MATLAB Tutorial

If you are new to MATLAB and Simulink You may wish to work through the introductory examples and watch the videos contained in this tutorial.

An Introduction to MATLAB

MATLAB is a high-level language for technical computing which is often used by engineers to help them design systems or analyze a system's behaviour. We present the following material in a tutorial style which can be used as a self-teaching exercise, or as a reminder for those who may have used MATLAB before and forgotten some of the notation. The following represents a beginner's route into the use of the MATLAB package for Signals and Systems and the Symbolic and Signal Processing toolboxes.

This document is available as a MATLAB Live Script (tutorial.mlx) in which all the MATLAB code is executable.

Getting Started with Matlab

Before you begin, you may wish to watch the video Getting Started with MATLAB (https://uk.mathworks.com/videos/getting-started-with-matlab-1564521672719.html) (Links to an external site.) by Erin Byrne of the MathWorks.

In this tutorial introduction to MATLAB we shall cover the following:

Table of Contents

If you are already comfortable working with MATLAB you may wish to skip the introductory materials and jump to Polynomials.

Basic Operations

Arithmetic Operations

We can assign a numerical value (data) to a variable using the equal (=) sign. For example, type: a = 2 and press return. MATLAB returns:

```
» a = 2

a =

     2
```

or b = -5.2

```
» b = -5.2

b =

    -5.2
```

Exercise 1: Basic Operators

Execute the commans below. The basic arithmetic operations use the operators shown. Use values of a = 4 and b = -2.1.

In [ ]:
```
a = 4
b = -2.1
```

Plus (+)

In [ ]:
```
a+b
```

Minus (-)

In [ ]:
```
a-b
```

Multiply (*)

In [ ]:
```
a*b
```

Power (^)

In [ ]:
```
a^b
```

Divide

In [ ]:
```
a/b
```

Vectors

To define a row vector we type each element of the vector (separated by a comma) between square brackets and set it equal to a variable. For example, to create the row vector x, enter into MATLAB:

```
» x = [1,2,3,4]

x =
    1     2     3     4
```

To enter a column vector, separate the elements by a semicolon; for example:

```
» y = [1;2,3;4]

y =
    1
    2
    3
    4
```

We can determine the size of the vectors x and y by using the size command.

```
» size(x)

ans =  1  4

» size(y)

ans =  4  1
```

If we want to create a vector with elements between 0 and 5 evenly spaced in increments of 0.5, we can use:

```
» t = 0:0.5:5

t =

         0    0.5000    1.0000    1.5000    2.0000    2.5000    3.0000
3.5000    4.0000    4.5000    5.0000
```

Exercise 2: Vectors

A. Enter the first three prime numbers as a row vector.

In [ ]:
```
% put your code here
```

B. Repeat (A), but use a column vector.

In [ ]:
```
% put your code here
```

C. The following table shows some measurements of voltage across a resistor as a function of timeTime (sec)

Time (s)	**0.0**	**1.0**	**2.0**	**3.0**	**4.0**	**5.0**	**6.0**	**7.0**	**8.0**	**9.0**
Voltage (v)	1.1	1.2	1.3	1.4	1.5	1.4	1.3	1.2	1.1	1.0

Enter the time as a column vector called `time`

In [ ]:
```
% put your code here
```

Enter the voltage as a column vector called `voltage`

In [ ]:
```
% put your code here
```

Execute the command `plot(time, voltage)`

In [ ]:
```
% put your code here
```

On the resulting graph:

- Which variable is on the independent axis?
- Which variable is on the dependent axis?

Vector manipulation

Manipulating vectors is almost as easy as creating them. Try the following operations:

In [ ]:
```
a = [1, 2, 3, 4]
```

In [ ]:
```
x = a+2
```

In [ ]:
```
x = a-2
```

In [ ]:
```
x = a*2
```

In [ ]:
```
x = a/2
```

Let `b = [1, 2, 3]` . What happens if we try to add two variables (vectors: a+b) of different dimensions? What is the error message?

In [ ]:
```
% put your code here
```

(We note that, in the example `x = a + 2` , variable `a` is size 1 x 4 and `2` is size 1 x 1. Technically they could not be added together, as the dimensions are not compatible. However MATLAB assumes, when you add/subtract/multiply/divide a variable by a number, that all elements of the vector should be operated on. This is not true with two vectors, as we see now.)

For

In [ ]:
```
x = [1, 2, 3, 4]
```

and

y = [5; 6; 7; 8]

would the following operations be acceptable?

