University of Illinois at Urbana-Champaign
Department of Electrical and Computer Engineering

ECE 298JA Fall 2017

MATLAB/Octave Tutorial

---

1 Overview

The goal of this tutorial is to help you get familiar with MATLAB/Octave and to learn the basics of how to do computations and create plots in MATLAB/Octave.

1.1 Matlab/Octave

Either the University of IL, or Mathworks, the company that sells MATLAB has had a change in policy, which has resulted in a major change in policy regarding the license fee. For this reason everyone is scrambling for alternatives. One is Python and the other is Octave. As best I know Python has more of a learning curve. If you already know (have used) MATLAB, you may find Octave more to your liking. If not, then use Python, which is presently used in several other courses here at the Univ. of IL.

1.2 Octave

The program Octave works just like MATLAB, however the license is very different. Octave is open-source and free. You can download it from octave.org. If you have any problems installing or running it, talk to me. Students in the past have been very happy with its performance, with few minor glitches which have work-arounds. If the install takes more than 1 hour, stop and talk to Prof. Allen.

2 Getting Started

- Get familiar with MATLAB/Octave by using tutorials and demos found in MATLAB/Octave. You can click `Start` ≫ `MATLAB/Octave` ≫ `Demos` to start the help screen. For octave `help demo` works for me, out of the box. For Octave you may need to install the documentation. It may be easier to go to the web for some of the help menues.

- Your MATLAB/Octave interface will have a number of windows tiled together, such as the command window, current directory, workspace (which shows your current variables), and editor (where you can write and save scripts). You can move these windows around by clicking and dragging with the mouse.

- In the command window, type `help` or `doc` followed by a command to get information about the command. Try the following:

```
>>help plot
>>help stem
>>help cos
>>help sin
>>help exp
>>doc for
>>doc ones
>>doc zeros
```

- What if you do not know a command name? Type `helpdesk` to start the help window. You can then type keywords in the search bar. Also, Google can be very useful to find MATLAB/Octave commands, since its search engine is more powerful.

- Variables in MATLAB/Octave can hold numbers (dimensions 1 x 1), vectors (dimensions 1 x N or N x 1) or matrices (dimensions N x M):

```
>>x = ones(1,1)
>>x = ones(1,5)
>>x = zeros(5,1)
>>x = zeros(5,6)
```

- Putting a semicolon after a variable assignment prevents unnecessary 'echoing' (try the following):

```
>>x = ones(2,2)
x =
      1    1
      1    1
>>x = ones(2,2);
```

- You may repeat recently used commands by hitting the up arrow until you find the desired command, then press Enter.

Exercise 1
1) Find some MATLAB/Octave commands that generate random numbers (there are a few).
2) Generate a 3×3 random matrix.

- The result of a computation can be assigned to a variable (e.g. `x`, as shown below). If you type a variable name into the command window without a semicolon, MATLAB/Octave will show you what is stored in that variable. If your computation is not assigned to a variable, MATLAB/Octave stores it in the variable `ans`.

```
>>x = 0.25*0.25;
>>x
x=
      0.0625
>>0.25*0.25
ans =
      0.0625
```

- It is very easy to manipulate complex numbers in MATLAB/Octave. You can assign a complex number of the form `a+b*1i` directly to a variable and perform arithmetic operations (it is recommended to use `1i` or `1j` instead of `i` or `j` to accelerate your code):

  ```
  >>x = 2+3*1i
  >>y = 3+4*1j
  >>x+y
  ans =
        5.000 + 7.000i
  ```

Exercise 2

1) Find MALAB commands to calculate the conjugate, absolute value, real and imaginary part of a complex number.

2) Calculate the conjugate, absolute value, real part and imaginary part of $3 + 4i$

- 'Loops' allow us to perform an operation repeatedly based on an index. In MATLAB/Octave, a simple 'for loop' to construct the vector `x=[0,1,2...9]` can be written (where `n` is the index) as,

  ```
  >> for  n = 1:10
         x(n) = n - 1;
     end
  >> x
  ```

- We will learn later how to save code such as loops in a script. For now, notice that MATLAB/Octave does not perform the computation (e.g. does not return to '>>') until you type 'end'

- The above loop generates integers from 0 to 9. You can also use a 'while loop,' which performs the operation until the specified condition is met:

  ```
  n=0
  while n <10,
      x(n) = n-1;
      n = n+1;
  end
  ```

- In general MATLAB/Octave is not very efficient with loops. We will see later that one can avoid some loops by using the `colon` operator (for instance, above you can just set `x=[0:9]`). See more information about flow control in MATLAB/Octave in `help` under :

  `MATLAB/Octave/Getting Started/Programming/Flow Control` .

