the magnitude response and $\angle H(\omega)$ is the phase response. You can use MATLAB to plot the magnitude and phase responses as follows:

Example: $H(z) = \frac{Z+1}{Z^2+2Z+3}$to plot magnitude and phase responses use the following command lines. Define two vectors b and a where b is the coefficients of the polynomial in the numerator, and a is the coefficients of the polynomial in the numerator.

Note that they have to be the same length and start with the coefficient of the highest order term. Then define w as the vector of frequencies over which you want to plot the magnitude and phase responses.

MATLAB Program:

OUTPUT:

Conclusion:

_____
_____
_____
_____

Questions:

1. _____
_____
_____
_____
_____

2. _____
_____
_____
_____
_____

3. _____
_____
_____
_____
_____

Rubrics for Practical Assessment:

Cognitive (3)	Affective (3)	Psychomotor (3)	Total (9)

Sign of Course Teacher with Date

Quest for Excellence	Maharashtra Institute of Technology, Aurangabad	LABORATORY MANUAL

Practical Experiment Instruction Sheet

EXPERIMENT TITLE: Program for developing GUI in MATLAB.

EXPERIMENT NO. : **MIT(T)/ ETC** / _____/S&S/10

Class: SY E&TC	DEPARTMENT: Electronics & Telecommunication Engineering.		
LABORATORY :	Location:-	PART: II	Date:_ _/_ _/20_ _

Experiment No. 10

Aim:-To study application of GUI in MATLAB

Theory:-

A graphical user interface (GUI) is a graphical display that contains devices, or components, that enable a user to perform interactive tasks. To perform these tasks, the user of the GUI does not have to create a script or type commands at the command line. Often, the user does not have to know the details of the task at hand. The GUI components can be menus, toolbars, push buttons, radio buttons, list boxes, and sliders. In MATLAB, a GUI can also display data in tabular form or as plots, and can group related components.

GUI Working:-

Each component, and the GUI itself, is associated with one or more user-written routines known as callbacks. The execution of each callback is triggered by a particular user action such as a button push, mouse click, selection of a menu item, or the cursor passing over a component. This kind of programming is often referred to as event-driven programming. In event-driven programming, callback execution is asynchronous, controlled by events external to the software. In the case of MATLAB GUIs, these events usually take the form of user interactions with the GUI.

Ways to Build MATLAB GUIs

A MATLAB GUI is a figure window to which you add user-operated controls. You can select, size, and position these components as you like. Using callbacks you can make the components do what you want when the user clicks or manipulates them with keystrokes.

You can build MATLAB GUIs in two ways:

- Use GUIDE (GUI Development Environment), an interactive GUI construction kit.
- Create M-files that generate GUIs as functions or scripts (programmatic GUI construction).

The first approach starts with a figure that you populate with components from within a graphic layout editor. GUIDE creates an associated M-file containing callbacks for the GUI and its components. GUIDE saves both the figure (as a FIG-file) and the M-file. Opening either one also opens the other to run the GUI.

In the second, programmatic, GUI-building approach, you code an M-file that defines all component properties and behaviors; when a user executes the M-file, it creates a figure, populates it with components, and handles user interactions. The figure is not normally saved between sessions because the M-file creates a new one each time it runs.

As a result, the M-files of the two approaches look different. Programmatic M-files are generally longer, because they explicitly define every property of the figure and its controls, as well as the callbacks. GUIDE GUIs define most of the properties within the figure itself. They store the definitions in its FIG-file rather than in its M-file. The M-file contains callbacks and other functions that initialize the GUI when it opens.

MATLAB software also provides functions that simplify the creation of standard dialog boxes, for example to issue warnings or to open and save files. The GUI-building technique 3 you choose depends on your experience, your preferences, and the kind of application you need the GUI to operate.

You can combine the two approaches to some degree. You can create a GUI with GUIDE and then modify it programmatically. However, you cannot create a GUI programmatically and later modify it with GUIDE.