In [ ]:
```
x+y
```

In [ ]:
```
x*y
```

In [ ]:
```
y*x
```

In this case, we note that we cannot add `x + y` since they do not have appropriate dimensions. However, if we use transpose `x'` (where `'` indicates the transpose), we can then evaluate:

In [ ]:
```
x + y'
```

The `size` command could be used to verify the dimensions of the vectors.

Exercise 3: Vector Manipulation

Try the following vector operations and note the results. Ensure the dimensions of x and y are appropriate for the operations attempted. Note that some of the operators are different from scalar operators.

Plus (+)

Must be same dimension

In [ ]:
```
x+y
```

Minus (-)

In [ ]:
```
x-y
```

Multiply (*)

Must be appropriate dimensions

In [ ]:
```
x*y
```

Multiply element by element (.*)

Must be same dimension

In [ ]:
```
x.*y
```

Divide element by element (./)

In [ ]:
```
x./y
```

Divide element by element (.\ )

In [ ]:
```
x.\y
```

Explain in words the three operations: `.*` , `./` and `.\` .

Think of an example where we might need this type of operation.

Remark: Note if we put a semi-colon (;) at the end of a MATLAB command, MATLAB does not return the values to the window. Try:

In [ ]:
```
a=[1, 2, 3];
```

In [ ]:
```
b=[-2, -3, 4];
```

In [ ]:
```
c=a+b
```

In [ ]:
```
c=a-b
```

Polynomials

We can enter the coefficients of a polynomial as a row vector. The coefficients should be entered in descending order of the powers of x. For example

$$p(x) = x^2 + 10s + 3$$

can be represented by

```
» p = [1, 10, 3];
```

How would we enter the following polynomial, q?

$$q(t) = t^3 + 4t^2 + 3t + 1?$$

In [ ]:
```
% put your code here
```

MATLAB can interpret a vector of length n+l as the coefficients of an n^{th} order polynomial. Thus, if the polynomial is missing any coefficients, we must enter zeros in the appropriate place in the vector. For example

$$y(s) = s^4 + 1$$

would be represented in MATLAB as:

```
matkab
» y = [6, 0, 0, 0, 1];
```

Finding roots

Use the following command:

```
» roots([6, 0, 0, 0, 1])
```

or

```
» roots(y)
```

How would we enter the following polynomial, m:

$$m(s) = 4s^4 + s^3 + 2s$$

In [ ]:
```
% put your code here
```

Determine the roots of m.

Multiplying/Dividing polynomials

The product of two polynomials is found by taking the convolution of their coefficients. The function `conv` will do this for us.

Polynomial: $x(s) = s + 2$

In [ ]:
```
x = [1, 2]
```

Polynomial: $y(s) = s^2 + 4s + 8$

In [ ]:
```
y = [1, 4, 8]
```

Polynomial: $z(s) = x(s)y(s)$

In [ ]:
```
z = conv(x,y)
```

Dividing two polynomials is just as easy. The `deconv` function will return the result, Q, as well as the remainder, R. Note that if there is more than one output for a function, they should be put inside square brackets and be separated by commas. Try to divide $z(s)$ by $y(s)$:

In [ ]:
```
[Q, R] = deconv(z,y)
```

How do we check the result?

Exercise 4: Polynomials

We wish to plot a second-order polynomial

$$y(t) = 4t^2 + 2t - 3$$

and check the roots of the equation.

We can do this in two ways.

1. We can enter the polynomial coefficients as a vector and use the `roots` command.

In [ ]:
```
% put your code here
```

1. We can plot the function y against time and note the points where the function crosses the time axis (that is points where $y = 0$).

   a. Create a vector `t` , which has values evenly spaced from -5 to 5, in steps of 0.5.

   b. Calculate $y(t) = 4t^2 + 2t - 3$ for all values of the time vector, `t` . Since `t` is a vector, think carefully how you will calculate $4t^2 + 2t - 3$.

   c. Request a plot using `plot(t,y)` . Find the roots on the graph. Are the roots on the graph similar to the answer in 1 above?

