**Lab 0 Introduction to MATLAB**

MATLAB stands for "**MAT**rix **LAB**oratory" which is an interactive, matrix-based computer program for scientific and engineering numeric computation and visualization. The aim of MATLAB is to enable us to solve complex numerical problems, without having to write programs in traditional languages like C and FORTRAN. Thus, MATLAB interprets commands like Basic does, instead of compiling source code like C and FORTRAN require. By using the relatively simple programming capability of MATLAB, it is very easy to create new commands and functions in MATLAB. In addition, these developed MATLAB programs (or scripts) can run without modification on different computers with MATLAB. Today, MATLAB has evolved into a very powerful programming environment by providing numerous toolboxes such as signal processing, image processing, and control, optimization, and statistics computations. The emphasis of this beginner's guide is on the basic MATLAB commands with investigation on some aspects of signal processing.

**Matrices**

MATLAB has only one data type: a complex-valued floating-point matrix. A vector is simply a matrix with either one row [a row vector] or one column [a column vector]. A number, or scalar, is simply 1-by-1 matrix. Variables in MATLAB must start with an alphabetic character, and must contain only alphabetic characters, numeric characters, and the underscore character, e.g. **data** is okay, but **Data!** is not.

MATLAB is also case sensitive, e.g. **data** and **Data** are not the same identifier. For example, enter the following vectors and matrix in MATLAB as

**>> rowvector = [1 2 3]**

**rowvector =**

**1 2 3**

**>> columnvector = [1; 2; 3]**

columnvector =

1

2

3

**>> matrix2x2 = [1 2; 3 4]**

matrix2x2 =

1 2

3 4

In the matrix addressing and subscripting, MATLAB is similar enough to C to cause some extremely annoying errors. One such error is that array and vector indexing begins with 1 and not 0.

**>> x = [1 3 5 7 9]**

x =

1 3 5 7 9

**>> x(0)**

??? Subscript indices must either be real positive integers or logicals.

**>> x(1)**

**ans =**

1

For example, a 3-by-3 matrix **A** is defined as

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

**A =**

1 2 3

4 5 6

7 8 9

The element in the i'th row and j'th column of **A** is referred to in the usual way:

**>> A(2,3)**

**ans =**

6

In this example the expression was not explicitly assigned to a variable so MATLAB automatically assigned the result to the variable **ans**. It's very easy to modify matrices:

**>> A(3,2) = 10**

**A =**

1 2 3

3 5 6

7 10 9

**Martix and Array Operations**

MATLAB supports the following matrix arithmetic operations:

**+ addition - subtraction**

**\* multiplication ^ power**

**\ left division / right division**

**' transpose**

To experience the matrix arithmetic in the MATLAB, enter the following **A** and **b** matrices

**>> A = [1 2 3 ; 4 5 6 ; 7 8 10], b = [2; 4; 6]**

The transpose of a matrix is the result of interchanging rows and columns. MATLAB denotes the

[Conjugate] transpose by following the matrix with the single-quote [apostrophe].

**>> B = A'**

**B =**

 1 4 7

 2 5 8

 3 6 10

Scalars multiply matrices as expected, and matrices may be added in the usual way; both are done

"Element by element."

**>> 2\*A**

**ans =**

 2 4 6

 8 10 12

 14 16 20

Also try the following two commands:

**>> A/3**

 0.3333 0.6667 1.0000

 1.3333 1.6667 2.0000

 2.3333 2.6667 3.3333

**>> A + [b,b,b]**

 3 4 5

 8 9 10

 13 14 16

Scalars added to matrices produce a "strange" result, but one that is sometimes useful; the scalar is added to every element.

**>> A+1**

ans =

2 3 4

5 6 7

8 9 10

Matrix multiplication requires that the sizes match. If they don't, an error message is generated.

**>> A\*b**

ans =

 28

 64

 106

**>> b'\*A**

ans =

 60 72 90

**>> A\*A'**

 14 32 53

 32 77 128

 53 128 213

**>> A'\*A**

 66 78 97

 78 93 116

 97 116 145

**>> b'\*b, b\*b'**

ans =

 56

ans =

 4 8 12

 8 16 24

 12 24 36

To perform point-wise operation (array operation) on matrices, we can use the "point-star" operator, e.g. **A.\*B**. In general, where "point" is used with another arithmetic operator it modifies that operator's usual matrix definition to a point-wise one. Thus we have **./** and **.^** for point-wise division and exponentiation.

Try the following commands to gain more experience on point-wise operations.

**>> A^2, A.^2**

ans =

 30 36 45

 66 81 102

 109 134 169

ans =

 1 4 9

 16 25 36

 49 64 100

**>> A.\*A, b.\*b**

ans =

 1 4 9

 16 25 36

 49 64 100

ans =

 4

 16

 36

**>> 1./A**

ans =

 1.0000 0.5000 0.3333

 0.2500 0.2000 0.1667

 0.1429 0.1250 0.1000

**>> 1./A.^2**

ans =

 1.0000 0.2500 0.1111

 0.0625 0.0400 0.0278

 0.0204 0.0156 0.0100

**MATLAB Built-in Functions**

MATLAB has a number of matrix building functions such as:

**eye Identity matrix**

**zeros Matrix of zeros**

**ones Matrix of ones**

For example, **zeros(m,n)** produces an **m**-by-**n** matrix of **zeros** and **zeros(n)** produces an

**nxn** zero matrix. In addition, if **A** is a **m**-by-**n** matrix, then **zeros(A)** produces a matrix of zeros

of the same size as **A**. In our example **A** is 3-by-3 matrix, so

**>> zeros(3)**

ans =

0 0 0

0 0 0

0 0 0

If **b** is a vector, **diag(b)** is the diagonal matrix with **b** down the diagonal; if **A** is a square matrix, then **diag(A)** is a vector consisting of the diagonal of **A**. Matrices can be built from blocks. For example, if A is a 3-by-3 matrix, then

**B = [A, zeros(3,2); zeros(2,3), eye(2)]**

will build a certain 5-by-5 matrix.

**B** =

 1 2 3 0 0

 4 5 6 0 0

 7 8 10 0 0

 0 0 0 1 0

 0 0 0 0 1

**Scalar, Vector and Matrix Functions**

MATLAB has some functions operate essentially on scalars while operate element-by-element when applied to a matrix. The most common scalar functions are

**sin cos tan**

**asin acos atan**

**exp sign rem (remainder)**

**abs sqrt log (natural log)**

**round floor ceil**

Another useful MATLAB functions are vector-oriented which operate on vectors (row or column matrices). If they are applied to a matrix, they are computed on a column-by-column basis.

**Colon Operator and Sub-matrices**

The colon operator is very useful for creating index arrays, creating vectors of evenly spaced values, and accessing submatrices. The colon notation works from the idea that an index range can be generated by giving a **start**, **step**, and then the **end**. Therefore, a regularly spaced vector of integers is obtained via

**iii = start:step:end**

Without the **step** parameter, the increment is 1. This sort of counting is similar to the notation used in FORTRAN DO loops. For example, the expression 1:6 is actually a row vector [1 2 3 4 5 6]. Try

**» 1:6**

ans =

1 2 3 4 5 6

The numbers need not be integers. For example,

**» 0.1:0.2:0.9**

ans =

0.1000 0.3000 0.5000 0.7000 0.9000

Also try

**» 6:-1:1**

ans =

6 5 4 3 2 1

The following statements will, for example, generate a sinusoidal wave vector **y**. Try it.

**x = [0.0:0.1:2.0]';**

**y = sin(x);**

Note that since **sin** operates entry-wise, it produces a vector **y** from the vector **x**.

The colon notation can be used to access submatrices of a matrix. If you start with the matrix **A**, then **A(2,3)** is the scalar element located at the 2nd row, and 3rd column of **A**. But you can also pull out a 43 sub-matrix via **A(2:5,1:3)**. If you want an entire row, the colon serves as a wild card: i.e., **A(2,:)** is the 2nd row. You can even flip a vector by just indexing backwards: **x(L:-1:1)**. Finally, it is sometimes necessary to just work with all the values in a matrix, so **A(:)** gives a column vector that is just the columns of **A** concatenated together.

**Plotting and Graphics**

MATLAB is capable of producing 2-D and 3-D plots, displaying images, and even creating and playing movies. The two most common plotting functions that will be used in the Digital Signal Processing are **plot** and **stem**. The basic forms of **plot** and **stem** are the same with **plot(x,y)** producing a connected plot with data **points {{x(1),y(1)}, {x(2),y(2)},** **..., {x(N),y(N)}}** and **stem** producing a "lollipop" presentation of the same data.

Multiple plots per page can be done with the subplot function. To set up a 32 tiling of the figure window, use **subplot(3,2, title\_number)**. For example, **subplot(3,2,3)** will direct the next plot to the third tile which is in the second row, left side. For example, the above two figures can be put together in one page using the following commands:

**>>subplot(2,1,1); plot(tt,xx);**

**>>subplot(2,1,2); stem(tt,xx);**

In addition, the command **grid** will place grid lines on the current graph. The graphs can be given title, axes labeled, and text placed within the graph with the following commands which take a string as an argument.