Before Designing a GUI

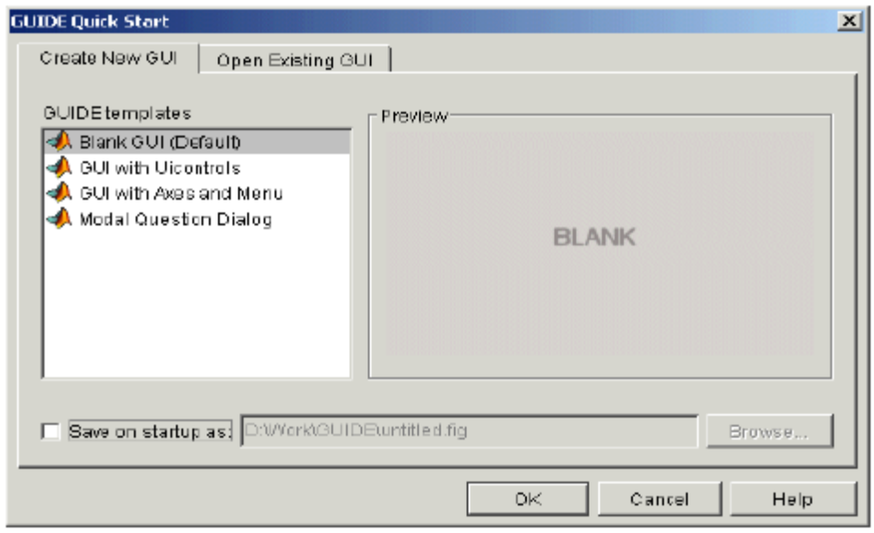
Before creating the actual GUI, it is important to decide what it is you want your GUI to do and how you want it to work. It is helpful to draw your GUI on paper and envision what the user sees and what actions the user takes.

Starting GUIDE

There are many ways to start GUIDE. You can start GUIDE from the:

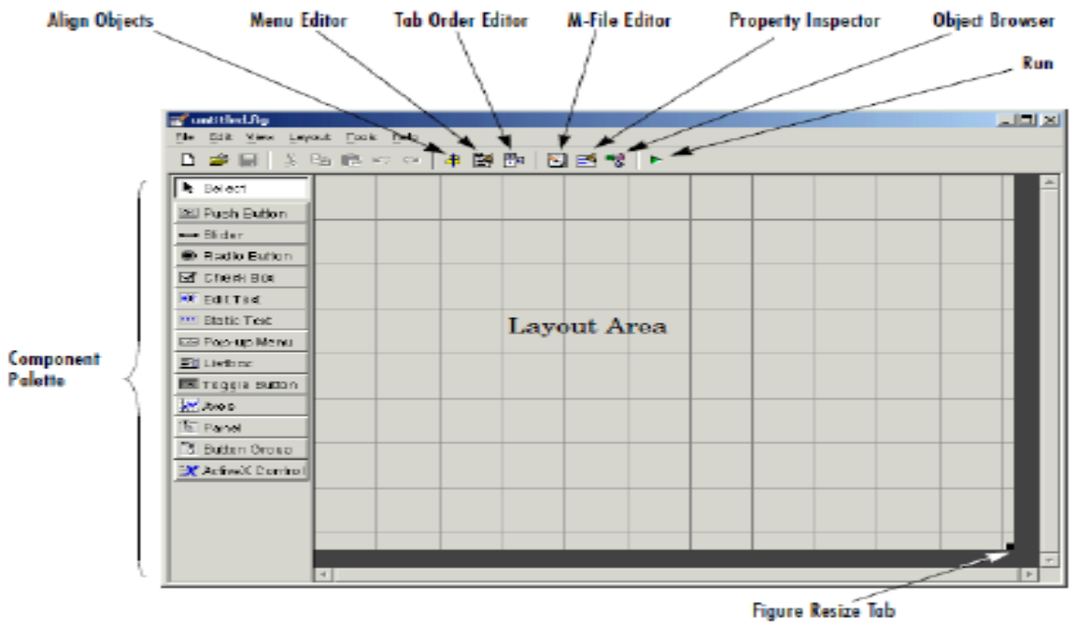
- Command line by typing guide
- Start menu by selecting MATLAB
- GUIDE (GUI Builder) MATLAB File menu by selecting New
- GUI MATLAB toolbar by clicking the GUIDE button

However you start GUIDE, it displays the GUIDE Quick Start dialog box shown in the following figure.



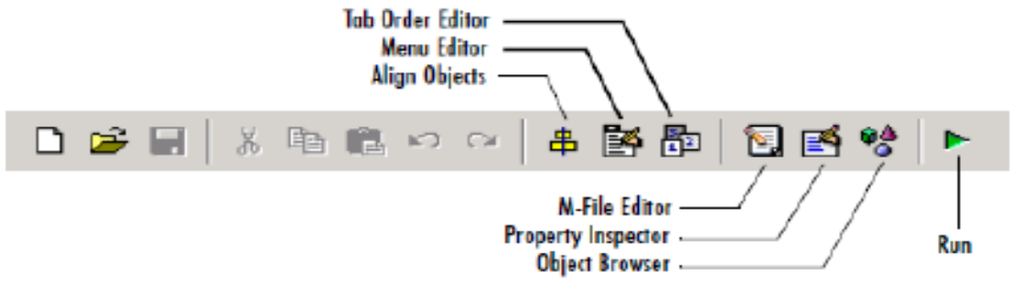
GUIDE Tools Summary

The GUIDE tools are available from the Layout Editor shown in the figure below. The tools are called out in the figure and described briefly below.

