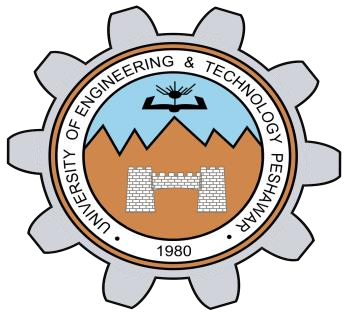
**Lab report 02**



**CSE 402L**

**Digital Signal Processing Fall 2024**

**Submitted by: Naveed Ahmad**

**Registration No.: 22PWCSE2165**

**Class Section: B**

**Semester :5th**

*“On my honor , as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work”*

Student Signature: \_\_\_\_\_\_\_\_\_\_\_\_

Submitted to:

**Dr. Yasir Saleem Afridi**

Oct 08, 2024

Department of Computer Systems Engineering

University of Engineering and Technology Peshawar

# 1.1 PLAYING WITH MATLAB

The following steps will introduce you to MATLAB by letting you play with it.

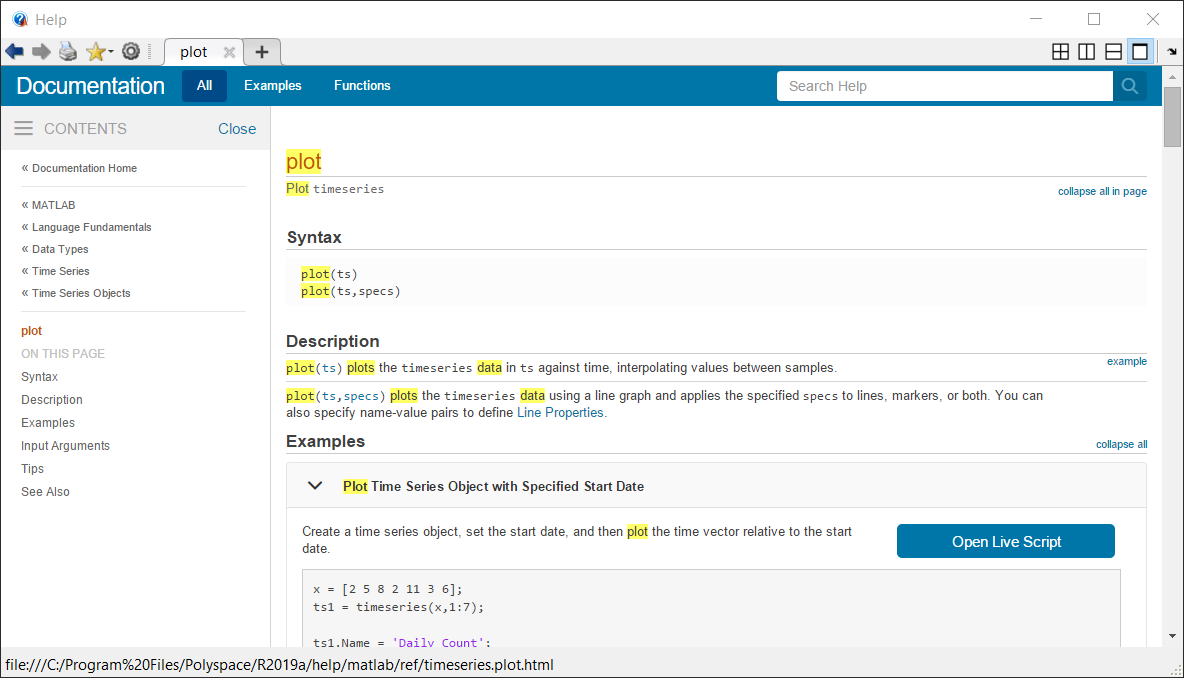
1. Run the MATLAB help desk by typing doc. The help desk provides a hypertext interface to the MATLAB documentation. Two links of interest are Getting Started and Getting Help in MATLAB. Both are under Documentation Set.