**title** graph title **xlabel** x-axis label **y-label** y-axis label.

**Lab 1 Elementary Signals and Basic Signal Operations**

## OBJECTIVES

* To plot some elementary discrete-time signals using MATLAB,
* To perform operations on dependent variables of discrete-time signals.

## THEORY

### Unit Step Signal

The unit step signal in discrete-time is defined as:



### Unit Impulse Signal

The unit impulse signal in discrete-time is given as:



### Real Exponential Signal

A discrete-time real exponential signal is expressed as:



where A is the amplitude of the signal at and a is a real number. For , the signal decays while for , the signal grows exponentially.

### Sinusoidal Signal

The sinusoidal signal in discrete-time is given by the formula:



where A is the amplitude. If n is taken to be dimensionless, then both andhave units of radians.

### Amplitude Scaling

Let denotes a discrete-time signal. Then the signal resulting from amplitude scaling applied to is given by: 

where c is the scaling factor. In other words, the value of is obtained by multiplying the corresponding value of by a scalar c for each sample n of the signal.

### Signal Addition

Let and denote a pair of discrete-time signals, then the signalobtained by the addition of and is defined by: ****

### Signal Multiplication

Let and denote a pair of discrete-time signals, then the signalobtained by the addition of and  is defined by: 

This means that for each n¸ the value of is given by the product of the corresponding values ofand

## Generating Discrete-Time Sequences using MATLAB

Type the following code in the MATLAB m file editor and run the program.

**Program No: 01**

close all; clear all; clc;

N=10; % Declaring the domain of the signal

n=[-N:1:N]; % Signal indexing vector

**%% Unit Impulse**

impulse=zeros(1,length(n)); % generates zeros from 1 till length of n

impulse(n==0)=1; % plot 1 at n=0

stem(n,impulse); % plot discrete values of n versus values of impulse

axis([-N N -2 2]); % indicating limits of x-axis and y-axis

grid on; xlabel('Time-Index');

ylabel('Amplitude'); title('Unit Impulse');

**Program 02:**

%% Unit step

clear all; close all; clc;

N=10; n=[-N:1:N];

step=zeros(1,length(n)); % generates zeros from 1 till length of n

step(n>=0)=1; % plot ones at all positive values

figure;

stem(n,step); % plot discrete values of n versus values of stem

axis([-N N -2 2]); % indicating limits of x-axis and y-axis

grid on; xlabel('Time-Index'); ylabel('Amplitude');title('Unit Step');

**Program 03:**

%% Real exponential (Decaying)

Clear all; close all;

N=10; n=[-N:1:N];

A=1; % assigning a constant value

alpha=-0.1;

expo=A\*exp(alpha\*n); % returns the exponential for each element of n

figure;

stem(n,expo); % plot discrete values of n versus values of expo

axis([-N N 0 max(expo)]); % indicating limits of x-axis and y-axis

grid on

xlabel('Time-Index'); ylabel('Amplitude'); title('Decaying Exponential');

 **Program 04:**

%% Real exponential (Growing)

Clear all; clear all; clc ;

A=1; % assigning a constant value

N=10; n=[-N:1:N];

alpha=0.1;

expo2=A\*exp(alpha\*n); % returns the exponential for each element of n

figure;

stem(n,expo2); % plot discrete values of n versus the values of expo2

axis([-N N 0 max(expo2)]); % indicating limits of x-axis and y-axis

grid on

xlabel('Time-Index');

ylabel('Amplitude');

title('Growing Exponential');

**Program 05:**

%% Sinusoidal Signals

clear all; clear all; clc ;

A=1; N=10; n=[-N:1:N];

omega=pi/6; phi=0;

cosino=A\*cos(omega\*n+phi); % returns cosine for each element of n

figure; stem(n,cosino); % plot discrete values of n versus values of cosino

axis([-N N -1 1]); % indicating limits of x-axis and y-axis

grid on ; xlabel('Time-Index'); ylabel('Amplitude'); title('Cosine function');



**Program 06:**

%% Amplitude Scaling

%% Unit Impulse

Clear all; clear all; clc;

N=10; n=[-N:1:N];

impulse=zeros(1,length(n)); % generates zeros from 1 till length of n

impulse(n==0)=1; % plot 1 at n=0

impulse2=0.5\*impulse; % scaling previous impulse value

figure; subplot(1,3,1);

stem(n,impulse); % plot discrete values of n versus values of impulse

axis([-N N -2 2]); % indicating limits of x-axis and y-axis

xlabel('Time-Index'); ylabel('Amplitude'); title('Impulse'); grid on ;

subplot(1,3,2);

stem(n,impulse2); % plot discrete values of n versus values of impulse2

axis([-N N -2 2]); % indicating limits of x-axis and y-axis

xlabel('Time-Index'); ylabel('Amplitude'); title('0.5\*Impulse'); grid on;

impulse3=-2\*impulse; % scaling previous impulse values on negative axis

subplot(1,3,3); stem(n,impulse3); % plot discrete values of n versus values of impulse3

axis([-N N -2 2]); % indicating limits of x-axis and y-axis

xlabel('Time-Index'); ylabel('Amplitude'); title('-2\*Impulse'); grid on;



**Program 07:**

%% Addition of Signals

Clear all; clear all; clc;

N=10; % Declaring the domain of the signal

n=[-N:1:N]; % Signal indexing vector

step=zeros(1,length(n)); % generates zeros from 1 till length of n

step(n>=0)=1; % plot ones at all positive values

%% Sinusoidal Signals

A=1; omega=pi/6; phi=0;

cosino=A\*cos(omega\*n+phi); % returns cosine for each element of n

zn=step+cosino; % adding values of step and cosine as calculated before

figure; subplot(1,3,1);

stem(n,step); % plot discrete values of n versus values of step

axis([-N N -2 2]); % indicating limits of x-axis and y-axis

grid on ; xlabel('Time-Index'); ylabel('Amplitude'); title('Signal 1');

subplot(1,3,2);

stem(n,cosino); % plot discrete values of n versus values of cosino

axis([-N N -2 2]); % indicating limits of x-axis and y-axis

grid on; xlabel('Time-Index'); ylabel('Amplitude'); title('Signal 2');

subplot(1,3,3); stem(n,zn); % plot discrete values of n versus values of zn

axis([-N N -2 2]); % indicating limits of x-axis and y-axis

grid on; xlabel('Time-Index'); ylabel('Amplitude'); title('Sum');



**Program 08:**

%% Real exponential (Decaying)

Clear all; clear all; clc ;

A=1; % assigning a constant value

alpha=-0.1;

expo=A\*exp(alpha\*n); % returns the exponential for each element of n

%% Sinusoidal Signals

A=1;

omega=pi/6;

phi=0**;**

z2n=expo+cosino; % addition of expo and cosino values

figure; subplot(1,3,1);

stem(n,expo); % plot discrete values of n versus values of expo

axis([-N N -max(z2n) max(z2n)]); % indicating limits of x-axis and y-axis

grid on;xlabel('Time-Index');ylabel('Amplitude');title('Signal 1');

subplot(1,3,2);

stem(n,cosino); % plot discrete values of n versus values of cosino

axis([-N N -max(z2n) max(z2n)]); % indicating limits of x-axis and y-axis

grid on;xlabel('Time-Index');ylabel('Amplitude');title('Signal 2');

subplot(1,3,3);

stem(n,z2n); % plot discrete values of n versus values of z2n

axis([-N N -max(z2n) max(z2n)]); % indicating limits of x-axis and y-axis

grid on; xlabel('Time-Index');ylabel('Amplitude');title('Sum');



**Program 09:**

%% Multiplication of two signals

clear all; clear all; clc;

N=10; % Declaring the domain of the signal

n=[-N:1:N]; % Signal indexing vector

step=zeros(1,length(n)); % generates zeros from 1 till length of n

step(n>=0)=1; % plot ones at all positive values

%% Real exponential (Decaying)

A=1; % assigning a constant value

alpha=-0.1;

expo=A\*exp(alpha\*n); % returns the exponential for each element of n

pn=(expo).\*step;

figure; subplot(1,3,1);

stem(n,expo); % plot discrete values of n versus values of expo

axis([-N N 0 max(expo)]); % indicating limits of x-axis and y-axis

grid on ; xlabel('Time-Index'); ylabel('Amplitude'); title('Signal 1');

subplot(1,3,2);

stem(n,step); % plot discrete values of n versus values of step

axis([-N N 0 max(expo)]); % indicating limits of x-axis and y-axis

grid on ; xlabel('Time-Index'); ylabel('Amplitude'); title('Signal 2');

subplot(1,3,3); stem(n,pn); % plot discrete values of n versus values of pn

axis([-N N 0 max(expo)]); % indicating limits of x-axis and y-axis

grid on; xlabel('Time-Index'); ylabel('Amplitude'); title('Product');



## Exercises

For ,

1. Let and , plot
2. 
3. 
4. 
5. 
6. 

**Hint:** You can use the ones( ) and zeros( ) functions of MATLAB introduced earlier in the introduction to MATLAB.

1. Plot 5u(n)
2. Plot 2×sin(ω0n + π/6) where ω0 has the same value as given above.
3. Add the two signals generated in Problem 2 and Problem 3, and plot the result.
4. Generate the signal for A=2 and a = -0.5.
5. Let , and plot . Repeat the problem by varying the phase angle of to 2and respectively and comment on the result.

