Signal Analysis
Lab1
Getting Started with MATLAB

Matlab Information

On initiation of MATLAB, **command window** appears in which all the exercises will be done. The **work space** and **command history** windows show the variables defined and past commands respectively. **Command window** can be cleared by **clc** command**. Work space** can be cleared by **clear all** command.

Statement

In the **Command Window**, all statements are typed at command prompt (>>).

Statements preceded by **%** are non executable statements.

Results are stored as variable **ans** in workspace and displayed in the command window.

```
>> 2 (press Enter)
ans=
      2
```

After the statement is complete, pressing **Enter** key causes the command to be executed.

```
>> log10(10)  (press Enter)
ans=
      1
```

Multiple statements separated by commas, can by typed at one prompt. The statements are executed from left to right.

```
>>  x=3, y=x+2
x=
   3
y=
   5
```

If statement is terminated with a semicolon (;) then the statement is executed but the result is not printed to the command window. The value entered or computed can be displayed by entering the variable at the prompt.

```
>>x=2;
>>
>>x
x=
    2
```

MATLAB has many built in functions. Use the command **help** for searching MATLAB function. Use the command **help function** to get help using the function.

Previously entered statements in the command Window are stored in a buffer. They can be recovered by using the Up Arrow key.

Data can be stored as variables. Variable names are case sensitive. Variable names cannot start with a number and cannot contain punctuation or special characters.

All entered and computed values remain in the Workspace and can be used or printed to the Command Window. Any variable can be overwritten later. To determine which variables are stored in the Workspace and their sizes, we can use the command **whos**.

M-Files
Files with the filename extension **.m** are executable files. Functions that are already defined and used are contained in function m-files in MATLAB toolboxes.

Script M-Files
Open the **Editor/Debugger** using **File-New-M-file.** This can also be accomplished using the **new file** icon on the toolbar. Begin with the commands **clear all, close all** and **clc**. You should begin all script M-files with these commands to clear out all previously used variable, close open figure windows and refresh the command window.
MATLAB function names are reserved file names. If you name an M-file with the same file name as a function, MATLAB will execute your file instead of the function when it is called. **Do not name an M-file with a reserved file name.**

To execute a script M-file, type the script name at the command prompt and press enter. If your script contains errors, MATLAB will display an error message below the prompt, along with the line number that generated the error. If your script does not contain errors, you will see another prompt when the script has executed.

Storage and retrieval commands
Command **save *filename variables*** saves the variables from the workspace in the file filename.mat in the MATLAB/work directory. The command **load *filename*** loads file ***filename*** from the MATLAB/work directory.

Numeric Format

Complex number
It consists of real part and imaginary part. We can use **i** or **j** as an indicator for imaginary **i**. MATLAB will recognize **i** or **j** as imaginary indicators. The real part, imaginary part, amplitude, and angle in radians of complex number are given by the functions **real, imag, abs,** and **angle,** respectively.

```
>>x=2 + 3j, b=real(x), c=imag(x), d=abs(x), e=angle(x)
>> x =
 2.0000 +3.0000i
b =
     2
```

```
c =
    3
d=
   3.6056
e=
   0.9828
```
We can also enter a complex number as x=2+j*3, where * indicates multiplication.

Matrices and vectors

All numeric values are stored in matrices. The single values considered in the above sections are stored in (1x1) matrices. Single row or single column matrices are called vectors. We enter a *(n x m)* matrix (*n* rows, *m* columns) by entering the individual matrix term values row by row within square brackets. We can enter more than one row on a single statement line if we use semicolons between rows.

```
>> a=[3 4; 2 1]
>>a =
       3 4
       2 1
>>b= [1.5 -2.4 3.5 0.7; -6.2 3.1 -5.5 4.1; 1.1 2.2 -0.1 0]
>> b=
      1.5000   - 2.4000    3.5000    0.7000
     - 6.2000    3.1000  - 5.5000   4.1000
      1.1000     2.2000  - 0.1000   0
```

Values stored in a matrix can be referenced by an index. All MATLAB indices begin with 1, and cannot be 0 or negative. You can specify values in a matrix using a single index, or by specifying its row and column indices.

```
>>b(1)
>> ans=
          1.5000
>>e=b(2, 3), f=b([2 3], [1 3]), g=b(2, [3 4])
>> e=
       - 5.5000
f=
   - 6.2000 -5.5000
     1.1000 -0.1000
g=
   -5.5000 4.1000
```

We can also create a matrix by concatenating (joining) several vectors. The vectors used must have the correct number of rows and columns to make the resulting matrix proper.

```
>>h=[1 2 3], k=[4; 7], m=[5 6; 8 9]
h=
      1 2 3
k=
    4
    7
m=
   5 6
   8 9
>>n=[h; k m]
n=
  1 2 3
  4 5 6
  7 8 9
```

Vectors

Single-row or single-column matrices are called vectors. They are entered like matrices.

```
>>a=[3 5 9], b=[3; 5; 9]
a=
   3 5 9
b=
    3
    5
    9
```

Colon (:) notation can be used to enter a set of equally spaced, sequential numbers.