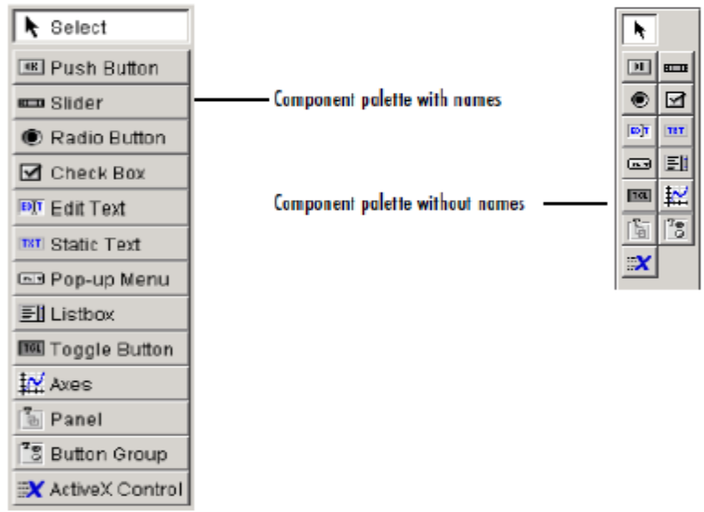


Show Toolbar

Displays the following toolbar in the Layout Editor window.



Show Names in Component Palette:
When you first open the Layout Editor, the component palette contains only icons. To display the names of the GUI components, select **Preferences** from the **File** menu, check the box next to **Show names in component palette**, and click **OK**.



Available Components
The component palette at the left side of the Layout Editor contains the components that you can add to your GUI. You can display it with or without names

Component	Description
Push Button	Push buttons generate an action when clicked. For example, an **OK** button might apply settings and close a dialog box. When you click a push button, it appears depressed; when you release the mouse button, the push button appears raised.
Toggle Button	Toggle buttons generate an action and indicate whether they are turned on or off. When you click a toggle button, it appears depressed, showing that it is on. When you release the mouse button, the toggle button remains depressed until you click it a second time. When you do so, the button returns to the raised state, showing that it is off. Use a button group to manage mutually exclusive toggle buttons.
Radio Button	Radio buttons are similar to check boxes, but radio buttons are typically mutually exclusive within a ngroup of related radio buttons. That is, when you select one button the previously seleted button is deselected. To activate a radio button, click the mouse button on the object. The display indicates the state of the button. Use a button group to manage mutually exclusive radio buttons.
Check Box	Check boxes can generate an action when checked and indicate their state as checked or not checked. Check boxes are useful when providing the user with a number of independent choices, for example, displaying a toolbar.
Edit Text	Edit text components are fields that enable users to enter or modify text strings. Use edit text when you want text as input. Users can enter numbers but you must convert them to their numeric equivalents.
Static Text	Static text controls display lines of text. Static text is typically used to label other controls, provide directions to the user, or indicate values associated with a slider. Users cannot change static text interactively.

Slider	Sliders accept numeric input within a specified range by enabling the user to move a sliding bar, which is called a slider or thumb. Users move the slider by clicking the slider and dragging it, by clicking in the trough, or by clicking an arrow. The location of the slider indicates the relative location within the specified range.
List Box	List boxes display a list of items and enable users to select one or more items.
Pop-Up Menu	Pop-up menus open to display a list of choices when users click the arrow.
Axes	Axes enable your GUI to display graphics such as graphs and images. Like all graphics objects, axes have properties that you can set to control many aspects of its behavior and appearance. See “Axes Properties” in the MATLAB Graphics documentation and commands such as the following for more information on axes objects: plot, surf, line, bar, polar, pie, contour, and mesh. See Functions — By Category in the MATLAB Documentation for a complete list.
Panel	Panels arrange GUI components into groups. By visually grouping related controls, panels can make the user interface easier to understand. A panel can have a title and various borders. Panel children can be user interface controls and axes as well as button groups and other panels. The position of each component within a panel is interpreted relative to the panel. If you move the panel, its children move with it and maintain their positions on the panel.
Button Group	Button groups are like panels but are used to manage exclusive selection behavior for radio buttons and toggle buttons.

GUI:

Conclusion:

_____
_____
_____
_____

Questions:

1.

_____
_____
_____
_____
_____
_____

2.

_____
_____
_____
_____
_____
_____

3.

_____
_____
_____
_____
_____
_____

Rubrics for Practical Assessment:

Cognitive (3)	Affective (3)	Psychomotor (3)	Total (9)

Sign of Course Teacher with Date