

Laboratory 1 – Introduction to MATLAB for Signals and Systems

INTRODUCTION to MATLAB

MATLAB is a powerful computing environment for numeric computation and visualization. MATLAB is designed for ease of use and behaves as a high-level programming language that is tailored for signal processing, communication, complicated calculations and control tasks. It is used by professionals in industry and academia worldwide in research, development, and design.

MATLAB is available on many other platforms. MATLAB runs under Microsoft Windows, Linux X Windows, or Mac OS. This introduction is not intended to present you with everything you need to know about MATLAB; it is merely to bring you to a point where you can do the following labs in the course.

All versions of MATLAB are compatible in file storage format and M-file format, so data stored on one system can be transferred to another without loss. Each MATLAB session has at least two windows (see Fig. 1): a text window, where commands are typed and data is displayed, and a workspace, where the name of the variable and its value are displayed.

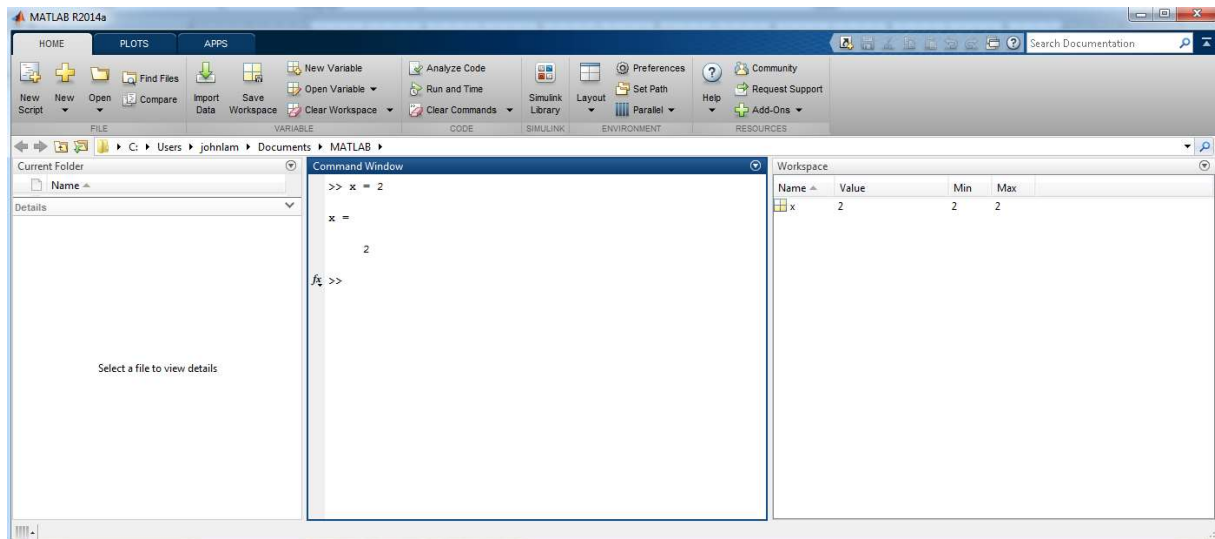


Fig. 1 Command window in MATLAB

Basic operations in MATLAB

Matrices and vectors make up the heart of MATLAB computations. In this section, matrix and vector manipulations will be introduced. A vector is a one-dimensional list of values, an $m \times 1$ or $1 \times m$ matrix. Vectors hold single signals or lists of data. They can be assigned a name and treated as any other variable in MATLAB; however, operations performed on vectors are done element by element. As an example of this, consider the function $y = 3x+2$. If we want to plot y as a function of x , we first create an x vector containing data points in the range of interest. Suppose the range is from 0 to 5, using every integer point. There are several ways to create this data set in MATLAB. The first way is to type in every point:

```
>> x = 0:1:5
```

```
x =
```

```
    0    1    2    3    4    5
```

This generates a row vector \mathbf{x} , i.e. a 1×6 matrix containing six elements, the integers 0 through 5. Note that the semicolon keeps MATLAB from echoing the results of your command back to the screen. An easier way to generate this same vector is to use a range-generating statement:

```
>> x = 0:5
```

```
x =
```

```
    0    1    2    3    4    5
```

The colon operator acts like the word “to”, in effect generating the function “0 to 5”. A step size of 1 is the default value. A different step size, positive, negative, real or integer, can be specified by placing the step value between the beginning and end of the range, as in \mathbf{z} below:

```
>> z = 0:0.01:5;
```

generates 501 data points that are 0.01 apart, starting from 0 and ending at 5.