**EXERCISE MATLAB:**

Question 01:

For -10≤n≤10

Let and , plot

**Matalb Program for *x1[n]***

close all; clear all; clc;

N=10; % Declaring the domain of the signal

n=[-N:1:N]; % Signal indexing vector

%% Unit Impulse

N=10;

n=[-N:1:N];

impulse=zeros(1,length(n)); % generates zeros from 1 till length of n

impulse(n==0)=1; % plot 1 at n=0

impulse(n==1)=1; % plot 1 at n=1

impulse(n==2)=1; % plot 1 at n=2

stem(n,impulse); % plot discrete values of n versus values of impulse

axis([-N N -2 2]); % indicating limits of x-axis and y-axis

grid on; xlabel('Time-Index'); ylabel('Amplitude'); title('Unit Impulse');

**Matalb Program for *x2[n]***

close all; clear all; clc;

N=10; % Declaring the domain of the signal

n=[-N:1:N]; % Signal indexing vector

%% Unit Impulse

N=10; n=[-N:1:N];

impulse=zeros(1,length(n));

impulse(n==-1)=1; % plot 1 at n=-1

impulse(n==0)=-1; % plot 1 at n=0

impulse(n==1)=1; % plot 1 at n=1

impulse(n==2)=1; % plot 1 at n=2

stem(n,impulse); % plot discrete values of n versus values of impulse

axis([-N N -2 2]); % indicating limits of x-axis and y-axis

grid on ; xlabel('Time-Index'); ylabel('Amplitude');

title('Unit Impulse');

zn=step1+step2; % adding values of step1 and step 2

figure; subplot(1,3,3);

stem(n,zn); % plot discrete values of n versus values of zn

axis([-N N -2 2]); % indicating limits of x-axis and y-axis

grid on ; xlabel('Time-Index'); ylabel('Amplitude'); title('Sum');

***a. x1[n]+ x2[n]=??***

**Matlab Program:**

close all; clear all; clc;

N=10; % Declaring the domain of the signal

n=[-N:1:N]; % Signal indexing vector

%% Unit Impulse 1

impulse=zeros(1,length(n)); % generates zeros from 1 till length of n

impulse(n==0)=1; % plot 1 at n=0

impulse(n==1)=1; % plot 1 at n=1

impulse(n==2)=1; % plot 1 at n=2

impulse1=1\*impulse; % scaling previous impulse value

subplot(1,3,1);

stem(n,impulse1); % plot discrete values of n versus values of impulse

axis([-N N -2 2]); % indicating limits of x-axis and y-axis

grid on ; xlabel('Time-Index'); ylabel('Amplitude'); title('2\*impulse 1');

%% Unit Impulse 2

impulse=zeros(1,length(n)); % generates zeros from 1 till length of n

impulse(n==-1)=1; % plot 1 at n=-1

impulse(n==0)=-1; % plot 1 at n=0

impulse(n==1)=1; % plot 1 at n=1

impulse(n==2)=1; % plot 1 at n=2

impulse2=1\*impulse; %scaling previous value

subplot(1,3,2);

stem(n,impulse2); % plot discrete values of n versus values of impulse

axis([-N N -2 2]); % indicating limits of x-axis and y-axis

grid on ; xlabel('Time-Index'); ylabel('Amplitude'); title('Unit Impulse 2');

%% Addation of impulse 1 & 2

s=(impulse1)+(impulse2);

subplot(1,3,3);

stem(n,s); % plot discrete values of n versus values of zn

axis([-N N -3 3]); % indicating limits of x-axis and y-axis

grid on ; xlabel('Time-Index'); ylabel('Amplitude'); title('sum');



***b. 2x1[n]- x2[n]=??***

**Matlab Program**

close all; clear all; clc;

N=10; % Declaring the domain of the signal

n=[-N:1:N]; % Signal indexing vector

%% Unit Impulse 1

impulse=zeros(1,length(n)); % generates zeros from 1 till length of n

impulse(n==0)=1; % plot 1 at n=0

impulse(n==1)=1; % plot 1 at n=1

impulse(n==2)=1; % plot 1 at n=2

impulse2=2\*impulse; % scaling previous impulse value

subplot(1,3,1);

stem(n,impulse2); % plot discrete values of n versus values of impulse

axis([-N N -2 2]); % indicating limits of x-axis and y-axis

grid on ; xlabel('Time-Index'); ylabel('Amplitude'); title('2\*impulse 1');

%% Unit Impulse 2

impulse=zeros(1,length(n)); % generates zeros from 1 till length of n

impulse(n==-1)=1; % plot 1 at n=-1

impulse(n==0)=-1; % plot 1 at n=0

impulse(n==1)=1; % plot 1 at n=1

impulse(n==2)=1; % plot 1 at n=2

subplot(1,3,2);

stem(n,impulse); % plot discrete values of n versus values of impulse

axis([-N N -2 2]); % indicating limits of x-axis and y-axis

grid on; xlabel('Time-Index'); ylabel('Amplitude'); title('Unit Impulse 2');

% subtraction of 2\*impulse and impulse 2

s=impulse2-impulse;

subplot(1,3,3);

stem(n,s); % plot discrete values of n versus values of zn

axis([-N N -3 3]); % indicating limits of x-axis and y-axis

grid on ; xlabel('Time-Index'); ylabel('Amplitude');

title('2\*impulse 1-Unit Impulse 2');



***c. x1[n]+ 3x2[n]=??***

**Matlab Program:**

close all; clear all; clc;

N=10; % Declaring the domain of the signal

n=[-N:1:N]; % Signal indexing vector

%% Unit Impulse 1

impulse=zeros(1,length(n)); % generates zeros from 1 till length of n

impulse(n==0)=1; % plot 1 at n=0

impulse(n==1)=1; % plot 1 at n=1

impulse(n==2)=1; % plot 1 at n=2

impulse1=1\*impulse; % scaling previous impulse value.

subplot(1,3,1)

stem(n,impulse1); % plot discrete values of n versus values of impulse

axis([-N N -6 6]); % indicating limits of x-axis and y-axis

grid on ; xlabel('Time-Index'); ylabel('Amplitude'); title('impulse 1');

%% Unit Impulse 2

impulse=zeros(1,length(n)); % generates zeros from 1 till length of n

impulse(n==-1)=1; % plot 1 at n=-1

impulse(n==0)=-1; % plot 1 at n=0

impulse(n==1)=1; % plot 1 at n=1

impulse(n==2)=1; % plot 1 at n=2

impulse2=3\*impulse; % scaling previous impulse value

subplot(1,3,2);

stem(n,impulse2); % plot discrete values of n versus values of impulse

axis([-N N -6 6]); % indicating limits of x-axis and y-axis

grid on ; xlabel('Time-Index'); ylabel('Amplitude'); title('3\*impulse 2');

%% subtraction of 2\*impulse and impulse 2

s=(impulse1)+(impulse2);

subplot(1,3,3); stem(n,s); % plot discrete values of n versus values of zn

axis([-N N -6 6]); % indicating limits of x-axis and y-axis

grid on ; xlabel('Time-Index'); ylabel('Amplitude');

title('impulse 1 + 3\*impulse 2 ');



***d. x1[n]x2[n]=??***

**Matlab Program:**

close all; clear all; clc;

N=10; % Declaring the domain of the signal

n=[-N:1:N]; % Signal indexing vector

%% Unit Impulse 1

impulse=zeros(1,length(n)); % generates zeros from 1 till length of n

impulse(n==0)=1; % plot 1 at n=0

impulse(n==1)=1; % plot 1 at n=1

impulse(n==2)=1; % plot 1 at n=2

impulse1=1\*impulse; % scaling previous impulse value

subplot(1,3,1)

stem(n,impulse1); % plot discrete values of n versus values of impulse

axis([-N N -2 2]); % indicating limits of x-axis and y-axis

grid on ; xlabel('Time-Index'); ylabel('Amplitude'); title('impulse 1');

%% Unit Impulse 2

impulse=zeros(1,length(n)); % generates zeros from 1 till length of n

impulse(n==-1)=1; % plot 1 at n=-1

impulse(n==0)=-1; % plot 1 at n=0

impulse(n==1)=1; % plot 1 at n=1

impulse(n==2)=1; % plot 1 at n=2

impulse2=1\*impulse; % scaling previous value

subplot(1,3,2)

stem(n,impulse2); % plot discrete values of n versus values of impulse

axis([-N N -2 2]); % indicating limits of x-axis and y-axis

grid on ; xlabel('Time-Index'); ylabel('Amplitude'); title('impulse 2');

