ACS College of Engineering
Approved by AICTE New Delhi, Affiliated to VTU, Belagavi
(A Unit of RajaRajeswari Group of Institutions)



DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

SUBJECT: BASIC SIGNAL PROCESSING LABORATORY (IPCC)

SUBJECT CODE: 21EC33

PREPARED BY:
Dr.Prajith Prakash Nair
Associate Professor, ECE
ACS College of Engineering

TABLE OF CONTENTS

Exp. No. 1 (a): Program to create and modify a vector (Array)

Exp. No. 1 (b): Program to create and modify a Matrix

Aim: To create a vector and perform various operations on it

Apparatus: Matlab Software, PC

Program:

To create an array with four elements in a single row, separate the elements with either a comma (,) or a space.

```
a = [1 2 3 4]
```

To create a matrix that has multiple rows, separate the rows with semicolons.

```
a = [1 3 5; 2 4 6; 7 8 10]
```

Another way to create a matrix is to use a function, such as ones, zeros, or rand. For example, create a 5-by-1 column vector of zeros.

```
z = zeros (5, 1)
```

OUTPUT :

Exp. No. 2: Programs on basic operations on matrix

Aim: To create a matrix and perform basic operation on it

Apparatus: Matlab Software, PC

Program:

MATLAB allows you to process all of the values in a matrix using a single arithmetic operator or function.

```
a + 10
```

To transpose a matrix, use a single quote (') :

```
a'
```

You can perform standard matrix multiplication, which computes the inner products between rows and columns, using the * operator. For example, confirm that a matrix times its inverse returns the identity matrix:

```
p = a*inv(a)
```

To perform element-wise multiplication rather than matrix multiplication, use the .* operator:

```
p = a.*a
```

The matrix operators for multiplication, division, and power each have a corresponding array operator that operates element-wise. For example, raise each element of a to the third power:

```
a.^3
```

Concatenation **is the process of joining arrays to make larger ones. In fact, you made your first array by concatenating its individual elements. The pair of square brackets [] is the concatenation operator**.

```
A = [a,a]
```

Concatenating arrays next to one another using commas is called *horizontal* concatenation. Each array must have the same number of rows. Similarly, when the arrays have the same number of columns, you can concatenate *vertically* using semicolons.

```
A = [a; a]
```

Complex numbers have both real and imaginary parts, where the imaginary unit is the square root of -1.

```
sqrt(-1)
```

To represent the imaginary part of complex numbers, use either i or j.

```
c = [3+4i, 4+3j; -i, 10j]
```

OUTPUT

Exp. No. 3: Program to solve system of linear equations.

Aim: To create and solve a system of linear equations

Apparatus: Matlab Software, PC

Program:

If you do not have the system of linear equations in the form `AX = B`, use `equationsToMatrix` to convert the equations into this form. Consider the following system.

$2-x+y-z=3$; $x+2y+3z=-10$; $x+y+z=2$

Declare the system of equations.

```
syms x y z
eqn1 = 2*x + y + z == 2;
eqn2 = -x + y - z == 3;
eqn3 = x + 2*y + 3*z == -10;
```

Use equationsToMatrix to convert the equations into the form AX = B. The second input to equationsToMatrix specifies the independent variables in the equations.

```
[A,B] = equationsToMatrix([eqn1, eqn2, eqn3], [x, y, z])
```

Use linsolve to solve AX = B for the vector of unknowns X.

```
X = linsolve(A,B)
```

Exp. No. 4: Program for Gram-Schmidt orthogonalization.

Aim: To create and solve Gram-Schmidt orthogonalization

Apparatus: Matlab Software, PC

Program:

```
A=[1,2,3;2,3,4;4,5,6];
[m,n]=size(A);
Q=zeros(m,n);
R=zeros(n,n);
For j=1:n
v=A(:,j);
for i=1:j-1
R(I,j)=Q(:,i)'*A(:,j);
v=v-R(i,j)*Q(:,i);
end
R(j,j)norm(v);
Q(:,j)=v/R(j,j);
end
```

Output:

Exp. No. 5: Program for Eigen Values and Eigen Vectors.