3 MATLAB/Octave functions for creating & manipulating vectors and 'signals'

'Signals,' as we refer to them in electrical engineering, are sets of values that may be functions of time or space (e.g. $y(x) = f(x)$, $y(t) = f(t)$, $y(t) = f(x(t))$, a speech waveform, a video), which are often

expressed as vectors. In ECE 298, in addition to using MATLAB/Octave for large calculations, you will explore various mathematical functions using MATLAB/Octave, which are related to real-world signals and physics. To learn how to manipulate real-world signals, you may consider taking Signal Processing (ECE 310/311) in the future!

- The colon operator can be used to define a vector. Let us say we want to create a vector `x` which holds the integers from 0 to 100. One way is to use a loop. Another way to do this is shown below,

    ```
    >> x = [0:100];
    ```

- All vectors in MATLAB/Octave are indexed starting with 1

    ```
    >> x(1)
    ans =
        0
    >> x(2)
    ans =
        1
    ```

- Trigonometric functions, such as sines and cosines, are used often in mathematics and maybe composed to form 'signals' (e.g. music). A continuous-time, complex exponential function has the form α^t, where α is a complex scalar. Sine and cosine functions can be built from complex exponential functions by setting $\alpha = e^{\pm i2\pi f}$,

$$\cos(2\pi ft) = \frac{1}{2}\left(e^{i2\pi ft} + e^{-i2\pi ft}\right)$$
$$\sin(2\pi ft) = \frac{1}{2}\left(e^{i2\pi ft} - e^{-i2\pi ft}\right)$$

- Note that in MATLAB/Octave, `t` will not be a continuous variable, rather we can solve for the sine and cosine values as a set of time values stored in `t`.

MATLAB/Octave has functions `cos`, `sin` and `exp` to create vectors using these mathematical functions.

Exercise 3

Generate the following vector

$$x(t) = \sin(\frac{2\pi t}{5}), \text{for 501 points over the interval } t = [0, 5]$$

1) Define t

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

2) Generate $x(t)$

```
>> x = sin(2*pi*t/5);
```

3) Plot this sine function for the given values of `t` using `plot` command

```
>> plot(t,x);
```

4) Try this with far fewer samples, `t=0:5`. In this case the curve that results from the `plot` command is misleading - why?

- MATLAB/Octave has several commands to help you label the plots appropriately, as well as to print them out. `title` places its argument over the current plot as the title. `xlabel` and `ylabel` allow you to label the axes. Every plot or graph you generate must have a title and the axes must be labeled clearly.

Exercise 3 (continued)

4) Label the plot

```
>> title('A sine signal: sin(2*pi*t/10)');
>> xlabel('t (Seconds)');
>> ylabel('Amplitude');
```

- MATLAB/Octave also allows you to add, subtract, multiply, divide, scale and exponentiate vectors. let us define the two signals `x1` and `x2`,

```
>>x1 = sin((pi/4)*[0:15]);
>>x2 = cos((pi/7)*[0:15]);
```

Try the following

```
>>y1 = x1 + x2
>>y2 = x1 - x2
>>y3 = x1 .* x2
>>y4 = x1 ./ x2
```

```
>>y5 = 2*x1
>>y6 = x1 .^2
>>y7 = x1 * x2
>>y8 = x1 * x2'
```

For multiplying, dividing, and exponentiating on a term by term basis ('element-wise'), you must precede the operator with a period .* instead of * alone. Also note x2' converts the row vector x2 into a column vector, computing the conjugate transpose ('hermitian' transpose) of the argument. If you want to transpose x2 without conjugating it use x2.'.

4   MATLAB/Octave scripts and functions

- MATLAB/Octave allows us to create m-files to save lists of commands. There are two types of m-files: 'scripts' and 'functions'

- A command 'script' is a text file of MATLAB/Octave commands whose filename ends in a **.m**, saved in the current working directory (or elsewhere in your MATLAB/Octave path). A script has no input or output arguments. If you type the name of the file (without .m) in the command prompt, the commands contained in the script file will be executed.

Exercise 3 (continued)
5) Create a script file based on Example 1 and run it from the terminal.

- An m-file implementing a 'function' is a text file with a title ending in '.m' whose first word is function. The rest of the first line of the file specifies the input and output arguments.

- The following m-file is a function called foo. It accepts input x and returns y and z which are equal to 2*x and 5/9*(x-32) respectively

```
function [y,z] = foo(x)
%[y,z] = foo(x) accepts a numerical argument x and
% returns two arguments y and z, where y is 2*x and z is (5/9)*(x-32)
y = 2*x;
z = (5/9)*(x-32);
```

Copy and paste the above text into a .m file, and save it in your current directory as foo.m. Try

```
>> help foo
>>[y,z] = foo(-40)
>>[y,z] = foo(225)
```

6

Exercise 4

Create a function that takes the frequency of the signal as the input and the signal as the output. Run the function from the terminal. Try $f = 1/5, 1/10$, etc.

```
function x = my_sine(f)
%% generating a sine signal
t = 0:0.01:5; x = sin(2*pi*f*t);
stem(t,x);
title(['A sine signal: sin(2*pi*t',num2str(f),')']);
xlabel('t (Seconds)'); ylabel('Amplitude');
```