Plotting signals in MATLAB

Plotting signals is a basic skill in MATLAB that will be used frequently in analyzing signals and systems. Plots in MATLAB are generated using the `plot` function. The function `plot(x, y)` generates a 2-D plot where the values of the vector \mathbf{x} indicate points along the horizontal axis. The values in the vector \mathbf{y} will be plotted on the vertical axis. Vectors \mathbf{x} and \mathbf{y} must have the same number of elements.

To plot a signal in MATLAB, a signal is basically represented by a vector or called the **signal vector**. This vector represents values of the signal at specified points in time. These time points are stored in a separate vector, called the **time vector**. MATLAB plots the signal by plotting the points in the signal vector vs. the points in the time vector on an x - y grid and then connecting the points. Properly labeling the signal(s) on a graph is essential for proper presentation.

Example 1: Consider the CT signal $x(t) = \sin(t)/2$. Plot $x(t)$ for the interval: $0 \leq t \leq 5$ in MATLAB, label the x -axis as “time” and the y -axis with the original equation of the signal. The MATLAB instructions are given below:

```
clear
t = 0:0.01:10;           ← define the time vector, with an increment of 0.01
x = sin(t)/2;
plot(t,x);              ← need to define the independent variable first, in this case, “t”
xlabel('time');         ← text displayed in the x or y or z label must be in “
ylabel('x(t) = sin(t)/2');
grid;                   ← display the grid lines on the plot
```

Fig. 2 displays the result in MATLAB.

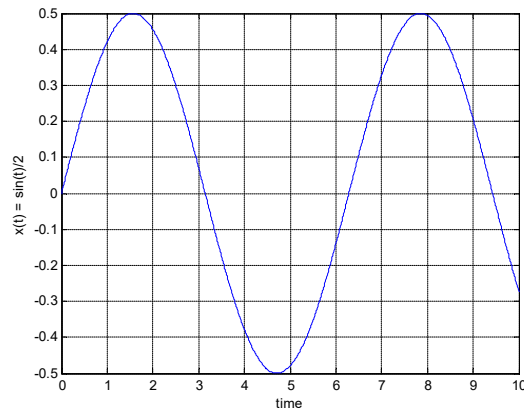


Fig. 2

To perform operations on the signal in MATLAB it is necessary to perform operations on the signal vector. Often these operations take the form of arithmetic operations on elements of the signal vector. For example, to square a signal, it is necessary to square each element of the signal vector. Since the ordinary arithmetic operations in MATLAB are matrix operations, the usual arithmetic symbols must be modified. To perform arithmetic operations on each element of a matrix, the operator is preceded by a period or what MATLAB calls, a “dot operator”. For example, to square a vector \mathbf{x} , the MATLAB command is: “ $\mathbf{x}.\mathbf{x}$ ”.

Example 2: Consider the CT signal $x(t) = 2t\cos(t)$. Plot $x(t)$ for the interval: $0 \leq t \leq 20$ in MATLAB. The MATLAB instructions are given below:

```
clear
t = 0:0.01:20;
x = 2*t.*cos(t);
plot(t,x);
xlabel('time');
ylabel('x(t) = 2tcos(t)');
grid;
```

← note how the “dot operator” is inserted between “ t ” and “ $*$ ”

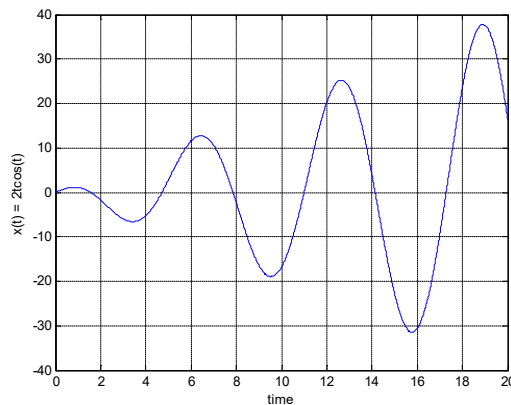


Fig. 3

Evaluating complex numbers and functions in MATLAB

Complex analysis is commonly used in numerical calculations and engineering problems, such as

- Complex roots in polynomials
- Impedance and apparent power in electrical circuits

- Complex poles in system stability analysis

In MATLAB, **i** and **j** are built-in constants that return $\sqrt{-1}$. For example, a complex number $z = 2 + 3i$ can be defined in MATLAB as:

```
>> z = 2+3i
z =
    2.0000 + 3.0000i
```

Alternatively, the function **complex(x1,x2)** can also be used to define a complex number:

```
>> z = complex(2,3)
z =
    2.0000 + 3.0000i
```

The complex number shown in the previous example is expressed in *Cartesian or rectangular form*. As illustrated in the lecture notes, complex numbers can also be expressed in *polar form* (i.e. expressed with magnitude and phase of the complex number). In MATLAB, a complex number expressed in *rectangular form* can be transformed to polar form using the function **cart2pol()**. For example,

```
>> [phase_angle,magnitude] = cart2pol(2,3)
phase_angle =
    0.9828
magnitude =
    3.6056
```

Similarly, the function **pol2cart()** can be used to transform a complex number from polar form to its rectangular form. Other useful functions in MATLAB for complex numbers analysis include:

- **abs()** that allows you to obtain the magnitude of a complex number in rectangular form,
- **angle()** that allows you to obtain the phase of a complex number in rectangular form.