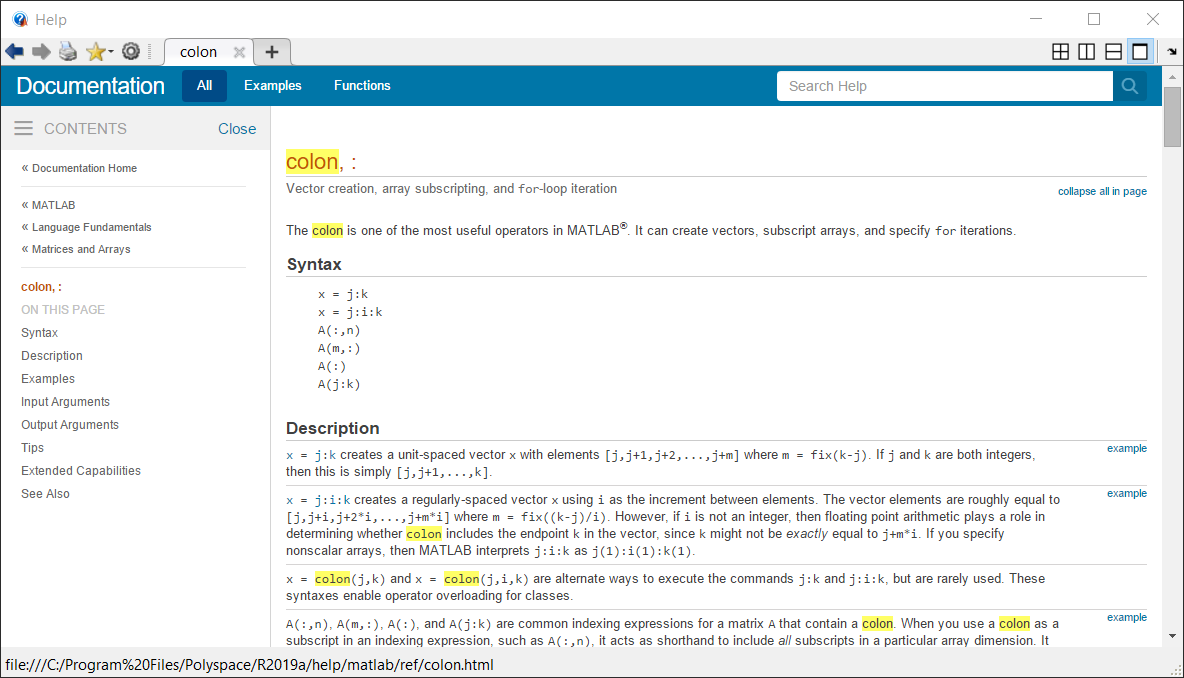
A screenshot of a computer

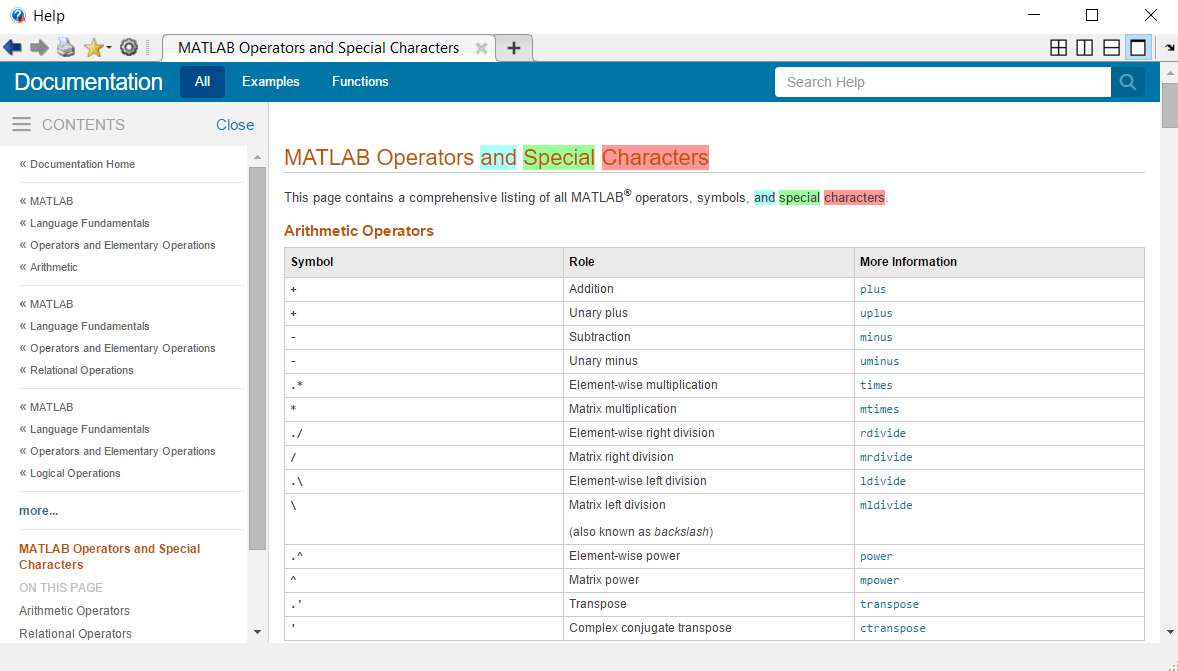
Description automatically generated

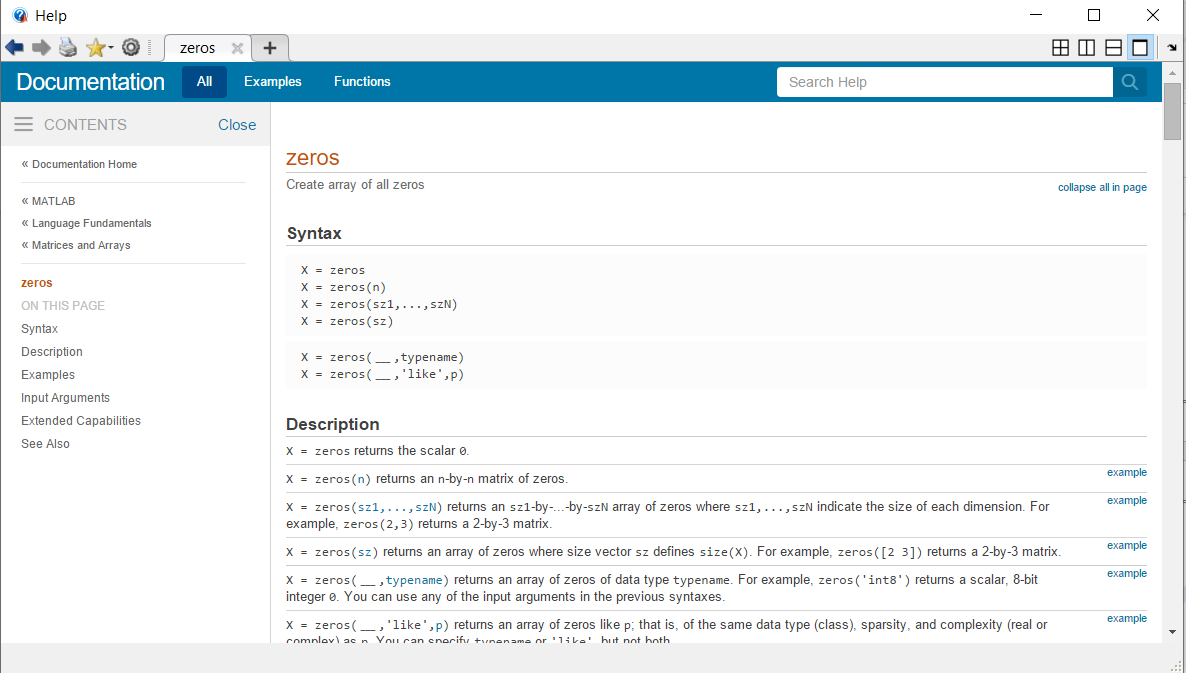
1. Explore the MATLAB helpwin capability available at the command line. Try the following:

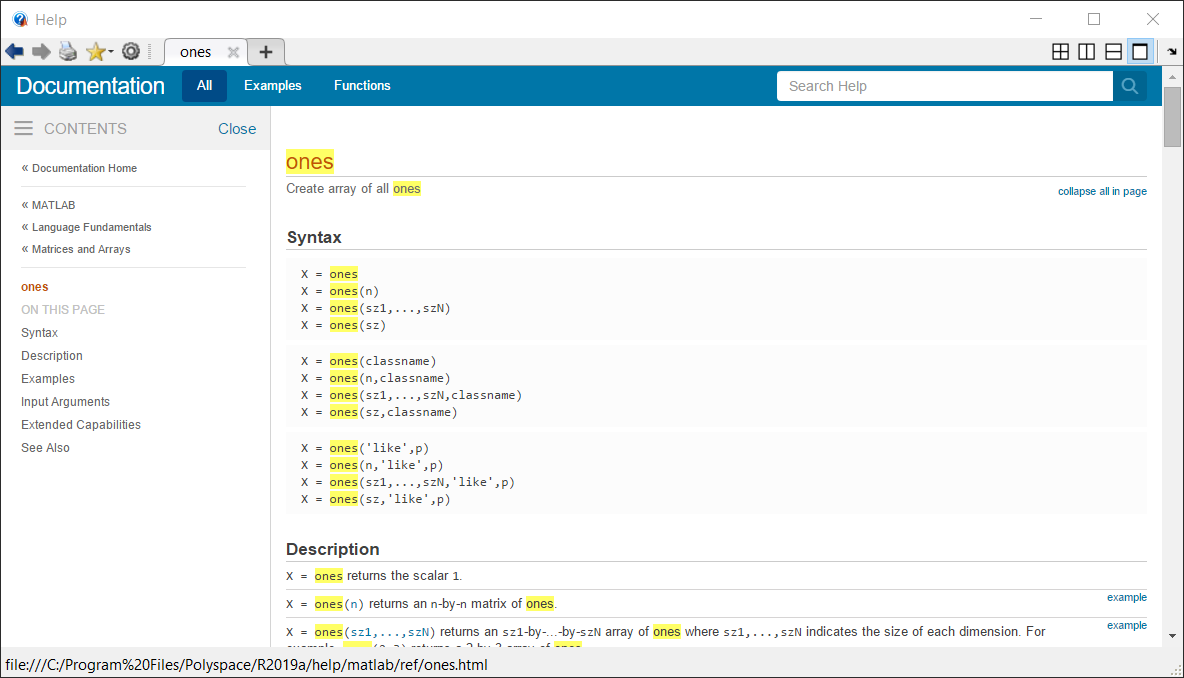
|  |  |
| --- | --- |
| helpwin helpwin plot |  |
| helpwin colon helpwin ops helpwin zeros helpwin ones | %<--- a VERY IMPORTANT notation |
| lookfor filter | %<--- keyword search |