Variable = start : step : end;

```
>>c=2:5, d=3:2:9
c=
   2 3 4 5
d=
   3 5 7 9
```

This example evaluates the function y = sqrt (x) at samples from 0.5 to 2.0 in steps of 0.25.

```
>> x=0.5:0.25:2.0;
>>y=sqrt(x);
>>x
x=
    0.5000 0.7500 1.0000 1.2500 1.5000 1.7500 2.0000
>>y
y=
   0.7071 0.8660 1.0000 1.1180 1.2247 1.3229 1.4142
```

Vectors also use indices to reference values. Multiple indices will return multiple values.

```
>>f=[10 5 4 7 9 0];
```

```
>>g=[2 5 6]; h=f(g);
>> f
f=
  10 5 4 7 9 0
>>h
h=
   5 9 0
```

Arrays
Matrices or vectors can also be interpreted as two-dimensional or one-dimensional arrays, respectively.

Special Number Notation
Two special numbers are provided in MATLAB. They are pi and infinity and are given the notation **pi** and **Inf**, respectively. In addition, operations such as **0/0** or **sin(inf)** produce undefined results that is given the notation **NaN**, which stands for Not a Number.

Character Strings
In addition to numbers, MATLAB can also store and use text. Text is stored in character *strings*. We designate a string with single quotes (‘ ’).

```
>>'Signal Analysis'
ans=
Signal Analysis
```

The character strings are stored in arrays with one character corresponding to one array element. Therefore, we can select a portion of a character string to use or print.

```
>>v='SIGNAL analysis'
v=
SIGNAL analysis
>>v1=v([1:3])
v1=
SIG
```

Arithmetic Operations
MATLAB defines all arithmetic operations in matrix terms. Use the command **help arith** and **help slash** to list the arithmetic operators. Matrix sizes must be appropriate (conformable) for all arithmetic matrix operations. Examples are provided below.

```
>>a=[1 2; 3 4]; b=[3 1; 7 8]; c=[2 4];
>> d=a+b, e=c*a, f=a^2, g=c'
d=
      4 3
     10 12
```

```
e=
   14 20
f=
   7 10
  15 22

g=
   2
   4

>>h=a\b, k=b/a
h=
   1.0000 6.0000
   1.0000 -2.5000
k =
   - 4.5000 2.5000
   - 2.0000 3.0000
```

To use an operator on an element by element basis, rather than in a matrix math sense, we use the modifier **'.'** It is implicit that scalar multiplication is performed on each element.

```
>>m=a.*b, n=b./a, o=b.^a
>> m =
         3   2
         21 32
n =
    3.0000 0.5000
    2.3333 2.0000

o =
    3     1
    343 4096
```

Function M-Files

Function m-files are like script m-files except that variable values may be passed into or out of function m-files. Also, variables defined and manipulated only inside the file do not appear in the Workspace. The first line of a function m-file starts with the word function and defines the function name and input and output variables.

For example

function [z,w] = siganal(x,y)

is the first line of the function m-file **siganal.m**. The input variables of this function are x and y and the output variables are z and w. Each input and output is an array or matrix. If the array is (1x1), then the variable has a single real or complex value. A function m-file name must match the function name for it to execute. Here is an example of a function m-file for siganal.m,

```
function [z,w] = siganal(x,y)
% siganal is an example of a function. This function
% computes the sum and difference of x and y
z = x+y;
w = x-y;
```

An example of calling this function is shown below.

```
>> x=[3 4 5];y=[1 2 3];
>> [z,w]=siganal(x,y)
z =
    4 6 8
w =
    2 2 2
```

Many functions, such as **z = mean(x)**, are built-in MATLAB functions. However, others are contained in function m-files in MATLAB toolboxes. A function m-file (or a script m-file) can be executed as long as it is in the MATLAB path.

Logical Operations

The logical operations AND, OR, and NOT, are specified by &, |, and ~ , (ampersand, pipe, and tilde) respectively. These can be used in conjunction with the relational operations (<,<=,>,>= ,== ,~=) to construct arrays of zeros and ones (0 - 1 arrays). The ones correspond to elements for which the logic operation is true.

```
>>a=[1 1 2; 6 6 5], b=a>1&a<6
a=
   1 1 2
   6 6 5
b=
   0 0 1
    0 0 1
```

Mathematical Functions

MATLAB contains a set of built-in mathematical functions. All of these functions are applied to arrays on an element-by-element basis. A partial list is given below.

sqrt	-	square root
real	-	complex number real part
imag	-	complex number imaginary part
abs	-	complex number magnitude or absolute value of a real number
angle	-	complex number angle
exp	-	exponential base e
log	-	logarithm base e
log10	-	logarithm base 10