A CT exponential complex function can be defined in MATLAB using the **exp()** function. Note that the CT exponential complex function is expressed in polar form. For example, $e^{i\pi/3}$ is defined in MATLAB as:

```
>> exp(pi/3i)
ans =
    0.5000 - 0.8660i
```

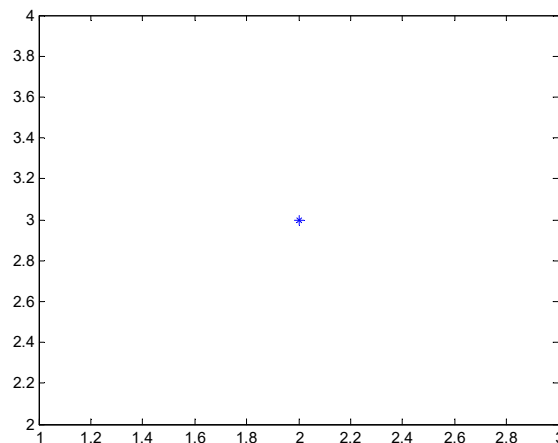
Note that MATLAB has evaluated the expression and echoed the result to the screen, expressed in rectangular form as the variable **ans**.

Plotting complex numbers and functions in MATLAB

The `plot` function introduced previously can be used to plotting complex numbers and functions in MATLAB. For example, to plot the complex number $z = 2 + 3i$

```
>> z = complex(2,3);
>> plot(real(z), imag(z), '*')
```

The `real()` and `imag()` return the real and imaginary part of the complex number. The ‘*’ parameter to the plot function tells MATLAB to generate a * for each data point instead of a ‘connected-dot’ display. Since we only plotted one data point, this is very useful.



To plot a complex signal, it is often useful to plot both the real and imaginary parts as a function of time. We will leave this as an exercise for you.

User-defined functions

So far, we have already seen the use of several different built-in functions in MATLAB, such as `sin`, `complex`, `exp`. A “user-defined function” is a function that the programmer defines, customizes so that it can be called anytime.

A user-defined function in MATLAB is a special kind of M-file. The first line in the file defines the function, both giving it a name and indicating what values are to be passed as arguments to the function and those that are to be generated by the function, much like a subroutine in other programming languages. Once you have created a function you can use it just as would use a MATLAB-supplied function.

The general form of a “function” file with n number of inputs and returns a single output looks like this:

```
function output_name = function_name(input1, input 2, input 3, ..., inputn)
% The comment section briefly describes what the function does

MATLAB statements go here; must include putting a value in the output
argument

end
```

A useful feature of the “function” files is that the lines that follow the function definition and begin with a comment symbol (%) are printed when help is requested for your function. **Note: that the name of the file must be the same as the name of the function!**

function_name.m

The following example illustrates the use of a user-defined function called “**check**”. A brief description about this function is provided in the comment section.

```
function [output] = check(x1,x2)
% The function CHECK evaluates if the input value x2 is
% greater than or equal to the other input value x1. If
% it is, then the output is given by x2*cos(x1), otherwise
% the output is given by x1*sin(x2)

if (x2 >= x1)
    output = x2*cos(x1);
else
    output = x1*sin(x2);
end
```

You should also become familiar with other conditional statements such as: for-loop and while-loop. In the Command Window, type the following to see the details.

```
>> help for          or          >> help while
```

Relational Expressions

The example given in the previous section illustrated the use of a relational operator (i.e >=).

Expressions that are conceptually either true or false are called relational expressions; they are also called Boolean expressions.

A list of relational operators in MATLAB is given below:

Operator	Meaning
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
==	equality
~=	inequality

Relational operators can be combined with a time array to plot piecewise functions. Try these on your own through the laboratory exercises.


Laboratory 1 exercises (Deadline: January 31 at 12pm)

Submitting your M-files and published report using the EECS submit service:

You can submit multiple files stored on your computer using the web-based submit service:
<https://webapp.eecs.yorku.ca/submit/index.php> .

You will need to provide your EECS login credentials to submit your file(s). Once you have logged in, select “2016 - 17” for Academic Year, then choose the Course (2602) and the Assignment (lab1_M). Browse for your file or files, and press the Submit Files button. You may submit files as many times as you like. The most recent submission overwrites all previous submissions.