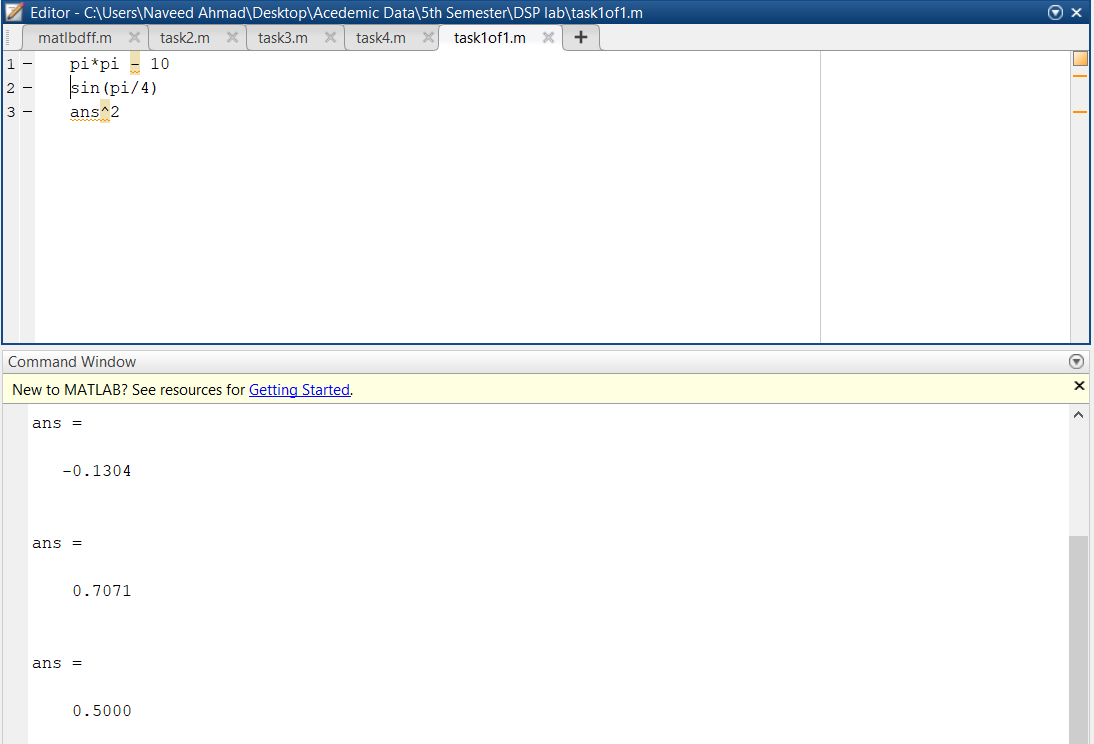




1. Use MATLAB as a calculator. Try the following:

pi\*pi-10sin(pi/4)

ans ˆ 2 %<--- "ans" holds the last result



* **Pi Squaring and Subtraction**:

the first operation calculates the square of pi and subtracts 10, resulting in approximately -0.1304.

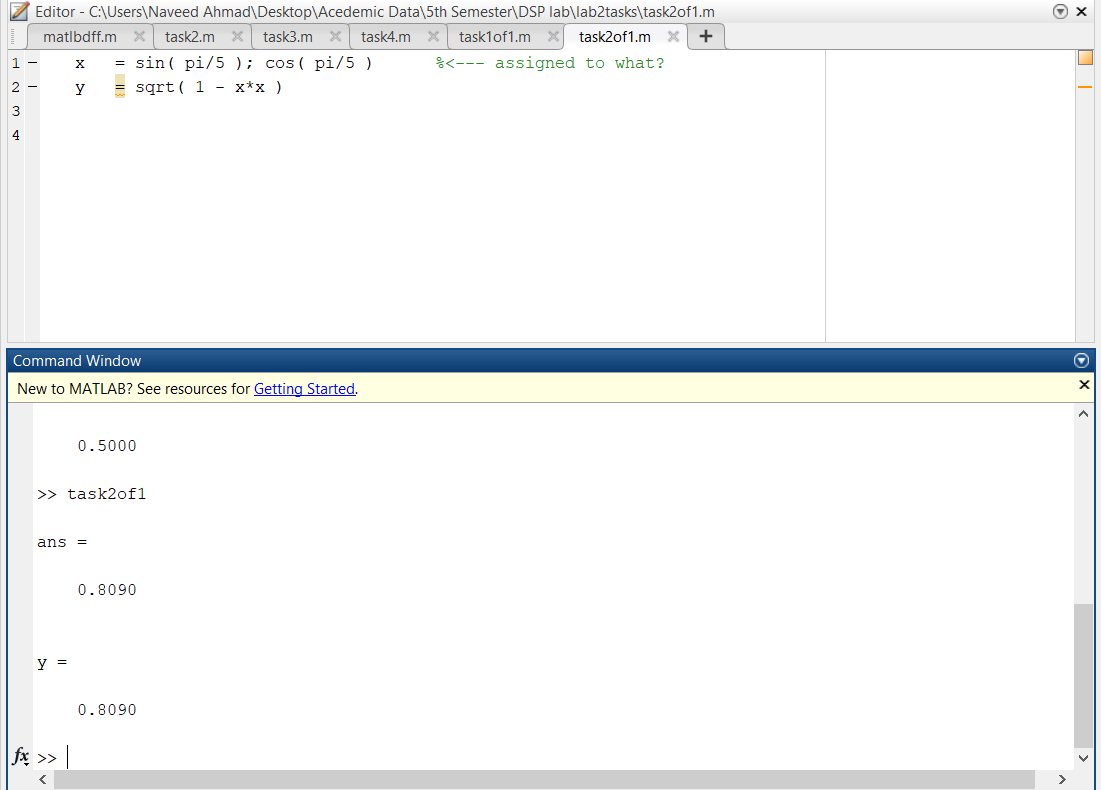
* **Sine Calculation**:

The second operation computes the sine of π4\frac{\pi}{4}4π, which gives 0.7071 (the sine of 45 degrees).

* **Squaring the Sine Result**:

The third operation squares the previous result (0.7071), yielding 0.5000.

1. Do variable name assignment in MATLAB. Try the following:
   1. = sin( pi/5 ); cos( pi/5 ) %<--- assigned to what?
   2. = sqrt( 1 - x\*x )



* + **Sine of Pi/5**:

x is calculated as the sine of π5\frac{\pi}{5}5π, giving 0.5878.

* + **Cosine of Pi/5**:

The code calculates the cosine of π5\frac{\pi}{5}5π, but doesn’t save it.

* + **Square Root Calculation**:

y is the square root of 1−x21 - x^21−x2, resulting in 0.8090.