% multipalction

m=(impulse1).\*(impulse2); % product of unit impulse 1 and 2

subplot(1,3,3);

stem(n,m); % plot discrete values of n versus values of zn

axis([-N N -2 2]); % indicating limits of x-axis and y-axis

grid on ; xlabel('Time-Index'); ylabel('Amplitude');

title(' product of unit impulse 1 and 2 ');

****

***e. x1[n]u[n]=??***

**Matlab Program**

close all; clear all; clc

N=10; % Declaring the domain of the signal

n=[-N:1:N]; % Signal indexing vector

%% Unit Impulse 1

impulse=zeros(1,length(n)); % generates zeros from 1 till length of n

impulse(n==0)=1; % plot 1 at n=0

impulse(n==1)=1; % plot 1 at n=1

impulse(n==2)=1; % plot 1 at n=2

impulse1=1\*impulse; % scaling previous impulse value

subplot(1,3,1)

stem(n,impulse1); % plot discrete values of n versus values of impulse

axis([-N N -2 2]); % indicating limits of x-axis and y-axis

grid on ; xlabel('Time-Index'); ylabel('Amplitude'); title('impulse ');

%% unit step

step=zeros(1,length(n)); % generates zeros from 1 till length of n

step(n>=0)=1; % plot ones at all positive values

subplot(1,3,2)

stem(n,step); % plot discrete values of n versus values of stem

axis([-N N -2 2]);% indicating limits of x-axis and y-axis

grid on ; xlabel('Time-Index'); ylabel('Amplitude'); title('Unit Step');

%% Product step and impulse

pr=impulse.\*step;

subplot(1,3,3);

stem(n,pr);

axis([-N N -2 2]);% indicating limits of x-axis and y-axis

grid on ; xlabel('Time-Index'); ylabel('Amplitude'); title('product');



***Question 02:***

5u[n]=??

**Matlab Program**

close all; clear all; clc;

N=10; % Declaring the domain of the signal

n=[-N:1:N]; % Signal indexing vector

%% unit step

step=zeros(1,length(n)); % generates zeros from 1 till length of n

step(n>=0)=5; % plot ones at all positive values

stem(n,step); % plot discrete values of n versus values of stem

axis([-N N -6 6]);% indicating limits of x-axis and y-axis

grid on

xlabel('Time-Index');

ylabel('Amplitude');

title('Unit Step');



***Question 03:***

Plot 2×sin(ω0n + π/6) where ω0 has the same value as given above.

**Matlab Program**

%% Sinusoidal Signals

A=2; N=10; n=[-N:1:N];

omega=pi/6; phi=pi/6;

cosino=A\*sin(omega\*n+phi); % returns cosine for each element of n

figure; stem(n,cosino); % plot discrete values of n versus values of cosino

axis([-N N -3 3]); % indicating limits of x-axis and y-axis

grid on ; xlabel('Time-Index'); ylabel('Amplitude'); title('Cosine function');

***Question 04:***

Add the two signals generated in Problem 2 and Problem 3, and plot the result.

**Matlab Program**

close all; clear all; clc;

%% Sinusoidal Signals

A=2; N=10; % Declaring the domain of the signal

n=[-N:1:N]; % Signal indexing vector

omega=pi/6; phi=pi/6;

cosino=A\*sin(omega\*n+phi); % returns cosine for each element of n

subplot(1,3,1);

stem(n,cosino); % plot discrete values of n versus values of cosino

axis([-N N -8 8]); % indicating limits of x-axis and y-axis

grid on ; xlabel('Time-Index'); ylabel('Amplitude'); title('Cosine function');

%% unit step

step=zeros(1,length(n)); % generates zeros from 1 till length of n

step(n>=0)=5; % plot ones at all positive values

subplot(1,3,2);

stem(n,step); % plot discrete values of n versus values of stem

axis([-N N -8 8]);% indicating limits of x-axis and y-axis

grid on ; xlabel('Time-Index'); ylabel('Amplitude'); title('Unit Step');

%% Addation of Sinusoidal and unit signals

s=cosino+step;

subplot(1,3,3);

stem(n,s);

axis([-N N -8 8]);% indicating limits of x-axis and y-axis

grid on ; xlabel('Time-Index'); ylabel('Amplitude'); title('sum');

****

***Question 05***

Generate the signal for A=2 and a = -0.5.

**Matlab Program:**

close all; clear all; clc

% Real exponential (Decaying)

N=10; n=[-N:1:N]; A=1; % assigning a constant value

alpha=-0.5; expo=A\*exp(alpha\*n); % returns the exponential for each element of n

subplot(1,3,1)

stem(n,expo); % plot discrete values of n versus values of expo

axis([-N N 0 max(expo)]); % indicating limits of x-axis and y-axis

grid on ; xlabel('Time-Index'); ylabel('Amplitude'); title('Decaying Exponential');

%% Unit step

step=zeros(1,length(n)); % generates zeros from 1 till length of n

step(n>=0)=1; % plot ones at all positive values

subplot(1,3,2)

stem(n,step); % plot discrete values of n versus values of stem

axis([-N N -2 2]);% indicating limits of x-axis and y-axis

grid on ; xlabel('ime-Index'); ylabel('Amplitude'); title('Unit Step');

%% Product of Real exponential (Decaying) and unit step

pr=expo.\*step; %% Product of Real exponential (Decaying) and unit step

subplot(1,3,3); stem(n,pr);

axis([-N N -2 2]); %% indicating limits of x-axis and y-axis

grid on ; xlabel('Time-Index'); ylabel('Amplitude');

title('Product of Real exponential (Decaying) and unit step');



***Question 06:***

Let , and plot . Repeat the problem by varying the phase angle of to 2and respectively and comment on the result.

**Matlab Program**

close all; clear all; clc;

 %% Sinusoidal Signal 01

A1=2; N=10; n=[-N:1:N];

omega1=pi/3; phi1=pi/3;

cosino=A1\*cos(omega1\*n+phi1); % returns cosine for each element of n

subplot(1,3,1);

stem(n,cosino); % plot discrete values of n versus values of cosino

axis([-N N -3 3]); % indicating limits of x-axis and y-axis

grid on ; xlabel('Time-Index'); ylabel('Amplitude'); title('Cosine function 01');

%% Sinusoidal Signal 02

A2=2; omega=pi/6; phi=pi/3;

cosin1=A2\*cos(omega\*n+phi); % returns cosine for each element of n

subplot(1,3,2)

stem(n,cosin1); % plot discrete values of n versus values of cosino

axis([-N N -3 3]); % indicating limits of x-axis and y-axis

grid on ; xlabel('Time-Index'); ylabel('Amplitude'); title('Cosine function 02');

% sum Sinusoidal signal 01 & 02

s=cosino+cosin1; % sum

subplot(1,3,3);stem(n,s);

axis([-N N -4 4]); % indicating limits of x-axis and y-axis

grid on ; xlabel('Time-Index'); ylabel('Amplitude'); title('sum');

****

**Comparison of results when x1[n] has phase angle π/3, 2π/3, -π/3 respectively is given below**

****

****

****

**Lab 2 Continuous-time signals and operations on independent variables**

## OBJECTIVES

* To plot some elementary continuous-time signals using MATLAB,
* To perform operations on independent variables of continuous-time signals.

## THEORY

### Unit Step Signal

The unit step signal in continuous-time is defined as:



It should be noted that is discontinuous at .

### Unit Impulse Signal

The unit impulse signal in continuous-time is given as:



### Real Exponential Signal

A continuous-time real exponential signal is expressed as:



where A is the amplitude of the signal at and a is a real number. For , the signal decays while for , the signal grows exponentially.

### Sinusoidal Signal

The sinusoidal signal in continuous-time is given by the formula:



where A is the amplitude. If t is the time is the fundamental frequency, andis the phase angle.

### Reflection

Given a signal, its time reversed or reflected version is given by . The signal represents a reflected version of about .

### Time Scaling

Let denotes a continuous-time signal. Then the signal obtained by scaling the independent variable, time t, by a factor a is defined as



If a > 1, is a compressed version of. If 0 < a < 1, then is an extended version of.

### Time Shifting

Given a signal, its time-shifted version is defined by:



where is the time-shift. For, the waveform of is obtained by shifting towards the right, relative to the time-axis. If, is shifted to the left.

## Approximating Continuous-Time Signals using MATLAB

In this laboratory, we will study some elementary continuous-time signals and signal operations on the independent variable. It must be noted that MATLAB works on discrete time signals; hence, we will only be approximating the continuous-time signals by keeping a high sampling rate. Moreover, rather than the stem command used in the previous lab we will be using plot, as plot joins two consecutive samples with a line giving an appearance of a continuous-time signal. It can also be noted that in contrast to the previous lab where the independent variable was the sample number, we have used time as the independent variable. Moreover, rather than using a sampling interval of 0.01seconds which seems a more reasonable sampling rate, we have fixed the sampling interval to 0.011 seconds. This is done to avoid sampling the signal at t = 0, where the unit step function is undefined.

Type the following code in the MATLAB m file editor and run the program.



