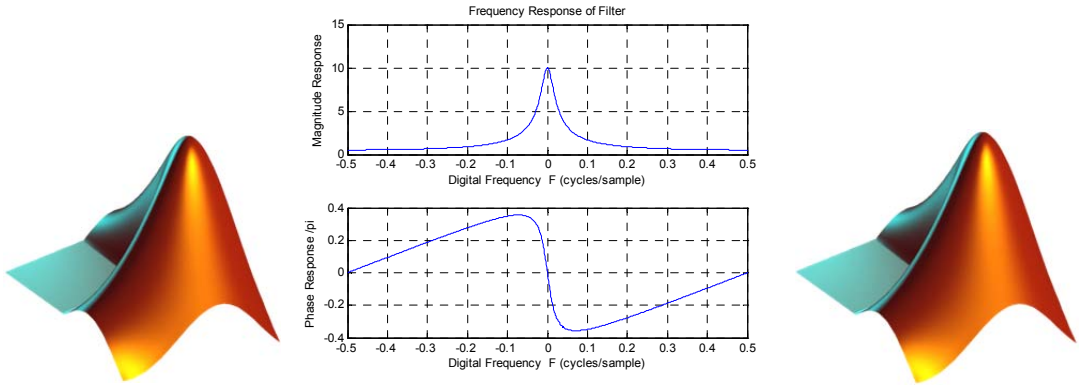
# EE 459 - Project 1
# INTRODUCTION TO MATLAB




**Learning Objectives:**

1. Understand how to use Matlab to do the following:
    a. Create arrays of values with
        i. Individual entered values
        ii. Fixed increment sequences
        iii. Fixed number of point incremental sequences
        iv. Linear or logarithmic increment sequences
    b. Compute complex function values using matrix input sequences
    c. Plot complex function magnitude and phase.
        i. Linear axis or Bode Plot formats
        ii. With labels, titles, and grids
        iii. With multiple subplots per figure
    d. Create sinusoidal signal sequences with particular magnitudes, phases, and sampling rates (sample time intervals)
    e. Create overlay plots (multiple plots on one set of axes)

## 1) Plotting the DTFT Frequency Response of an IIR Filter Using Matlab

Using Matlab, we will create several versions of frequency response plots for an IIR digital filter. This is an in-class, instructor-guided group project.

To prepare for this exercise, please copy the following files from PolyLearn to your default Matlab directory:

- *EE419_HF_exp1_proto.m* renamed to *EE419_HF_exp1.m*
- *HF_values.mat*

To demonstrate your success with this project, please turn in the following:

- Copy of all plots generated.
- Copy of your Matlab m-file that generated the plots.

When you are finished, **be sure to *always* copy your m-files to a place that you can retrieve them again in the future** (copy them to a memory stick, email them to yourself, etc.); as there is no guarantee they will still be on the lab computer next time you come in!!

## 2) Matlab Tutorial and Help Files

If you are unfamiliar with Matlab, you can quickly familiarize yourself with this very powerful and versatile tool using the "Help" menus and tutorials.

If you are working on a computer with Matlab installed, you can access the Help information by launching Matlab, and then either:

- Hit "F1" to bring up Help window.
  - Click on "Getting Started" to begin following the tutorial.
- Type "help" in the Matlab Command Window
  - Prints a list of links to help information on specific types of available functions
- To find out about the syntax and function arguments of any particular Matlab function, simply type in "help" followed by the name of the function in the Command Window (example: > **help plot**); and Matlab will print the help information for that particular function.

If you wish to learn Matlab but do not have access to a computer with Matlab installed, you can access similar tutorial information from either:

- The Mathworks website: http://www.mathworks.com/access/helpdesk/help/techdoc/matlab.html
- The EE459 PolyLearn site:
  - The tutorial for the "Student Version" of Matlab is available in PDF form, in the *Project 1* resources.

In these tutorials, some items you can skip for now would include:

- The Magic Function
- Linear Algebra
- Graphics - Mesh and Surface Plots
- Graphics - Images
- Graphics - Handle Graphics
- Graphics - Animations
- Programming – Other Data Structures
- Anything after: Scripts and Functions – Types of Functions

## 3) General Matlab Function to Plot Frequency Response from H(F) and F Values [Function M-file, without returned values]

Using the m-file created together in Section 1 (above) for plotting a particular DTFT frequency response function as your basis, create your own Matlab **FUNCTION** m-file for plotting any DTFT frequency response from its complex H(F) data point values (rather than the H(F) analytical function). The m-file should use the following function header format:

```
function plot_freq_responses(Fd, HF, fsample, figure_num)
```

where the arguments passed to it are:

*Fd* = an array of digital frequency values (in units of cycles/sample) that correspond to the H(F) frequency response data values

*HF* = an array of complex H(F) DTFT frequency response values to plot

*fsample* = sampling frequency (in units of samples / second)

*figure_num* = number of the 1st figure to use for the two plots

The user specifies the arrays containing the values of the complex "`HF`" data points and the array of corresponding digital frequencies "Fd" that go with each H(F) point, and your function plots |H(Fd)| *vs* Fd, and ∠H(Fd) *vs* Fd.