The trigonometric functions all apply to angles expressed in radians.

sin	-	sine
cos	-	cosine
tan	-	tangent

asin - arcsine
acos - arccosine
atan - arctangent
round - round to nearest integer
floor - round toward -∞
ceil - round toward ∞

Mathematical Expressions

We can combine arithmetic operations, 0 -1 arrays generated by logical operations, and mathematical functions into mathematical expressions. Often, these expressions take the form of equations, although they may also be used in flow control statements. The arithmetic operations follow the usual precedence rules. Many mathematical expressions require parentheses to construct the desired sequence of operations within the precedence rules.

```
>>t=0.1;
>> x=3^t*exp(t)-log10(t)
x=
   2.2335

>>y=[3^t*exp(t)-log10(t)+j*(3^t*exp(t)-log10(t))  3^t*exp(t)-log10(t)-j*(3^t*exp(t)-log10(t))]
y=
    2.2335+2.2335i    2.2335-2.2335i
```

Flow Control

MATLAB has flow control statements that we can use to repetitively or selectively execute other statements. The flow control statement affects all statements between itself and an associated end statement.

For Statement

The for statement permits us to execute the same set of statements repetitively for a designated number of times. To evaluate the summation $\boldsymbol{x[n] = \sum_{k=1}^{4} \frac{1}{4} e^{j\frac{2\pi}{4}kn}, 1 \leq n \leq 4}$

```
>>n=1:4;x=zeros(size(n));
>> for k=1:4,
x=x+0.25*exp(j*2*pi/4*k*n);
end
>>x
x=       0 - 0.0000i        0 - 0.0000i   0.0000 - 0.0000i   1.0000 - 0.0000i
```

For statements can be nested

```
for k=1:2,
for m=1:2,
y(k,m)=k+m;
end
end
```

```
>> y
y =
    2    3
    3    4
```

While Statement

While statement is like **for** statement except that execution stops when a logic expression is satisfied. The statements

```
>> n=2;
>>while 2*n<100;
n=2*n;
end
>> n
n=
  64
```

If Statement

The if statement permits us to execute statements selectively depending on the outcome of a logic expression.

```
>>for k=1:4;
if k==1; x(k)=3*k;
else if k==2|k==4; x(k)=k/2;
else; x(k)=2*k;
end;
end;
end;
>> x
x =
    3 1 6 2
```

Numeric Functions

find(A) Returns a one-dimensional row-array containing the indices of non-zero elements of a one-dimensional array. It can be used with 0 - 1 arrays to find indices of elements that have other values.

```
>>a=[0 1 2 4 0 4]; b=find(a)
>>b =
b=
    2 3 4 6
>>c=find(a>2);
>>c
c=
   4  6
```

zeros(m,n) returns an (m x n) array of zeros.

max(A) returns the value of the largest element in a one-dimensional array.

max(max(A)) returns the value of the largest element in a two-dimensional array.

min(A) returns the value of the smallest element in a one-dimensional array.

mean(A) returns the average value of all elements in a one-dimensional array and a one-dimensional row-array containing the mean values of the elements in the columns of a two-dimensional array.

meshgrid(A,1:n) returns an array having n rows where each row is the one-dimensional array A.

```
>>d=[1 2 3 4 5]; dm=meshgrid(d,1:3)
>> dm
dm=
     1 2 3 4 5
     1 2 3 4 5
     1 2 3 4 5
```

sum(A) returns the sum of the elements of A if A is a one-dimensional array. Returns a one-dimensional row-array that contains the sums of the columns of A if A is a two-dimensional array.

Plotting Functions and Commands

plot(x,y) plots the variable y versus the variable x in the current Figure Window by connecting data values with straight lines. Two or more functions of the same independent variable can be plotted on the same set of axes. To do so, remain in the same Figure Window and use the statements hold on and hold off after the first and last plot statements, respectively.

hold on is used after a plot command to draw multiple lines on one axis.

hold off is used to releases the hold on command.

xlabel('text') labels the x-axis of a plot with the text specified by 'text'.

ylabel('text') labels the y-axis of a plot with the text specified by 'text'.

title('text') places the title specified by 'text' above a plot.

text(x,y,'text') adds the text specified by 'text' to a plot at the location (x, y), where x and y are the horizontal and vertical axis coordinates, respectively.

legend creates a legend for multiple plots on one axis.

figure opens a new Figure Window.

subplot(m,n,ax) splits the Figure Window into a matrix of mxn axes.
For example **subplot(2,1,2)** specifies two rows and one column of axes with second axis being used.

axis([ranges]) sets scaling for the x-, y- and z-axes on the current plot according to
to the vector ranges = [xmin xmax ymin ymax zmin zmax]. If only a
2-D is being used, ranges = [xmin xmax ymin ymax].