**Signal 01:**

close all; clear all; clc;

T=1;% Declaring the domain of the signal

t=[-T:0.011:T];% Time indexing vector

%% Unit step

stp=zeros(1,length(t)); % plotting zeros from 1 till t

stp(t>0)=1; % plotting ones at t>0

figure; plot(t,stp);

axis([-T T -2 2]); % indicating limits of x-axis and y-axis

grid on ; xlabel('Time');

ylabel('Amplitude'); title('Unit Step');

**Signal 02:**

%% Real exponential (Decaying)

close all; clear all; clc;

T=1;% Declaring the domain of the signal

t=[-T:0.011:T];% Time indexing vector

A=1; alpha=-2;

expo=A\*exp(alpha\*t); % returns the exponential for each element of t

figure; plot(t,expo);

axis([-T T 0 max(expo)]); % indicating limits of x-axis and y-axis

grid on;

xlabel('Time');

ylabel('Amplitude');

title('Decaying Exponential');

**Signal 03:**

%% Real exponential (Decaying) only for t>0

close all; clear all; clc;



T=1;% Declaring the domain of the signal

t=[-T:0.011:T];% Time indexing vector

A=1; alpha=-2;

stp=zeros(1,length(t)); % plotting zeros from 1 till t

stp(t>0)=1; % plotting ones at t>0

expo=A\*exp(alpha\*t); % returns the exponential for each element of t

newexpo=expo.\*stp; % element-by-element product of the arrays expo and stp

figure; plot(t,newexpo);

axis([-T T 0 max(newexpo)]); % indicating limits of x-axis and y-axis

grid on; xlabel('Time'); ylabel('Amplitude'); title('Decaying Exponential for t>0');

**Signal 04**

%% Sinusoidal Signals

close all; clear all; clc;

T=1;% Declaring the domain of the signal

t=[-T:0.01:T];% Time indexing vector

A=1; omega=10;

phi=0; cosino=A\*cos(omega\*t+phi); % returns the cosine value for all values of t

figure; plot(t,cosino);

axis([-T T -1 1]); % indicating limits of x-axis and y-axis

grid on ; xlabel('Time'); ylabel('Amplitude');

title('Cosine function');

figure;

plot(omega\*t,cosino);

axis([-10\*T 10\* T -1.5 1.5]); % indicating limits of x-axis and y-axis

grid on ; xlabel('Angle'); ylabel('Amplitude'); title('Cosine function');



**Signal 05:**

%% Time shift

close all; clear all; clc;

T=1;% Declaring the domain of the signal

t=[-T:0.01:T];% Time indexing vector

%% Unit step shifted right (delayed by 0.2 seconds)

delayedstep=zeros(1,length(t)); % plotting zeros from 1 till length of t

delayedstep(t-0.2>0)=1; % delaying by 0.2 sec and plotting ones onwards

figure; plot(t,delayedstep);

axis([-T T -2 2]); % indicating limits of x-axis and y-axis

grid on; xlabel('Time');ylabel('Amplitude');

title('Delayed Unit Step');

**Signal 06**

%% Cosine wave delayed by 0.1 second

close all; clear all; clc;

T=1; % Declaring the domain of the signal

t=[-T:0.01:T];% Time indexing vector

A=0.5;

omega=10;

phi=-omega\*0.1;

delayedcosino=A\*cos(omega\*t+phi); % returns the cosine value for all values of t

figure; plot(t,delayedcosino);

axis([-T T -1 1]); % indicating limits of x-axis and y-axis

grid on ; xlabel('Time');label('Amplitude');

title('Delayed Cosine function');

figure; plot(omega\*t,delayedcosino);

axis([-10\*T 10\*T -1 1]); % indicating limits of x-axis and y-axis

grid on ; xlabel('Angle'); ylabel('Amplitude');

title('Delayed Cosine function');



**A compression of Cosine wave delayed by 0.1 second and without delay {Additional View}**

****

**Signal 07**

%% Time reversal (Reflection)

close all; clear all; clc;

 T=1; % Declaring the domain of the signal

t=[-T:0.01:T]; % Time indexing vector

t2=-t; % reversing the time axis

A=1; alpha=-2;

expo2=A\*exp(alpha\*t); % returns the exponential for each element of t

figure; plot(t2,expo2);

axis([-T T 0 max(expo2)]); % indicating limits of x-axis and y-axis

grid on ; xlabel('Time'); ylabel('Amplitude');

title('Time-reversed Decaying Exponential');

**A comparison of Decaying Exponential Signal with and without time reversal {Additional View}**



**Signal 08**

%% Time stretching (t/2)

close all; clear all; clc;

T=1;% Declaring the domain of the signal

t=[-T:0.01:T];% Time indexing vector

%%t/2

t2=2\*t; % up-scaling time axis

A=1; alpha=-2;

expo2=A\*exp(alpha\*t); % returns the exponential for each element of t

figure; plot(t2,expo2);

axis([-2\*T 2\*T 0 max(expo2)]); % indicating limits of x-axis and y-axis

grid on; xlabel('Time'); ylabel('Amplitude'); title('Decaying Exponential');

**A comparison of Decaying Exponential Signal with and without time stretching {Additional View}**

**Signal 09**

%%2t (Time Squeezing)

close all; clear all; clc;

T=1;% Declaring the domain of the signal

t=[-T:0.01:T];% Time indexing vector

%%2t

t2=t/2; % down-scaling time axis

A=1; omega=10; phi=0;

cosino=A\*cos(omega\*t+phi); % returns the cosine value for all values of t

figure; plot(t2,cosino);

axis([-T/2 T/2 -1 1]); % indicating limits of x-axis and y-axis

grid on ; xlabel('Time');

ylabel('Amplitude');

title('Cosine function');

**A comparison of Cosine Signal with and without time Squeezing {Additional View}**



**Signal 10**

%%3/2\*t {Scaling}

close all; clear all; clc;

T=1;% Declaring the domain of the signal

t=[-T:0.01:T];% Time indexing vector

%%3/2\*t

t2=2\*t/3; % scaling time axis

A=1; omega=10; phi=0;

cosino=A\*cos(omega\*t+phi); % returns the cosine value for all values of t

figure; plot(t2,cosino);

axis([-2\*T/3 2\*T/3 -1 1]); % indicating limits of x-axis and y-axis

grid on; xlabel('Time');

ylabel('Amplitude'); title('Cosine function');

## Exercises

For  and a sampling rate of 0.011 seconds, plot the following.

1. .
2. 
3. 
4. delayed by 0.6 seconds.

**Part (D) e-t delayed by 1 second**

Matlab Program:

close all; clear all; clc;

T=3;% Declaring the domain of the signal

t=[-T:0.011:T];% Time indexing vector

A=1; alpha=-1;

delayedexpo=A\*exp(alpha\*t); % returns the exponential for each element of t

figure; plot(t+1,delayedexpo);

axis([-2\*T 2\*T 0 max(delayedexpo)]); % indicating limits of x-axis and y-axis

grid on ; xlabel('Time'); ylabel('Amplitude');

title('Delayed Decaying Exponential');



**A compression of original and Delayed Decaying Exponential signals {Additional View}**



**Part (B) Plot e-t u(-t)**

Matlab Program

close all; clear all; clc;

T=3;% Declaring the domain of the signal

t=[-T:0.011:T];% Time indexing vector

A=1; alpha=-1;

%% Exponential Signal

expo=A\*exp(alpha\*t); % returns the exponential for each element of t

subplot(1,3,1);

plot(t,expo);

axis([-T T 0 max(expo)]); % indicating limits of x-axis and y-axis

grid on ; xlabel('Time'); ylabel('Amplitude'); title('Decaying Exponential');

%% Unit step with time reversal

stp=zeros(1,length(t)); % plotting zeros from 1 till t

stp(t<0)=1; % plotting ones at t<0

subplot(1,3,2); plot(t,stp);

axis([-T T -2 2]); % indicating limits of x-axis and y-axis

grid on ; xlabel('Time'); ylabel('Amplitude'); title('Unit Step');

%% Product of Unit step & Exponential Signal

pr=stp.\*expo; %% Product formula

subplot(1,3,3); plot(t,pr);

axis([-T T -20 20]); % indicating limits of x-axis and y-axis

grid on; xlabel('Time'); ylabel('Amplitude');

title('Product of Unit step & Exponential Signal');

**Part (a) Plot (t-1) u(t-3)**

Matlab Program

close all; clear all; clc;

%Unit step function with 0.3 sec delay

T=1;% Declaring the domain of the signal

t=[-T:0.011:T];% Time indexing vector

stp=zeros(1,length(t)); % plotting zeros from 1 till t

stp(t>0.3)=1; % plotting ones at t>0

subplot(1,3,1); plot(t,stp);

axis([-T T -2 2]); % indicating limits of x-axis and y-axis

grid on ; xlabel('Time'); ylabel('Amplitude'); title('u(t-3)');

%Function for t-1

z=t-1; %% as in Question

subplot(1,3,2); plot(t,z);

axis([-T T -2 2]); % indicating limits of x-axis and y-axis

grid on ; xlabel('Time'); ylabel('Amplitude'); title('t-1');