Aim: To create and solve Eigen values and eigen vector

Apparatus: Matlab Software, PC

Program:

```
A = [-2,-4,2;-2,1,2;4,2,5]

[Eigenvalues_A,Eigenvectors_A] = my_eigen(A)

function [eigenvalues, eigenvectors] = my_eigen(A)

%%%This function takes a n by n matrix A and

%%%returns the eigenvalues and their assocaited eigenvectors.

    [m,n] = size(A);

    assert(isequal(m,n), "A is not a square matrix.") %Check to make sure A is a squa
re matrix.

    syms l

    lI = eye(m)*l;

    char_poly = det(A - lI);

    eigenvalues = solve(char_poly,l);

    for i = 1:m %This is to make sure any eigenvalues that = 0 are actually 0.

        if abs(eigenvalues(i)) < 10^-5

            eigenvalues(i) = 0;

        end

    end

    eigenvalues = transpose(eigenvalues); %This makes it easier to know which

                                          %eigenvalue goes with which eigenvector.

    eigenvectors = zeros(m);  %This line is to make the code run smoother.

    for i = 1:m

       matrix = A - eye(m)*eigenvalues(i);

       a_eigenvector = null(matrix); %An eigenvector is the nullspace of a matrix min
us one of its eigenvalues along the diagonal.

       a_eigenvector = a_eigenvector/min(abs(a_eigenvector)); %This makes the eigenve
ctor look nice and readable.

       eigenvectors(:,i) = a_eigenvector;
```

```
    end
end
```

OUTPUT :

Exp. No. 6: Program to find Singular value decomposition.

Aim: To perform a program for SVD

Apparatus: Matlab Software, PC

Program :

```
U = Varg;
 V = Uarg;
 S = Sarg;
 A = Aarg';

 current_rank = size( U, 2 );
 m = U' * A;
 p = A - U*m;
 P = orth( p );
 P = [ P zeros(size(P,1), size(p,2)-size(P,2)) ];
 Ra = P' * p;
 z = zeros( size(m) );
 K = [ S m ; z' Ra ];
 [tUp,tSp,tVp] = svds( K, current_rank );
 Sp = tSp;
 Up = [ U P ] * tUp;
 Vp = V * tVp( 1:current_rank, : );
 Vp = [ Vp ; tVp( current_rank+1:size(tVp,1), : ) ];
 if ( force_orth )
  [UQ,UR] = qr( Up, 0 );
  [VQ,VR] = qr( Vp, 0 );
  [tUp,tSp,tVp] = svds( UR * Sp * VR', current_rank );
  Up = UQ * tUp;
  Vp = VQ * tVp;
  Sp = tSp;
 end;

 Up1 = Vp;
 Vp1 = Up;
```

Exp. No. 7: Program to generate discrete waveform

Aim: Write a program to generate dicrete waveform

Apparatus: Matlab Software, PC

Program :

```
x = rand(100,1) ;
y0 = rand ;
n = length(x) ;
y = zeros(n,1) ;
y(1) = y0 ;
for i = 2:n
    y(i)=y(i-1)+3*x(i) ;
end
stem(1:n,y)
```

Exp. No. 8: Program to perform basic operation on a signal

Aim: Write a program to perform basic operation on a signal

Apparatus: Matlab Software, PC

Program :

Addition

```
x=[1 2 3 4];
subplot(3,1,1);
stem(x);
title('X');
y=[1 1 1 1];
subplot(3,1,2);
stem(y);
title('Y');
z=x+y;
subplot(3,1,3);
stem(z);
title('Z=X+Y');
```