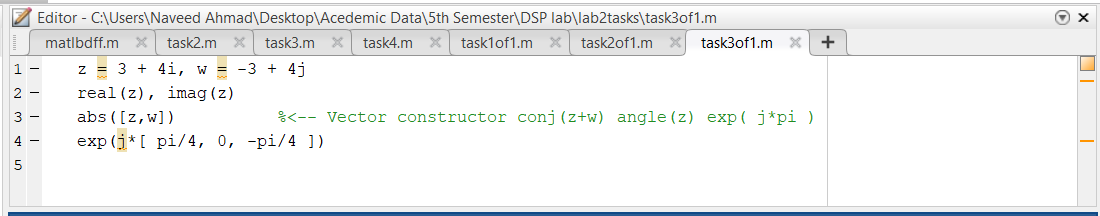
1. Complex numbers are natural in MATLAB. The basic operations are supported. Try the following:

z = 3 + 4i, w = -3 + 4j

real(z), imag(z)

abs([z,w]) %<-- Vector constructor conj(z+w) angle(z) exp( j\*pi )

exp(j\*[ pi/4, 0, -pi/4 ])



A screenshot of a computer

Description automatically generated

* + The code performs operations on complex numbers like real/imaginary parts, absolute values, conjugates, angle calculation, and computes exponentials using Euler's formula.

# WARM-UPS

## MATLAB Array Indexing

1. Make sure that you understand the colon notation. In particular, explain in words what the following MATLAB code will produce

jkl = 0 : 6

jkl = 2 : 4 : 17

jkl = 99 : -1 : 88

ttt = 2 : (1/9) : 4

tpi = pi \* [ 0:0.1:2 ];

A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Vector Creation**:

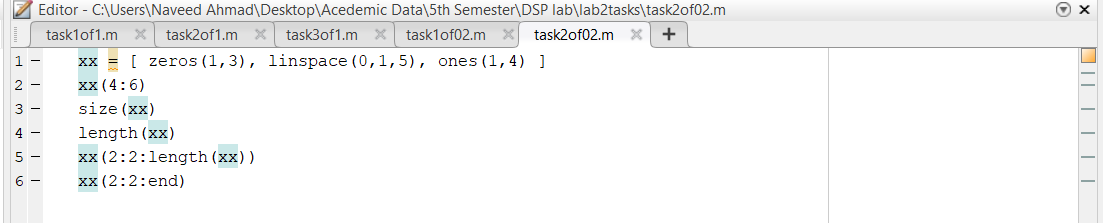
* It creates multiple vectors jkl, ttt, and tpi using different ranges and increments.

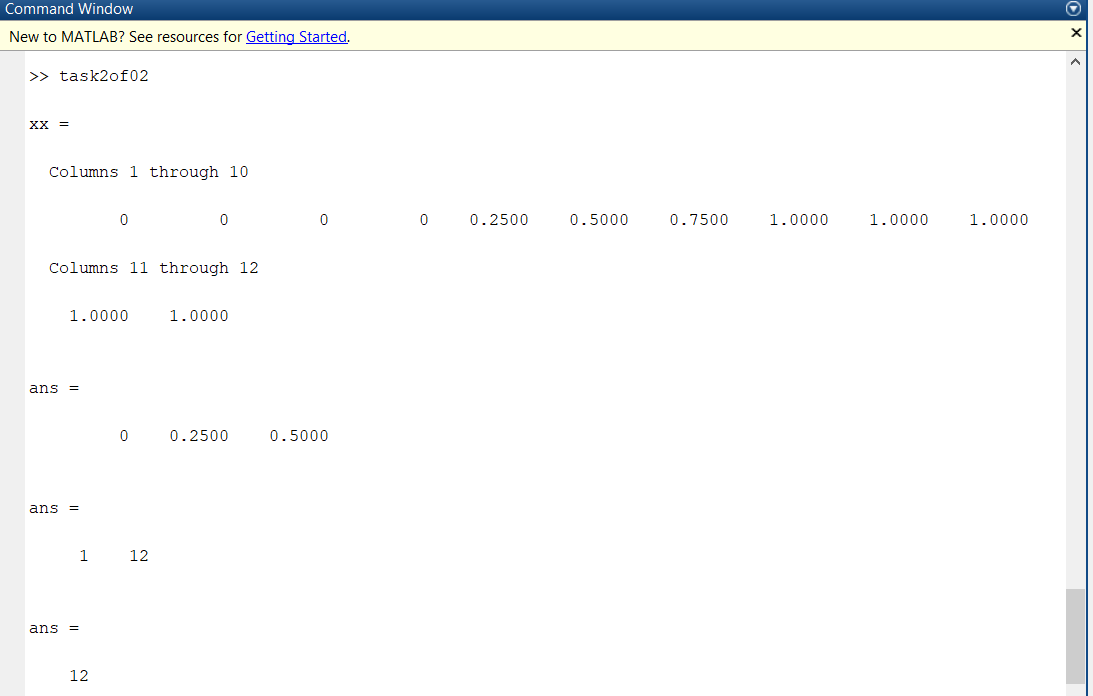
**Mathematical Calculations**:

* It multiplies pi with a vector [0:0.1:2], generating values in small steps of 0.1, and outputs the results in the Command Window.