In [ ]:
```
% put your code here
```

Matrices

Entering matrices into MATLAB is the same as entering a vector, except each row of elements is separated by a semicolon, try:

In [ ]:
```
B = [1, 2, 3, 4; 5, 6, 7, 8; 9, 10, 11, 12]
```

Matrices in MATLAB can be manipulated in many ways. For example, we can find the transpose of a matrix using the apostrophe key ('). Try

In [ ]:
```
C = B'
```

Remember that the order of multiplication is important when dealing with matrices. Would the following operations be allowed?

```
D = B + C
D = B * C'
```

In [ ]:
```
% put your code here
```

Now try these:

In [ ]:
```
D = C * B
```

In [ ]:
```
D = B * C
```

In [ ]:
```
E = D ^ 3 % power of a matrix
```

In [ ]:
```
X = inv(E) % inverse of a matrix
```

Exercise 5: Matrices

Find the solution of the following set of linear equations:

$$\begin{aligned} 2x_1 + 5x_2 + 3x_3 &= 6 \\ 3x_1 - 2x_2 + 4x_3 &= -2 \\ x_1 + 6x_2 - 4x_3 &= 3 \end{aligned}$$

Hint: We write this in the matrix form, that is, $A\ X\ =\ B$, where A is the matrix of coefficients of x_1, x_2, and x_3. X is the column vector, which will contain the solutions x_1, x_2, and x_3. B is the column vector of values on the right-hand side.

Answer: = $A^{-1}\ X\ = \ldots$

In [ ]:
```
% enter your code here
```

In MATLAB, you can also solve linear equations like this using the backslash operator `\` :

```
X = A\B
```

Try it:

In [ ]:
```
% enter your code here
```

Functions

MATLAB includes many standard functions (and constants). Each function is a block of code that accomplishes a specific task. With practice, we can even write special MATLAB files (m-files) to save re-typing the same commands.

Common functions

`sin` , `cos` , `log` ($\log_e$), `log10` ($\log_{10}$), `exp` , `sqrt` , `mean` , `std` (standard deviation).

Common constants

`pi` (= π) returns 3.1416 (rounded to 5 sig. figures). The variables `i` or `j` represent the square root of -1 (imaginary number)

Try the following sine function:

In [ ]:
```
x = sin(pi/4)
```

We wish to plot the sine wave of frequency 3 rad/s and amplitude 1 over a period of time between 0 to 20 seconds (in steps of 0.1 seconds).

Write down the sine wave expression: `y(t)` = ?

Calculate the vector y for the values of time given. Graph the results using an appropriate plot command.

In [ ]:
```
% insert your code here
```

Exercise 6: Functions

A sine wave of amplitude 5 and a frequency of 0.1 Hz is applied to the input of an analogue device, which has a constant gain of 1.5 over all frequencies. Calculate the output signal for the interval 0 to 10 seconds in increments of 0.1 seconds.

Use the command

```
plot(t,y)
```

to see a graph of this signal. What is the frequency of the sine wave on the plot? (Use the `grid` command to help work this out (approximately)).

Help window/Tips

To determine the usage of any function, enter:

```
» help function name
```

Try

In [ ]:
```
help sin
```

Documentation

MATLAB has an extensive online documentation system. To access it type:

In [ ]:
```
doc
```

The documentation system covers all the MATLAB functions, applications and toolboxes. Once there you have the choice of browsing the documentation or searching for particular functions.

You can give an argument to the doc function to get help on a particular function or concept. For example,

In [ ]:
```
doc elfun
```

provides an index of basic functions.

Tips

- Keep the documentation window open for quick access to help. Use search often!
- MATLAB variables are case sensitive, that is, `temp` and `Temp` would be two different variables.
- We can use the command `who` to see the variables in the MATLAB workspace.
- We can get the value of a particular variable at any time by typing its name. Use `who` to list all the variables in the workspace. Type the name of one of these variables to show its current value.
- We can use `clear all` to clear all the variables from the workspace or `clear` *variable name* to clear only a single variable.
- We can also have more than one statement on a single line, so long as we separate them with either a semicolon or comma. Try:

In [ ]:
```
gain = 20*log10(2); ang=angle(0.5*i);
```

Now, enter gain and angle to see the results.

In [ ]:
```
% enter your code here
```

If we do not assign a variable to a specific operation or result, MATLAB will store the outcome of the operation in a temporary variable called ans. For example

```
» 3+6

ans =

     9
```

Function Inputs and Outputs

Each function in MATLAB has a number of inputs and output variables. We should define all the necessary inputs before we run a function. (We can use the `help` function to check the inputs and outputs of any function we are going to use.) For example, if we wanted to find out about the use of the `abs` function, we could type

```
» help abs
```

MATLAB returns:

```
abs    Absolute value.
    abs(X) is the absolute value of the elements of X. When
    X is complex, abs(X) is the complex modulus (magnitude) of
    the elements of X.

    See also sign, angle, unwrap, hypot.

    Reference page for abs
    Other functions named abs
```

Therefore by typing

```
y = abs(x)
```

the variable `y` would hold the absolute values of `x`.