% product of Unit step function with 0.3 sec delay and "t-1" function

pr=stp.\*z;

subplot(1,3,3); plot(t,pr);

axis([-T T -2 2]); % indicating limits of x-axis and y-axis

grid on ; xlabel('Time'); ylabel('Amplitude'); title('(t-1)u(t-3)');



**Part (C) Plot **

Matlab Program

close all; clear all; clc;

%Unit step function with 0.3 sec delay

T=1;% Declaring the domain of the signal

t=[-T:0.011:T];% Time indexing vector

stp=zeros(1,length(t)); % plotting zeros from 1 till t

stp(t>0.3)=1; % plotting ones at t>0

subplot(1,3,1); plot(t,stp);

axis([-T T -2 2]); % indicating limits of x-axis and y-axis

grid on ; xlabel('Time'); ylabel('Amplitude'); title('u(-t+0.3)');

%Function for t-1

z=2\*t+0.5; %% as in Question

subplot(1,3,2); plot(t,z);

axis([-T T -2.5 2.5]); % indicating limits of x-axis and y-axis

grid on ; xlabel('Time'); ylabel('Amplitude'); title('2t+0.5');

% product of Unit step function with 0.3 sec delay and "t-1" function

pr=stp.\*z;

subplot(1,3,3); plot(t,pr);

axis([-2\*T 2\*T -4 4]); % indicating limits of x-axis and y-axis

grid on ; xlabel('Time'); ylabel('Amplitude'); title('(2t+0.5)u(-t+0.3)');



**Lab 3 Constructing periodic signals from harmonically related sinusoidal signals**

## OBJECTIVES:

To generate periodic signals by combining harmonically related sinusoidal signals.

## THEORY

A signal is periodic if, for some positive value of T,  for all t. The fundamental period of is the minimum positive nonzero value of t for which the above equation is satisfied.

The sinusoidal signal and the complex exponential signal are both periodic with fundamental frequency and fundamental period. Associated with the complex exponential signal is the set of harmonically related complex exponentials given as:



A linear combination of harmonically related complex exponentials of the form



is also periodic with period T. In the above equation, the term for k = 0 is a constant, while the terms for both k = 1 and k = -1have fundamental frequency equal to  and are referred to as fundamental components or the first harmonic components. In general, the components for k = N and k = -N are called Nth harmonic components.

## Constructing periodic signals from harmonically related signals in MATLAB:

Let 

where , , , .

The above equation can be re-written by collecting the harmonic components with same fundamental frequencies as follows:



Using Euler’s theorem:



In the following code, we will add these harmonically related components one by one to see the effect of adding each component on the shape of the resulting signal.

**Matlab Program 01:**

close all; clear all;

% time indexing vector

t=[-5:0.01:5]; h0=1;

% returns the cosine value for all values of t

h1=0.5\*cos(2\*pi\*t);

h2=cos(4\*pi\*t);

h3=2/3\*cos(6\*pi\*t);

subplot(2,2,1); plot(t,h0);

grid on; xlabel('Time'); ylabel('h0');

subplot(2,2,2); plot(t,h0+h1);

grid on; xlabel('Time'); ylabel('h0+h1');

subplot(2,2,3); plot(t,h0+h1+h2);

grid on; xlabel('Time'); ylabel('h0+h1+h2');

subplot(2,2,4); plot(t,h0+h1+h2+h3);

grid on; xlabel('Time'); ylabel('h0+h1+h2+h3');



*Type the following code and execute the program:*

close all; clear all; clc;

t=[-5:0.01:5]; % time indexing vector

x=zeros(1,length(t)); % plotting zeros from 1 till length of t

NumOfHarmonics=10;

% executing a statement a predetermined number of times

for n=1:2:NumOfHarmonics % indicating the starting and ending of loop

 x=x+4/pi\*1/n\*sin(n\*pi\*t/2);

end

plot(t,x);



**Lab 4 Convolution**

## OBJECTIVES:

To perform convolution of various discrete time signals and to study the properties of convolution sum.

## THEORY

Let a system is given by, where be the input signal and be the output signal. Let be the impulse response of the system i.e. . Then the response  of the system to any arbitrary input can be determined using the convolution sum as follows:



The convolution operation is

* Commutative, i.e.;
* Distributive i.e. ; and
* Associative i.e. 

## Convolution in MATLAB

In MATLAB, convolution of two signals can be performed by the function conv(h, x). If h is a vector of length M and x is a vector of length N, then the length of the output vector is M + N – 1.

h is a vector of length M

x is a vector of length N

y is a vector of length M + N - 1

Type and run the following code in MATLAB:

close all; clear all; clc;

% impulse response

h=[3 2 1 -2 1 0 -4 0 3];

% Sample number where origin exists

org\_h = 1;

nh=[0 : length(h)-1]- org\_h + 1; % indicating the length of sample

subplot(3,1,1); stem(nh,h); xlim([nh(1)-1 nh(end)+1]); grid on;

xlabel('Time index n'); ylabel('Amplitude'); title('Impulse Response h(n)');

% Input signal x(n)

x = [1 -2 3 -4 3 2 1]; % input sequence

org\_x = 1; % Sample number where origin exists

nx = [0 : length(x)-1]- org\_x + 1; % indicating the length of sample

subplot(3,1,2); stem(nx,x); xlim([nx(1)-1 nx(end)+1]); grid on;

xlabel('Time index n'); ylabel('Amplitude'); title('Input Signal x(n)');

% Output obtained by convolation

y = conv(h,x);

ny = [nh(1)+ nx(1) : nh(end)+nx(end)]; % indicating the length of convolved sample

subplot(3,1,3); stem(ny,y); xlim([ny(1)-1 ny(end)+1]);

grid on; xlabel('Time index n'); ylabel('Amplitude'); title('Output Obtained by Convolution');



**Exercises**

Write MATLAB code to perform convolution on the following signals:

1. x[n]=2-n for 0<=n<=5

 h[n]= (………………,0,0,1,1,1,0,0,………)

2. x[n]=[3 2 1]

 h [n]=[0.5 1 0.5]

3. x[n]=[1 2 3 1 2 3 1 2 3]

 h [n]=[1 -1]

**SOLUTION**

**Write MATLAB code to perform convolution on the following signals:**

**Question 01**

*x[n]=2-n for 0≤n≤5*

*h[n]= (………………,0,0,1,1,1,0,0,………)*

Matlab Program

% Discrete signal h[n]

% impulse response

h=[0,0,1,1,1];

% Sample number where origin exists

org\_h = 1;

nh=[0 : length(h)-1]- org\_h + 1; % indicating the length of sample

subplot(3,1,1); stem(nh,h); xlim([nh(1)-1 nh(end)+1]); grid on;

xlabel('Time index n'); ylabel('Amplitude'); title('Impulse Response h(n)');

% Input signal x(n)

n=0:1:5; x=2-n;

org\_x = 1; % Sample number where origin exists

nx = [0 : length(x)-1]- org\_x + 1; % indicating the length of sample

subplot(3,1,2), stem(nx,x); xlim([nx(1)-1 nx(end)+1]); grid on;

xlabel('Time index n'); ylabel('Amplitude'); title('Input Signal x(n)');

% Output obtained by convolation

y = conv(h,x);

ny = [nh(1)+ nx(1) : nh(end)+nx(end)]; % indicating the length of convolved sample

subplot(3,1,3); stem(ny,y); xlim([ny(1)-1 ny(end)+1]); grid on;

xlabel('Time index n'); ylabel('Amplitude');

title('Output Obtained by Convolution');

**Question 02**

*x[n]=[3 2 1]*

*h [n]=[0.5 1 0.5]*

% Discrete signal h[n]

% impulse response

h=[0.5 1 0.5];

% Sample number where origin exists

org\_h = 1;

nh=[0 : length(h)-1]- org\_h + 1; % indicating the length of sample

subplot(3,1,1); stem(nh,h); xlim([nh(1)-1 nh(end)+1]); grid on;

xlabel('Time index n'); ylabel('Amplitude'); title('Impulse Response h(n)');

% Input signal x(n)

x = [3 2 1]; % input sequence

org\_x = 1; % Sample number where origin exists

nx = [0 : length(x)-1]- org\_x + 1; % indicating the length of sample

subplot(3,1,2); stem(nx,x); xlim([nx(1)-1 nx(end)+1]); grid on;

xlabel('Time index n'); ylabel('Amplitude'); title('Input Signal x(n)');

% Output obtained by convolation

y = conv(h,x);

ny = [nh(1)+ nx(1) : nh(end)+nx(end)]; % indicating the length of convolved sample

subplot(3,1,3); stem(ny,y); xlim([ny(1)-1 ny(end)+1]); grid on;

xlabel('Time index n'); ylabel('Amplitude');

title('Output Obtained by Convolution');

**Question 03:**

*x[n]=[1 2 3 1 2 3 1 2 3]*

*h [n]=[1 -1]*

% Discrete signal h[n]

% impulse response

h=[1 -1];

% Sample number where origin exists

org\_h = 1;

nh=[0 : length(h)-1]- org\_h + 1; % indicating the length of sample

subplot(3,1,1); stem(nh,h); xlim([nh(1)-1 nh(end)+1]); grid on;

xlabel('Time index n'); ylabel('Amplitude'); title('Impulse Response h(n)');

% Input signal x(n)

x = [1 2 3 1 2 3 1 2 3]; % input sequence

org\_x = 1; % Sample number where origin exists

nx = [0 : length(x)-1]- org\_x + 1; % indicating the length of sample

subplot(3,1,2); stem(nx,x); xlim([nx(1)-1 nx(end)+1]); grid on;

xlabel('Time index n'); ylabel('Amplitude'); title('Input Signal x(n)');

% Output obtained by convolation

y = conv(h,x);

ny = [nh(1)+ nx(1) : nh(end)+nx(end)]; % indicating the length of convolved sample

subplot(3,1,3); stem(ny,y); xlim([ny(1)-1 ny(end)+1]); grid on;

xlabel('Time index n'); ylabel('Amplitude');

title('Output Obtained by Convolution');

**Lab 5 Properties of continuous-time Fourier series-I**

## OBJECTIVE:

*To study and verify various properties of Fourier series for continuous time signals.*

## THEORY

In Lab 3, we studied Fourier series and learnt how periodic signals can be constructed by adding harmonically related complex exponentials. In this and the next lab, we will study various properties of Fourier series for continuous-time periodic signals.