Files to be submitted:

1. Main M-file
 2. Published report of the main M-file in pdf (in your M-file, select the “PUBLISH” Tab and  click)
 3. M-file: **squarewave.m** for Q4
-

Q1:

Consider the CT signal: $x_1(t) = 0.1\sin(t) + 0.2\cos(\omega t + \theta)$

- a) Plot $x_1(t)$ with $\theta = 60^\circ$ for $0 \leq t \leq 10$ and $f = 2\text{Hz}$. Label the x - and y -axis properly.
- b) Investigate the waveforms by varying θ in $x_1(t)$ from 0° to 180° with an increment of 45° . Plot these waveforms on the same graph and provide a legend for these waveforms. Label the axes. You should investigate how to make multiple plots on the same graph.
- c) Is $x_1(t)$ a periodic signal? Explain.

Q2:

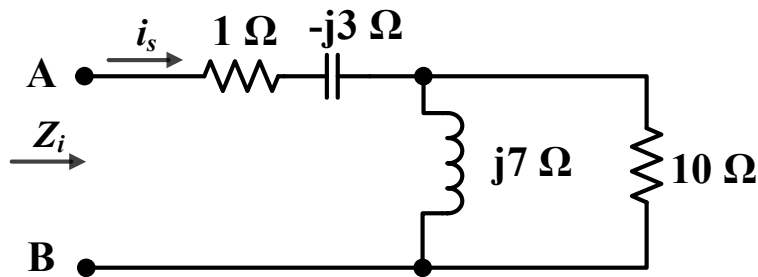
Consider the CT signal: $x_2(t) = e^{-0.1\pi t} \cos(2\pi t)$

- a) The exponential component in $x_2(t)$ is called the positive envelope of $x_2(t)$, call this function $e(t)$. Plot both $e(t)$ and $x_2(t)$ on the same graph for $0 \leq t \leq 10$ with $e(t)$ displayed in red dotted line. Label the x axis and y axis properly. Provide a legend for these waveforms.

b) A new signal $x_{2b}(t)$ is now obtained by multiplying $x_2(t)$ with the time-shifted unit-step function $u(t-2)$ as discussed in class. Plot $x_{2b}(t)$ for $0 \leq t \leq 10$. (*Hint*: you may want to use the **Heaviside** function in MATLAB to implement the unit-step function)

Q3:

a) Calculate the equivalent impedance Z_i in the following circuit, express your final answer in polar form.



b) Suppose a *sinusoidal* voltage signal (v_s) with an amplitude of 50V and a period of 0.2s is connected to the terminal A-B of the above circuit. Find i_s and express your final answer in polar form? Plot both v_s and i_s on the same graph for 2 periods. Provide a legend for these waveforms and label the axes.

Q4:

A symmetrical square wave signal with amplitude V and frequency f can be described by the following series:

$$x_{sq}(t) = \frac{4V}{\pi} \sum_{n=1,3,5}^{\infty} \frac{1}{n} \sin(n2\pi ft) \quad (1)$$

Write a user-defined function “**squarewave**” in MATLAB that displays *both the square waveform and its fundamental component*. The function will also *output the maximum instantaneous power of the fundamental component of $x_{sq}(t)$* . The function will allow users to input the frequency of the square wave signal (f), the amplitude V , the number of terms to be used in the series in (1), and the time array. Your output display must include a legend and a title.

Test your function by calling **squarewave.m** in your main M-file, and using the following parameters: $V = 10$, $f = 2$ Hz and 150 terms of the series.

Q5:

a) Sketch the following CT signal with respect to the time variable from -5 to 5 with a suitable time step. Sketch the real and imaginary components separately (i.e. use `subplot()` function in MATLAB). Provide a title for each plot. Label the axes. (*Hint: look for `sgn(t)` in MATLAB*)

$$x(t) = -3 \operatorname{sgn}(t) \cdot e^{j2\pi + 0.25t}$$

b) Suppose a new signal $x_{new}(t)$ is now created by multiplying $x(t)$ with a rectangular pulse signal as shown below:

$$x_{new}(t) = x(t) \cdot \operatorname{rect}\left(\frac{t}{4}\right)$$

Sketch the real and imaginary components of $x_{new}(t)$ separately with respect to the time variable from -5 to 5 with a suitable time step. Provide a title for each plot. Label the axes. **Any if-loops, for-loops and while-loops are NOT allowed for question 5.**

Q6:

If $x(t)$ is given by the following piecewise signal, plot $x(t)$ and $2x(t+1)$ on separate plots but on the same graph.

$$x(t) = \begin{cases} 0.25t & -2 \leq t < 0 \\ 1.5 & 0 \leq t < 0.5 \\ -t + 2 & 0.5 \leq t \leq 2 \end{cases}$$