(b) acting and/or inserting numbers into a vector is very easy to do. Consider the following definition of xx:

xx = [ zeros(1,3), linspace(0,1,5), ones(1,4) ] xx(4:6) size(xx) length(xx) xx(2:2:length(xx)) xx(2:2:end)



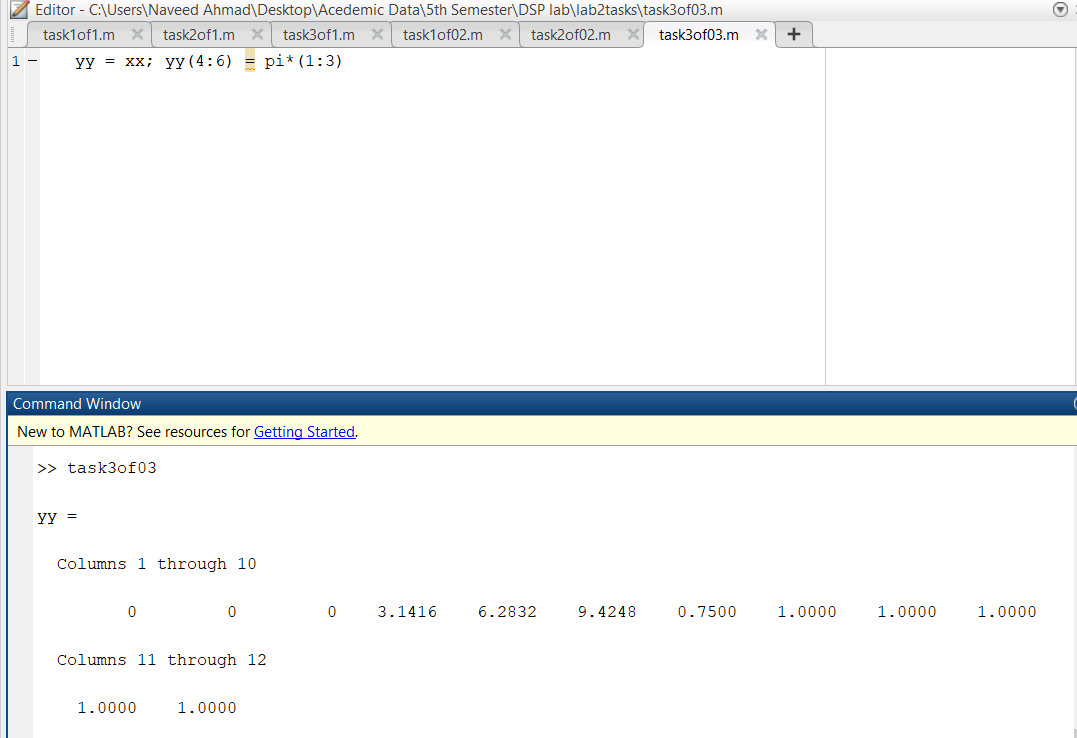




This code creates a vector xx by combining zeros, a linearly spaced sequence, and ones. Then, it selects and displays specific parts of the vector using different index ranges. The final output shows the length of the vector and every second element from it.

1. Observe the result of the following assignments:

yy = xx; yy(4:6) = pi\*(1:3)



A screenshot of a computer

Description automatically generated

## 2.2 MATLAB Script Files

1. Experiment with vectors in MATLAB. Think of the vector as a set of numbers. Try the following:

xk = cos( pi\*(0:11)/4 ) %<---comment: compute cosines

A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

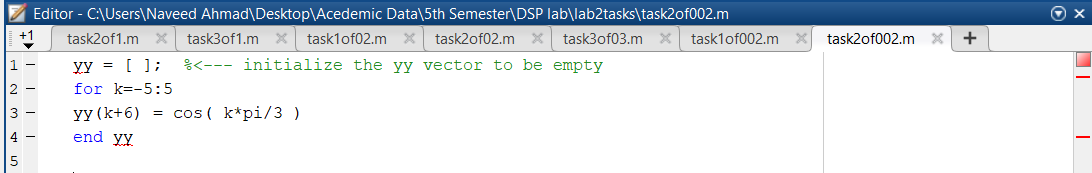
This code creates a vector xk containing the cosine values of angles from 0 to 11 multiplied by pi/4. The result is a sequence of cosine values for evenly spaced angles in radians

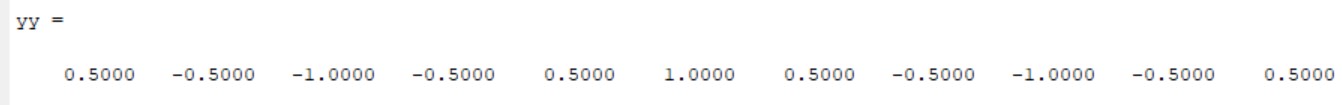
1. (A taste of vectorization) Loops can be written in MATLAB, but they are NOT the most efficient way to get things done. It’s better to always avoid loops and use the colon notation instead. The following code has a loop that computes values of the cosine function. (The index of yy() must start at 1.)

Rewrite this computation without using the loop (follow the style in the previous part).

yy = [ ]; %<--- initialize the yy vector to be empty for k=-5:5

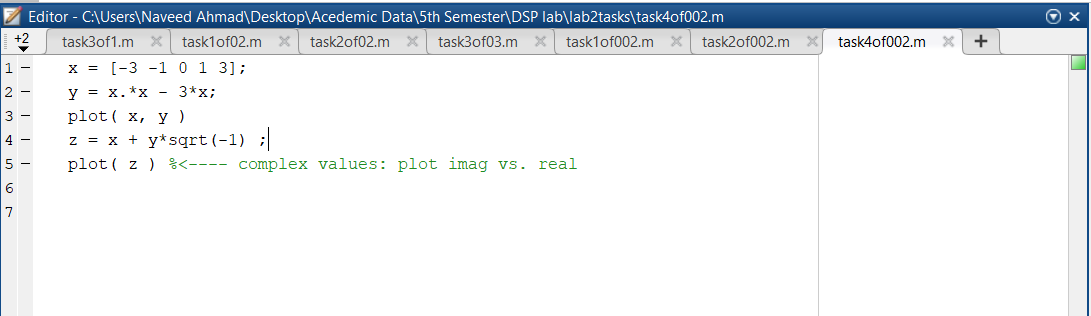
yy(k+6) = cos( k\*pi/3 ) end yy





*The code initializes an empty vector yy and uses a loop to calculate the cosine of angles from −5-5−5 to 555 (in multiples of π/3\pi/3π/3). The results are stored in yy, with an offset to align the index, starting from 111 to 111111 (where k+6k + 6k+6 adjusts the index).*