If we want to find the modulus and the angle of a complex number, we can enter the following:

In [ ]:
```
s = j*pi/4 % define complex number s
```

In [ ]:
```
g = 1/s % define complex function g
```

In [ ]:
```
mag = abs(g) % find the magnitude: output: mag input: g
```

In [ ]:
```
ang = angle(g) % find the angle in radians: output: ang input: g
```

Once we defined the output variables, they will be stored in MATLAB workspace and we can manipulate them as we wish.

Transfer functions

A transfer function can be entered into MATLAB using several different commands:

Method 1:

1. We can use the command `tf(num,den)`, where num and den are vectors of coefficients of the numerator and denominator polynomials, respectively. Enter the transfer function:

$$g_1(s) = \frac{3s+1}{s^2+3s+2}$$

In [ ]:
```
num = [3, 1]
```

In [ ]:
```
den = [1, 3, 2]
```

In [ ]:
```
g1 = tf(num,den) % enter the transfer function
```

1. Alternatively, we can define `s` as a transfer function and then use it to 'write' the transfer function directly.

In [ ]:
```
s=tf('s');
```

In [ ]:
```
g1 = (3*s + 1)/(s^2 + 3*s + 2)
```

Method 2:

We can use the command `zpk(zeros,poles,gain)`, where `zeros`, `poles` and `gain` are vectors of the zeros, poles and gain of the transfer function. For example, we can enter the following transfer function in the form

$$g_2(s) = \frac{4(s+5)}{(s+3)(s+10)}$$

In [ ]:
```
zeros = [-5]
```

In [ ]:
```
poles = [-3, -10]
```

In [ ]:
```
gain = 4
```

In [ ]:
```
g2 = zpk(zeros,poles,gain)
```

or equivalently

In [ ]:
```
g2 = zpk([-5],[-3 -10],4) % enter the transfer function
```

Alternatively, we can define `s` as a transfer function and then use it to 'write' the transfer function directly.

In [ ]:
```
s=zpk('s');
g2 = 4*(s + 5)/((s + 3)*(s + 10))
```

If we have a factor with complex poles in the transfer function,

The complex roots of the denominator can be entered easily as

In [ ]:
```
g3=zpk([5],[-2+3j,-2-3j],4)
```

If you want to use the symbolic version, you have to enter the complex poles as the coefficients of the quadratic $s^2 + 4s + 13$ as:

In [ ]:
```
g3=4*(s+5)/(s^2 + 4*s + 13)
```

If you need to, you can find the coefficients of the quadratic using conv:

In [ ]:
```
q =conv([1, 2+3j],[1, 2-3j])
```

Try entering the following transfer functions:

$$g4(s) = \frac{12(3s+2)}{(s+3+j7)(s+3-j7)}$$

In [ ]:
```
% put your code here
```

$$g_5(s) = \frac{2(s+8)}{(s+0.1)(s+2)(4s+6)}$$

In [ ]:
```
% put your code here
```

Transfer function manipulation

Once the various transfer functions representing components in a system have been entered, we can combine them together.

Use the transfer functions

$$g_6(s) = \frac{1}{s+2}$$

and

$$g_7(s) = \frac{5}{s+3}$$

In [ ]:
```
% put your code here
```

to complete the following.

Addition

In [ ]:
```
g6 + g7
```

Subtraction

In [ ]:
```
g6 - g7
```

Multiplication

In [ ]:
```
g6 * g7
```

Division

In [ ]:
```
g6 / g7
```

Combination of operations

In [ ]:
```
g6/(1+g6)
```

Now, that we have entered various transfer functions, it is easy to use some of the system analysis commands available from the various MATLAB toolboxes. The following commands represent some of the systems control analysis operations.