Let and denote two periodic signals with period T having Fourier series coefficients denoted by and respectively, i.e.

 and 

Then:

### Linearity

###

### Time Shifting



### Time Reversal



### Time Scaling



## Verifying the properties using MATLAB

Type the following code and execute:

**Program 01**

% Title: Properties of Continuous Time Fourier series, Linearity

close all; clear all; clc;

% Generation of 1st signal xt

t=-1.5:0.005:1.5; % time indexing vector

xcos=cos(2\*pi\*t); % returns cosine for all values of t

xt=xcos>0;

subplot(3,1,1); plot(t,xt);

set(gca,'ylim',[-0.1 1.1]);% sets the named properties to specified values on the object identified by gca

grid on; xlabel('t'); ylabel('x(t)');

title('(xt)Peridoic Square Wave(T=1, T1=0.250)')

% Generation of 2nd signal yt

T=1; T1=0.125; lenT=T/0.005;

ytemp=zeros(1,lenT); % plotting zeros from 1 till value of lenT

lenT1=T1/0.005;

ytemp(round(lenT/2)-lenT1:round(lenT/2)+lenT1-1)=ones(1,2\*lenT1);

yt=[ytemp ytemp ytemp 0];

% The last 0 added to make the size of yt equal to length(t)

subplot(3,1,2);plot(t,yt);

set(gca,'ylim',[-0.1 1.1]); % sets the named properties to specified values on the object identified by gca

grid on; xlabel('t');ylabel('y(t)');

title('(yt)Periodic Square Wave (T=1, T1=0.125)')

% Sum of 1st and 2nd signal

z1t=xt+yt;

subplot(3,1,3); plot(t,z1t);

set(gca,'ylim',[-0.1 2.1]); % sets the named properties to specified values on the object identified by gca

grid on ; xlabel('t');ylabel('y(t)'); title('xt+yt')

**Program 02:**

close all; clear all; clc;

% FS coefficients of periodic square waves

k=-50:50; T1=0.25; T=1; t=[-T:0.001:T];

ak=sin(k\*2\*pi\*(T1/T))./(k\*pi); % returns sine value of angle

% Manual correction for a0 -> ak(51)

ak(k==0)=2\*T1/T; T1=0.125;

bk=sin(k\*2\*pi\*(T1/T))./(k\*pi);

% Manual correction for b0 -> bk(51)

bk(k==0)=2\*T1/T;

% Application of linearity property of FS

ck=ak+bk;

% Reconstruction with M=50

w0=2\*pi/T;

zt=zeros(1,length(t));

for k=-50:50

 zt=zt + ck(k+51)\*exp(j\*k\*w0\*t);

end

figure;

plot(t,real(zt));grid;xlabel('t');ylabel('z(t)=x(t)+y(t)')

title('Reconstruction from ak+bk''s with 101 terms')



**Program 03:**

close all; clear all; clc;

%%Title: Properties of Continuous Time Fourier Series

%%Time Shifting

%Generation of periodic square wave

t=-1.5:0.005:1.5;

xcos=cos(2\*pi\*t); xt=xcos>0;

subplot(2,1,1); plot(t,xt);

grid on; xlabel('t'); ylabel('x(t)'); title('Periodic Square Wave (T=1, T1=0.25)')

set(gca,'ylim',[-0.2 1.2]);

% FS coefficients of periodic square wave

k=-50:50;

T=1;T1=0.25;

ak=sin(k\*2\*pi\*(T1/T))./(k\*pi);

% Manual correction for a0 -> ak(51)

ak(k==0)=2\*T1/T;

%Amount of time shift

t0=0.25;

% FS coefficients of the time shifted signal

w0=2\*pi/T;

bk=ak.\*exp(-j\*k\*w0\*t0);

% Reconstruction from bk's with 101 terms (M=50)

yt=zeros(1,length(t));

for k=-50:50

 yt=yt + bk(k+51)\*exp(j\*k\*w0\*t);

end

subplot(2,1,2); plot(t,real(yt));

grid on ;xlabel('t');ylabel('y(t)=x(t-0.5)');

title('Reconstruction from bk's with 101 terms')

set(gca,'ylim',[-0.2 1.2]); % sets the named properties to specified values on the object identified by gca

****

**Lab 6 Properties of continuous-time Fourier series-II**

## OBJECTIVE:

*To study and verify various properties of Fourier series for continuous time signals.*

## THEORY

This lab is a continuation of Lab 5, where we studied some properties of Fourier series for continuous time signals. In this lab we are going to study to two more properties namely, multiplication and conjugation.

Let and denote two periodic signals with period T having Fourier series coefficients denoted by and, respectively. Then

### Multiplication



It can be observed that the right side of the above equation is actually the convolution sum of the Fourier coefficients and.

### Conjugation and conjugate symmetry



## Verifying the properties using MATLAB

Type the following programs and execute:

**Program 01:**

% Title: Properties of Continuous Time Fourier Series

% Multiplication

close all; clear all; clc;

% Generation of 1Hz cosine

t=-1.5:0.005:1.5; % time indexing vector

xt=cos(2\*pi\*t);

subplot(3,1,1); plot(t,xt);

grid on; xlabel('Time');ylabel('x(t)');title('1 Hz Cosine');

set(gca,'ylim',[-1.2 1.2]);

% Generation of periodic square wave

yt=xt>0;

subplot(3,1,2);

plot(t,yt);

grid on; xlabel('Time'); ylabel('y(t)'); title('Periodic Square Wave (T=1, T1=0.25)')

set(gca,'ylim',[-1.2 1.2]);

% FS coefficients of periodic square wave

k=-50:50; T=1; T1=0.25;

ak=sin(k\*2\*pi\*(T1/T))./(k\*pi);

% Manual correction for a0 -> ak(51)

ak(51)=2\*T1/T;

% FS coefficients of 1Hz cosine (over k=-1..1)

bk=zeros(1,3); bk(1)=0.5;bk(3)=0.5;

ck=conv(ak,bk);

ck(103)=[];ck(1)=[];

% Reconstruction from ck's with 101 terms (M=50)

w0=2\*pi/T;

zt=zeros(1,length(t));

for k=-50:50

 zt=zt + ck(k+51)\*exp(j\*k\*w0\*t);

end

set(gcf,'defaultaxesfontsize',8)

subplot(3,1,3);plot(t,real(zt))

grid on; xlabel('t'); ylabel('z(t)=x(t)\*y(t)'); title('Reconstruction from ck's')

set(gca,'ylim',[-1.2 1.2]);



**Program 02:**

close all; clear all; clc;

% Title: Properties of Continuous Time Fourier Series

% Conjugation and Conjugate Symmetry

% Generation of 1Hz cosine (Real part of our signal)

t=-1.5:0.005:1.5;

xt=cos(2\*pi\*t);

subplot(2,2,1); plot(t,xt);

grid on; ylabel('x(t)');title('Real part of z(t), 1 Hz Cosine')

set(gca,'ylim',[-1.2 1.2]);

% Generation of periodic square wave (Imaginary part of our signal)

yt=xt>0;

subplot(2,2,2); plot(t,yt);

grid on; ylabel('y(t)'); title('Imag. part of z(t), Periodic Square Wave')

set(gca,'ylim',[-1.2 1.2]);

% Our perodic complex valued signal

zt=xt+j\*yt;

subplot(2,2,3);plot(t,real(zt));

grid on; xlabel('t'); ylabel('Re[z(t)]');

title('Real part of Reconstruction from dk''s')

set(gca,'ylim',[-1.2 1.2]);

% FS coefficients of 1Hz cosine, xt (over k=-50:50)

ak=zeros(1,101);

ak(50)=0.5;ak(52)=0.5;

% FS coefficients of periodic square wave, yt

k=-50:50;

T=1;T1=0.25;

bk=sin(k\*2\*pi\*(T1/T))./(k\*pi);

% Manual correction for a0 -> ak(51)

bk(51)=2\*T1/T;

% FS coefficients of zt (using linearty property of FS)

ck=ak+j\*bk;

% Flip ck's and conjugate

dk=conj(fliplr(ck));

% Reconstruction from dk's with 101 terms (M=50)

w0=2\*pi/T;zrt=zeros(1,length(t));

for k=-50:50

 zrt=zrt + dk(k+51)\*exp(j\*k\*w0\*t);

end

set(gcf,'defaultaxesfontsize',8)

subplot(2,2,4); plot(t,imag(zrt))

grid on; xlabel('t'); ylabel('Im[zr(t)]');

title('Imag. part of Reconstruction from dk''s')

set(gca,'ylim',[-1.2 1.2]);



**Lab 7 Studying z Transform Practically**

**OBJECTIVE**

To study z-transform practically using MATLAB

**THEORY**

In mathematics and signal processing, the **Z-transform** converts a discrete time-domain signal, which is a sequence of real or complex numbers, into a complex frequency-domain representation.