1. Plotting is easy in MATLAB for both real and complex numbers. The basic plot command will plot a vector y versus a vector x connecting successive points by straight lines. Try the following:



A screenshot of a computer

Description automatically generated

1. **Quadratic Calculation**:

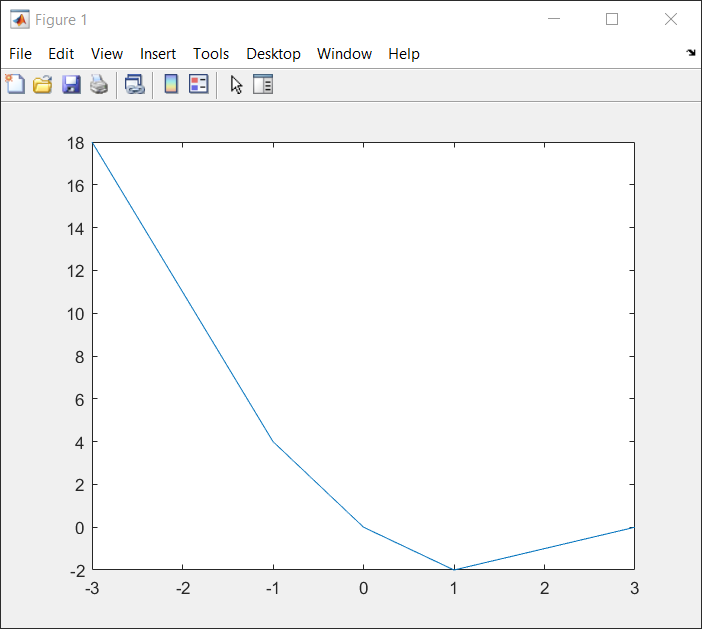
*The code defines a vector x and calculates the values of yyy using the quadratic expression y=x2−3xy = x^2 - 3xy=x2−3x.*

1. **First Plot**:

*The first plot visualizes the relationship between x and y, displaying a parabolic curve.*

1. **Complex Numbers**:

*The code then creates a complex vector z by combining x with y (multiplied by −1\sqrt{1}−1), and the second plot visualizes these complex values.*



(d) Use the built-in MATLAB editor to create a script file called mylab1.m containing the following lines:

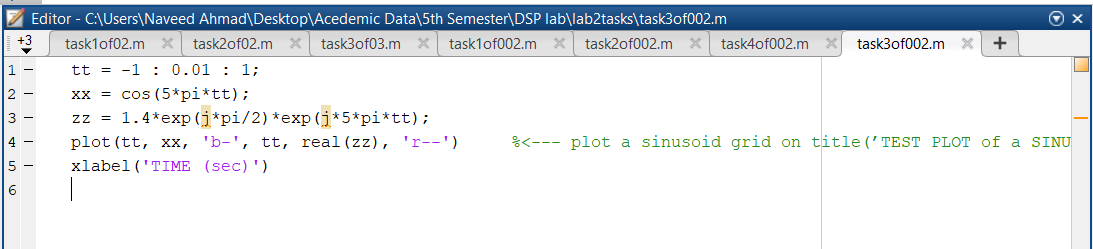
tt = -1 : 0.01 : 1;

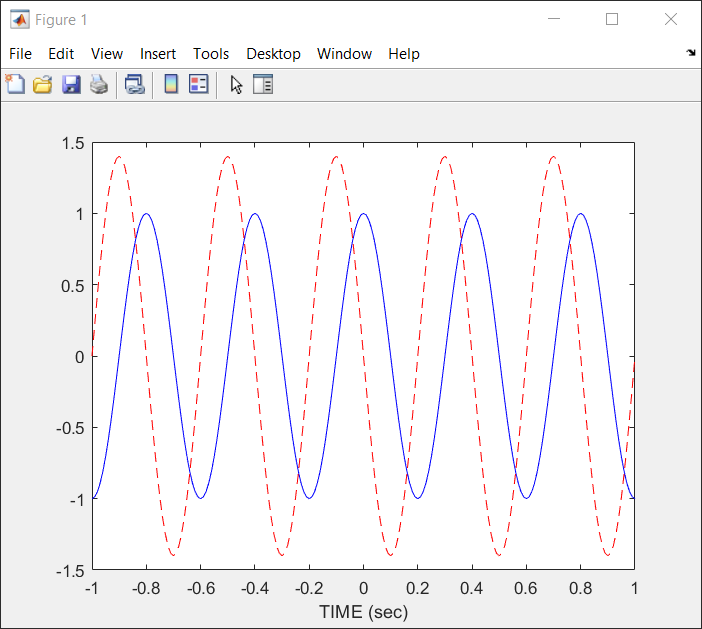
xx = cos(5\*pi\*tt);

zz = 1.4\*exp(j\*pi/2)\*exp(j\*5\*pi\*tt);

plot(tt, xx, 'b-', tt, real(zz), 'r--') %<--- plot a sinusoid grid on title(’TEST PLOT of a SINUSOID’)

xlabel('TIME (sec)')