Subtraction

```
n1=-2:1;
x=[1 2 3 4];
subplot(3,1,1);
stem(n1,x);
title('X') ;
axis([-4 4 -5 5]);
n2=0:3;
y=[1 1 1 1];
subplot(3,1,2);
stem(n2,y);
title('Y');
```

```
axis([-4 4 -5 5]);
n3 =min (min(n1) ,min( n2 ) ) : max ( max ( n1 ) , max ( n2 ) ); % finding the
duration of output signal
s1 =zeros(1,length (n3) );
s2 =s1;
s1 (find ( ( n3>=min( n1 ) ) & ( n3 <=max ( n1 ) )==1 ) )=x;
% signal x with the duration of output signal 'sub'
s2 (find ( ( n3>=min ( n2 ) ) & ( n3 <=max ( n2 ))==1) )=y;
% signal y with the duration of output signal 'sub'
sub=s1 - s2; % subtraction
subplot(3,1,3)
stem(n3,sub)
title('Z=X-Y');
axis([-4 4 -5 5]);
```

Multiplication

```
n1=-2:1;
x=[1 2 3 4];
subplot(3,1,1);
stem(n1,x);
title('X') ;
axis([-4 4 -5 5]);
n2=0:3;
y=[1 1 1 1];
subplot(3,1,2);
stem(n2,y);
title('Y');
axis([-4 4 -5 5]);
n3 =min (min(n1) ,min( n2 ) ) : max ( max ( n1 ) , max ( n2 ) ); % finding the
duration of output signal (out)
s1 =zeros(1,length (n3) );
s2 =s1;
s1 (find ( ( n3>=min( n1 ) ) & ( n3 <=max ( n1 ) )==1 ) )=x;
% signal x with the duration of output signal 'mul'
s2 (find ( ( n3>=min ( n2 ) ) & ( n3 <=max ( n2 ))==1) )=y;
% signal y with the duration of output signal 'mul'
mul=s1 .* s2; % multiplication
subplot(3,1,3)
```

```
stem(n3,mul)
title('Z=X*Y');
axis([-4 4 -5 5]);
```

Exp. No. 9: Program to perform convolution of two signals

Aim: Write a program to perform convolution of 2 signals

Apparatus: Matlab Software, PC

Program

```
p=input('Enter the limit for x');
q=input('Enter the limit for y');
x=input('Enter the elements for x');
y=input('Enter the elements for y');
n1=0:p ;
n2=0:q;
subplot(3,1,1);
stem(n1,x);
title('Signal - x(n)');
subplot(3,1,2);
stem(n2,y);
title('Signal - h(n)');
z=conv(x,y);
t=length(n1)+length(n2)-1;
s=0:t-1;
subplot(3,1,3);
stem(s,z);
title('Output - y(n)');
```

Exp. No. 10: Program to perform property verification

Aim: Write a program to perform associative , distributive and commutative property of signals

Apparatus: Matlab Software, PC

Programs

Exp. No. 11: Program to perform step response from impulse

Aim: Write a program to perform step response from impulse

Apparatus: Matlab Software, PC

Programs :

```
load('PulseResponseReflective100ps.mat');
plot(t,step)
xlabel('Time (Seconds)')
ylabel('Volts')
title('Step Response')
```

Exp. No. 12: Program to perform Z Transforms and Inverse Z transforms

Aim: Write a program to perform Z Transforms and Inverse Z transforms

Apparatus: Matlab Software, PC

Programs :

Z Transform

```
clc;
close all;
clear all;
syms 'z';
disp('If you input a finite duration sequence x(n), we will give you its z-transform');
nf=input('Please input the initial value of n = ');
nl=input('Please input the final value of n = ');
x= input('Please input the sequence x(n)= ');
syms 'm';
syms 'y';
f(y,m)=(y*(z^(-m)));
disp('Z-transform of the input sequence is displayed below');
k=1;
for n=nf:1:nl
    answer(k)=(f((x(k)),n));
   k=k+1;
end
disp(sum(answer));
```

Inverse Z transform :

```
clc;
close all;
clear all;
syms n;
syms k;
syms f(z);
f(z) = input('Please input a function to obtain its inverse z
transform ');
disp(iztrans(f(z)));
```