The Z-transform, like many other integral transforms, can be defined as either a one-sided or two-sided transform.

**Bilateral Z-transform**

The bilateral or two-sided Z-transform of a discrete-time signal x[n] is the function X(z) defined as

![X(z) = \mathcal{Z}\{x[n]\} = \sum_{n=-\infty}^{\infty} x[n] z^{-n} \ ]()

**Unilateral Z-transform**

Alternatively, in cases where x[n] is defined only for n ≥ 0, the single-sided or unilateral Z-transform is defined as

![X(z) = \mathcal{Z}\{x[n]\} =  \sum_{n=0}^{\infty} x[n] z^{-n} \ ]()

The z-transform is the discrete-time counter-part of the Laplace transform and a generalization of the Fourier transform of a sampled signal. The z-transform allows insight into the transient behavior, the steady state behavior, and the stability of discrete-time systems. A working knowledge of the z-transform is essential to the study of digital filters and systems

**MATLAB Code**

syms z n

ans=ztrans(1/4^n)

return

ans =

4\*z/ (4\*z-1)

**Inverse Z-Transform**

X(n) = Z-1 [ X(Z)]

X(Z) **=** 3\*Z / (Z+1)

**MATLAB Code**

syms Z n

ans=iztrans(3\*Z/(Z+1))

return

ans =3\*(-1) ^n

**Stability of a system**

A system is said to be stable if the poles lie inside of a unit circle on the z-plane.

**Pole Zero Diagrams for a Function in Z Domain**

Z plane command computes and display the pole-zero diagram of Z function.

zplane(b,a)

To display the pole value, use **root(a)**

To display the zero value, use **root(b)**

**X(Z) = [Z-2 + Z-1 ] / [1-2Z-1+3Z-2]**

**MATLAB Code**

b= [0 1 1]

a= [1 -2 +3]

roots(a)= 1.0000 + 1.4142i

 1.0000 - 1.4142i

roots(b)= -1

zplane(b,a)

****

**Frequency Response**

The Freqz function computes and display the frequency response of given Z- Transform of the function

freqz(b,a,npt,Fs)

b= Coeff. Of Numerator; a= Coeff. Of Denominator; Fs= Sampling Frequency

Npt= no. of free points between and Fs/2

**X(Z) = [2+ 5Z-1+9Z-2+5Z-3+3Z-4]/ [5+ 45Z-1+2Z-2+Z-3+Z-4]**

**MATLAB Code**

b= [2 5 9 5 3]

a= [5 45 2 1 1]

freqz(b,a)

**Exercise:**

1. Using zplane command plot the poles and zeros of the given z domain transfer function:



Is this system stable? Why?

****

**Ans.**

clearall

a=[1 -1.6180 1];

b=[1 -1.5161 .878];

roots(a)

ans =

 0.8090 + 0.5878i

 0.8090 - 0.5878i

roots(b)

ans =

 0.7581 + 0.5508i

 0.7581 - 0.5508i

zplane(b,a)

2. Using freqz function compute and display the frequency response of given z-transform of the



a=[1 -1.6180 1]; b=[1 -1.5161 .878];

freqz(b,a)



**Lab 8 Factored form of z Transform and its ROC**

**OBJECTIVE:**

To compute factored form of z transform and determine its ROC

**THEORY:**

**The Region of Convergence (ROC)**

The z-transform is an infinite power series, it exists only for those values of the variable z for which the series converges to a finite sum. The region of convergence (ROC) of X(z) is the set of all the values of z for which X(z) attains a finite computable value.

**Rational, Factored Forms of z-transform**

The function tf2zp can be used to determine the zeros and poles of a rational z-transform G(z) . The program statement to use is [z, p, k] = tf2zp(num,den) where num and den are row vectors containing the coefficients of the numerator and denominator polynomials of G(z) in ascending powers of z−1 and the output file contains the gain constant k and the computed zeros and poles given as column vectors z and p, respectively. The reverse process of converting a z-transform given in the form of zeros, poles, and the gain constant to a rational form can be implemented using the function zp2tf. The program statement to use is

[num,den] = zp2tf(z,p,k).

The factored form of the z-transform can be obtained from the zero-pole description using the function sos = zp2sos(z,p,k). The function computes the coefficients of each second-order factor given as an L × 6 matrix sos where



where the Lth row contains the coefficients of the numerator and the denominator of the Lth second-order factor of the z-transform G(z):



**MATLAB Code**

The following MATLAB code expresses the z-transform in factored form and helps to tell its ROC of the given z-transform:



**Program**

close all; clear all; clc;

num = input('Type in the numerator coefficients =');%1

den = input('Type in the denominator coefficients =');%3 %Conversion from rational to Factored form

[z,p,k]=tf2zp(num,den);

disp('Zeros are at');

disp(z);

disp('Poles are at');

disp(p);

disp('Gain Constant');

disp(k); %Determination of radius of the poles

radius=abs(p);

disp('Radius of the poles');

disp(radius); %Pole Zero Map using numerator and denominator coefficients

zplane(num,den) %Conversion from factored to second ordered factored

sos=zp2sos(z,p,k);

disp('Second Order Sections');

disp(sos);

**Ans.**

Zeros are at

 -4.0000

 -2.0000

 -1.0000 + 1.0000i

 -1.0000 - 1.0000i

Poles are at

 -3.2361

 1.2361

 0.5000 + 0.8660i

 0.5000 - 0.8660i

Gain Constant

 0.6667

Radius of the poles

 3.2361

 1.2361

 1.0000

 1.0000

Second Order Sections

 0.6667 4.0000 5.3333 1.0000 2.0000 -4.0000

 1.0000 2.0000 2.0000 1.0000 -1.0000 1.0000



**Exercise:**

1- Write a MATLAB program to compute and display the poles and zeros, to compute and display the factored form, and to generate the pole-zero plot of a z-transform. Determine its ROC.



**Program**

close all; clear all; clc;

num = input('Type in the numerator coefficients =');%1

den = input('Type in the denominator coefficients =');%3 %Conversion from rational to Factored form

[z,p,k]=tf2zp(num,den);

disp('Zeros are at');

disp(z); disp('Poles are at');

disp(p); disp('Gain Constant');

disp(k); %Determination of radius of the poles

radius=abs(p); disp('Radius of the poles');

disp(radius); %Pole Zero Map using numerator and denominator coefficients

zplane(num,den) %Conversion from factored to second ordered factored

sos=zp2sos(z,p,k); disp('Second Order Sections'); disp(sos);

**Ans.**

Zeros are at

 -1.0000 + 1.4142i

 -1.0000 - 1.4142i

 -0.2500 + 0.6614i

 -0.2500 - 0.6614i

Poles are at

 -0.6114 + 0.4314i

 -0.6114 - 0.4314i

 0.2114 + 0.5590i

 0.2114 - 0.5590i

Gain Constant

 0.4000

Radius of the poles

 0.7483

 0.7483

 0.5976

 0.5976

Second Order Sections

 0.4000 0.8000 1.2000 1.0000 -0.4229 0.3572

 1.0000 0.5000 0.5000 1.0000 1.2229 0.5599



2- Determine the ROC of the given equation



**Program**

close all; clear all; clc;

num = input('Type in the numerator coefficients =');%1

den = input('Type in the denominator coefficients =');%3 %Conversion from rational to Factored form

[z,p,k]=tf2zp(num,den);

disp('Zeros are at'); disp(z);

disp('Poles are at'); disp(p);

disp('Gain Constant');

disp(k); %Determination of radius of the poles

radius=abs(p);

disp('Radius of the poles');

disp(radius); %Pole Zero Map using numerator and denominator coefficients

zplane(num,den) %Conversion from factored to second ordered factored

sos=zp2sos(z,p,k); disp('Second Order Sections'); disp(sos);

**Ans.**

Zeros are at

 0.8090 + 0.5878i

 0.8090 - 0.5878i

Poles are at

 0.7581 + 0.5508i

 0.7581 - 0.5508i

Gain Constant

 1

Radius of the poles

 0.9370

 0.9370

Second Order Sections

 1.0000 -1.6180 1.0000 1.0000 -1.5161 0.8780