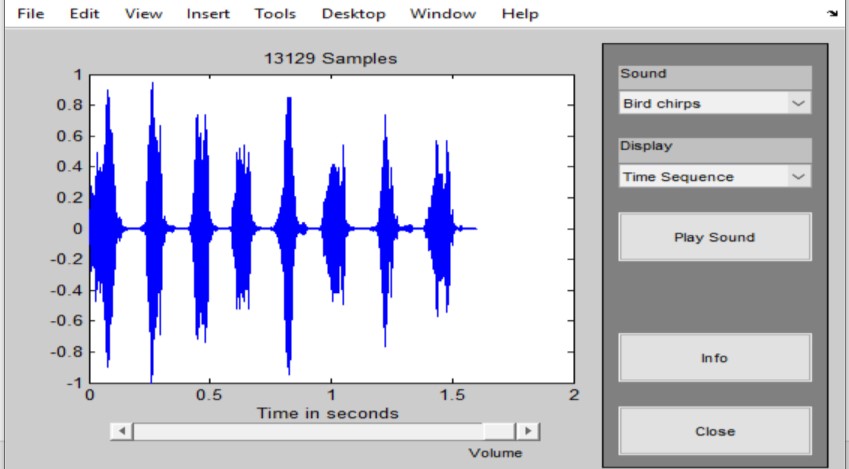


* **Initialization of Time Vector**:
* Discuss how the time vector tt is defined for signal representation, setting the range and resolution.
* **Cosine Wave Generation**:
* Explain the generation of the cosine wave using cos(5\*pi\*tt) and its significance in signal analysis.
* **Complex Exponential Signal**:
* Describe how the complex signal zz is created and the role of exponential functions in representing oscillations.
* **Plotting Techniques**:
* Highlight the plotting methods used, focusing on line styles and colors to distinguish between the cosine wave and its complex representation.
* **Graphical Enhancements**:
* Discuss the importance of grid lines and titles for enhancing plot clarity and understanding in MATLAB visualizations.

## 2.3 MATLAB Sound

The exercises in this section involve sound signals, so you should bring headphones to the lab for listening.

1. Run the MATLAB sound demo by typing xpsound at the MATLAB prompt. If you are unable to hear the sounds in the MATLAB demo then ask for help.

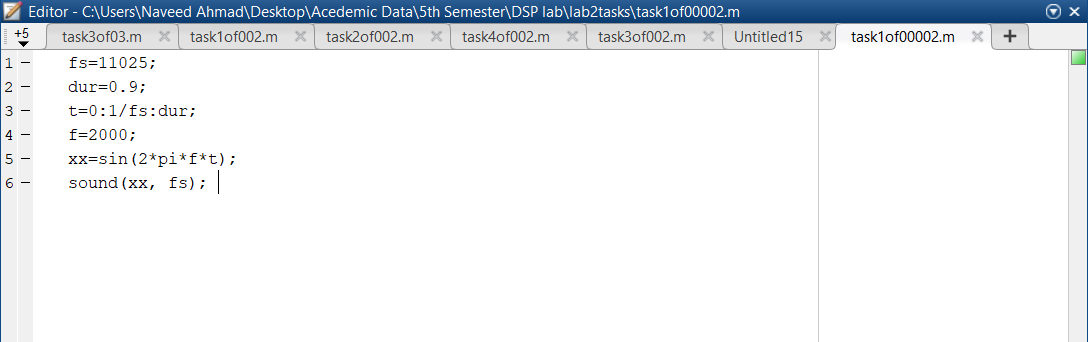


(b) Now generate a tone (i.e., a sinusoid) in MATLAB and listen to it with the soundsc() command.

The first two lines of code in part 2.2(d) create a vector xx of values of a 2.5 Hz sinusoid. The frequency of your sinusoidal tone should be 2000 Hz and its duration should be 0.9 sec. Use a sampling rate (fs) equal to 11025 samples/sec. The sampling rate dictates the time interval between time points, so the time-vector should be defined as follows:

* + 1. tt = 0:(1/fs):dur;

where fs is the desired sampling rate and dur is the desired duration (in seconds). Read the online help for both sound() and soundsc() to get more information on using this command. What is the length (number of samples) of your tt vector?



* **Sampling Frequency Definition**:
  + The code sets the sampling frequency fs to 11025 Hz, which is a common rate for audio processing, ensuring that the sound produced captures the desired frequency range accurately.
* **Duration and Time Vector Creation**:
  + The variable duration is set to 0.9 seconds, and the time vector t is generated from 0 to duration with increments of 1/fs. This vector defines the time intervals for generating the sound wave.
* **Frequency Specification**:
  + The variable f is set to 2000 Hz, which determines the frequency of the sine wave generated. This frequency falls within the audible range for humans.
* **Sine Wave Generation**:
  + The sine wave xx is created using the formula sin(2 \* pi \* f \* t), which generates a continuous oscillating signal based on the specified frequency and time vector.
* **Sound Playback**:
  + The function soundsc(xx, fs) is used to play the generated sine wave through the computer's audio output. The soundsc function scales the audio signal to utilize the full range of sound output, enhancing playback quality.