This function should create 2 different versions of the frequency response plot (two figures); each figure having 2 subplots - one for the magnitude response (on top) and one for the phase response (below).

- All plots should use a **linear** scale for the **frequency** axis (not log scale).
- Normalize the phase plot values so that the phases are shown as multiples of π (*i.e.* the range of the vertical axes of the phase plot should be -1 to +1, representing –π to +π radians).
- Label all axes appropriately, and provide a title for the each plot.

The two figures created should be the following:

(1): Plot the H(F) magnitude and phase vs. **digital** frequency, using standard line plots and a **linear** scale for the H(F) magnitude plot. Plot this in Figure # = *figure_num*.

(2): Plot the H(F) magnitude and phase vs. equivalent **analog** frequency [assuming ideal sampling], using standard line plots and a **dB** scale for the H(F) magnitude plot. Plot this in Figure # = *figure_num* + 1.

Test data to run on this program and verify its results are provided on PolyLearn. The .mat file: HF_values.mat provides both data arrays needed to test your program (**frequencies** and **HF_values**). You should **assume a sampling rate of 1000 samples/second**.

Test data to run on this program and verify its results are provided on PolyLearn. The .mat file: *HF_values.mat* provides both data arrays needed to test your program (**`frequencies`** and **`HF_values`**).

These test data arrays can be loaded into Matlab using the command:

```
load HF_values
```

**Turn in the following:**

- A printout of the m-file for your function
- A printout of the 2 figures produced; using the data provided on the PolyLearn site.

Refer to the Matlab Help information provided for:

```
plot, stem, abs, angle, title, xlabel, ylabel, subplot
```

## 4) Creating a Unit Sample Sequence [Matlab Function with Returned Values]

Using Matlab, create a **function that returns** a complete Unit Sample Sequence, beginning at n = 0, and extending for the number of samples specified by a user. The function should have the form:

```
function [dn, n] = unit_sample(number_of_samples);
```

Where the values returned are:

`dn` – has the unit sample sequence values

`n` – has the sample index values (starting at 0) corresponding to each of the dn samples

For example, after calling the function as:

```
[dn, n] = unit_sample(3);
```

The values in the returned arrays should be:

```
dn = [1 0 0]          n = [ 0 1 2 ]
```

NOTE that the function you create computes and returns the requested length sequences; it should NOT do any plotting. The plotting should be done by another m-file that **calls your function** and then plots the values returned using a "stem" plot, with the "bubbles" on top replaced by "dots"

a. For example, plot the values using the command : `stem(n, dn,'.');`

`stem( )` – works like `plot( )`, except using discrete stems for the plot.

`'.'` - changes the stem plot to use dots at the end of the stems, instead of circles o

Turn in the following:

- A printout of the m-file for your unit_sample( ) function
- A printout of the m-file that calls your unit_sample( ) function and plots its output.
- A printout of a stem plot of your function's output for a length = 16 sequence.

Refer to the Matlab Help information provided for:

`stem, zeros`

## 5) Creating Discrete-Time Sinusoidal Signals in Matlab, and Observing Sampling Effects

Use Matlab to produce several overlay plots (multiple plots in ONE figure, all on ONE set of axes) of a single-frequency sinusoidal signal, as it would appear using different sampling rates and phase angles. For each sample rate, define the "time" sample values in a vector (starting at t = 0 seconds) using the sampling rate to determine the spacing between time samples; and apply each time vector to a basic `cos()` function with the correct amplitude, frequency and phase shift parameters. The values you will plot are a sampled version of the continuous-time signal:

$$x(t) = A \cos( 2\pi f_a t + \theta )$$

| Analog Frequency $f_a$ (Hz) | Amplitude A | Phase Shift (radians) | Plot Duration (seconds) | Sampling Rate $F_s$ (samples/sec) | Plot Color | Plot Line Type | Plot Line Width (points) |
|---|---|---|---|---|---|---|---|
| 10 | 1 | $\pi / 3$ | 0.5 | 200 | blue | solid | 4 |
| 10 | 1 | $\pi / 3$ | 0.5 | 60 | red | solid | 3 |
| 10 | 1 | $\pi / 3$ | 0.5 | 20 | green | solid | 2 |
| 10 | 1 | $\pi / 2$ | 0.5 | 20 | green | dashed | 5 |
| 10 | 1 | $\pi / 3$ | 0.5 | 12 | black | solid | 6 |

Create an m-file (does not have to be a function) that computes and plots the required sinusoids. Be sure to label the axes of your plot, and add an appropriate title.

To help you create this plot, you should consult the Matlab help files for the following functions: `plot, cos, hold, title, xlabel, ylabel`. The line width of a plot can be changed by adding a special argument to the plot command. For example, to plot a **red** line with a line width of **6** points, the plot command would be:

```
plot( x, y, 'r', 'LineWidth', 6);
```

Turn in the following:

- The plot you created (I'll have to take your word on the colors, as this will probably be in glorious black-and-white!...unless you have access to a color printer.)
- A listing of your m-file that created the plots.
- Your interpretation of what you see in the plot. What does it show? Explain what's going on at the lower sample rates, for the last 3 cases in the Table above.