Time responses

- For a unit impulse response use the command: `impulse(g)`
- For a unit step response, use the command: `step(g)`
- For a step response of magnitude `K`, simply multiply the transfer function, `g`, by the numerical value of `K`: `step(K*g)`

Poles and zeros

- To see the roots of the numerator (called the zeros) of a transfer function use the command: `zero(g)`
- For the roots of the denominator (called the poles) of a transfer function (zeros of the denominator of g(s)) use the command: `pole(g)`
- For a pole-zero map use the command: `pzmap(g)`

Block diagram feedback

To determine the closed-loop transfer function for a system with negative feedback we can use the command:

```
» gc = feedback(g, h)
```

where `g` is the forward transfer function and `h` represents the transfer function of the components in the feedback path. If positive feedback is required then the following command can be used:

```
» gc = feedback(g, h, +1)
```

Exercise 7: Transfer Functions

Given the following transfer functions,

$$g_1(s) = \frac{3s+1}{s^2+3s+1}$$

and

$$g_2(s) = \frac{2s+3}{s^3+1}$$

- enter $g_1(s)$ using row vectors
- enter $g_2(s)$ using the `s = tf('s')` method

In [ ]:
```
% put your code here
```

1. Find the roots of the numerator and denominator polynomials using the `roots` function`
2. Compare with the results using the `zero` and `pole` functions
3. Examine where the roots are using the `pzmap` function

In [ ]:
```
% put your code here
```

Plotting

It is easy to create plots in MATLAB. Suppose we make a time vector and then compute the sine values of the vector at each time.

In [ ]:
```
t=0:0.25:10;
y = sin(t);
plot(t,y)
```

This illustrates that basic plotting is very easy in MATLAB, and the `plot` command has extensive add-on capabilities. Some useful features are given here.

Figure command

When we plot a graph, MATLAB opens a window called a "Figure Window". Every time we plot a graph, this figure window is updated. If we want to keep the old graphs, we can open a new window by using the command:

```
» figure
```

Try:

In [ ]:
```
step(g3)
```

In [ ]:
```
figure
step(2*g3)
```

Plotting several responses on the same axis

Plot the first response and then enter the command:

```
»  hold on
```

Try these commands:

In [ ]:
```
figure(2);
step(g3)
hold on
step(2*g3)
step(0.5*g3)
```

Enter hold off to return to the default mode.

In [ ]:
```
hold off
```

Finding coordinates of a point on a graph

Use `ginput(N)` (N is the number of points at which co-ordinates may be found). This will give a cursor, which we can move to a point on the figure using the mouse, and then press the left-hand button to see the co-ordinates in the MATLAB Command Window.

Example:

Type the following (these commands don't work in the Jupyter notebook. You have to run them from the command line).

```
step(g3)
```

and then

```
ginput(1)
```

Use the mouse to click on the final value of the step response. Look in MATLAB Command Window to see the values of x and y.

Labelling plots and axes

Use the following commands:

In [ ]:
```
step(g3)
xlabel('Time (sec)')
ylabel('Output')
title('Tutorial Example')
```

The following command allows text to be placed on a graph. This can be useful, for example, for labelling certain areas of a plot or for pointing out specific features on the plot. Again, it will only work from the command line, not in Jupyter notebook.

```
gtext('Step Response')
```

Move the cursor to a point on the graph, and click the left-hand button.

Frequency response graphs

The command `semilogx` is the same as plot except a logarithmic (base 10) scale is used for the x-axis. See the exercise for an example.

Exercise 8: Plotting

The input-output relationship of an RC circuit can be represented by the following function of a complex variable:

$$g(j\omega) = \frac{V_o(j\omega)}{V_i(j\omega)} = \frac{K}{j\omega\tau + 1}$$

where K is the gain and τ is the time constant.

- For $K = 10$ and $\tau = 5$, calculate the gain and phase of V_o/V_i for w = 0.01, 0.05, 0.1, 2, 5, 10 (rad/s).
- Plot the gain and phase using the `semilogx` function.

Hints:

- Enter the values K and τ as `K` and `tau`. Define a frequency vector, `w`.
- Calculate `g = K./(j*w*tau + 1)`. Note the use of the operator `./`.
- Calculate the gain in dB (`gain = 20 * log10(gain)`) and the phase in degrees (`phase = phase * 180/pi`).
- Plot the response using `semilogx(w,gain)` and `semilogx(w,phase)`.
- Label both plots.

In [ ]:
```
% put your code here
```

Summary

What we have covered

- To enter numbers vectors, matrices and transfer functions easily into the package
- To perform several different calculations and operations on different variables
- To plot graphs of results
- To form a step response of a transfer function and plots the output response

What we have not covered

- The Simulink tool which is used for system simulation.
- State space models.
- Digital systems.

Please consult the on-line documentation to find out how to use these facilities.