Lab Report No 2



Digital Signal Processing

Submitted By: Suleman Shah

Registration No: 21PWCSE1983

Section: C

“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:

Department of Computer Systems Engineering

University of Engineering and Technology Peshawar

**CSE 402L: Digital Signal Processing**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Demonstration of Concepts** | **Poor (Does not meet expectation (1))**  The student failed to demonstrate a clear understanding of the assignment concepts | **Fair (Meet Expectation (2-3))**  The student demonstrated a clear understanding of some of the assignment concepts | **Good (Exceeds Expectation (4-5)**  The student demonstrated a clear understanding of the assignment concepts | **Score**  **30%** |
| **Accuracy** | The student completed ( <50%) tasks and provided MATLAB code and/or Simulink models with errors. Outputs shown are not correct in form of graphs (no labels) and/or tables along with incorrect analysis or remarks. | The student completed partial tasks (50% - <90%) with accurate MATLAB code and/or Simulink models. Correct outputs are shown in form of graphs (without labels) and/or tables along with correct analysis or remarks. | The student completed all required tasks (90%-100%) with accurate MATLAB code and/or Simulink models. Correct outputs are shown in form of labeled graphs and/or tables along with correct analysis or remarks. | **30%** |
| **Following Directions** | The student clearly failed to follow the verbal and written instructions to successfully complete the lab | The student failed to follow the some of the verbal and written instructions to successfully complete all requirements of the lab | The student followed the verbal and written instructions to successfully complete requirements of the lab | **20%** |
| **Time Utilization** | The student failed to complete even part of the lab in the allotted amount of time | The student failed to complete the entire lab in the allotted amount of time | The student completed the lab in its entirety in the allotted amount of time | **20%** |

Lab No: 2.

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

helpwin

helpwin plot

helpwin colon %<--- a VERY IMPORTANT notation

helpwin ops

helpwin zeros

helpwin ones

lookfor filter %<--- keyword search

1. Run the MATLAB demos: type demo and explore a variety of basic MATLAB commands and plots.
2. Use MATLAB as a calculator. Try the following:

pi\*pi - 10

sin(pi/4)

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

1. Do variable name assignment in MATLAB. Try the following:

x = sin( pi/5 );

cos( pi/5 ) %<--- assigned to what?

y = sqrt( 1 - x\*x )

ans

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 ])

# Warm-Up

## 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 ];



1. Extracting 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) ]

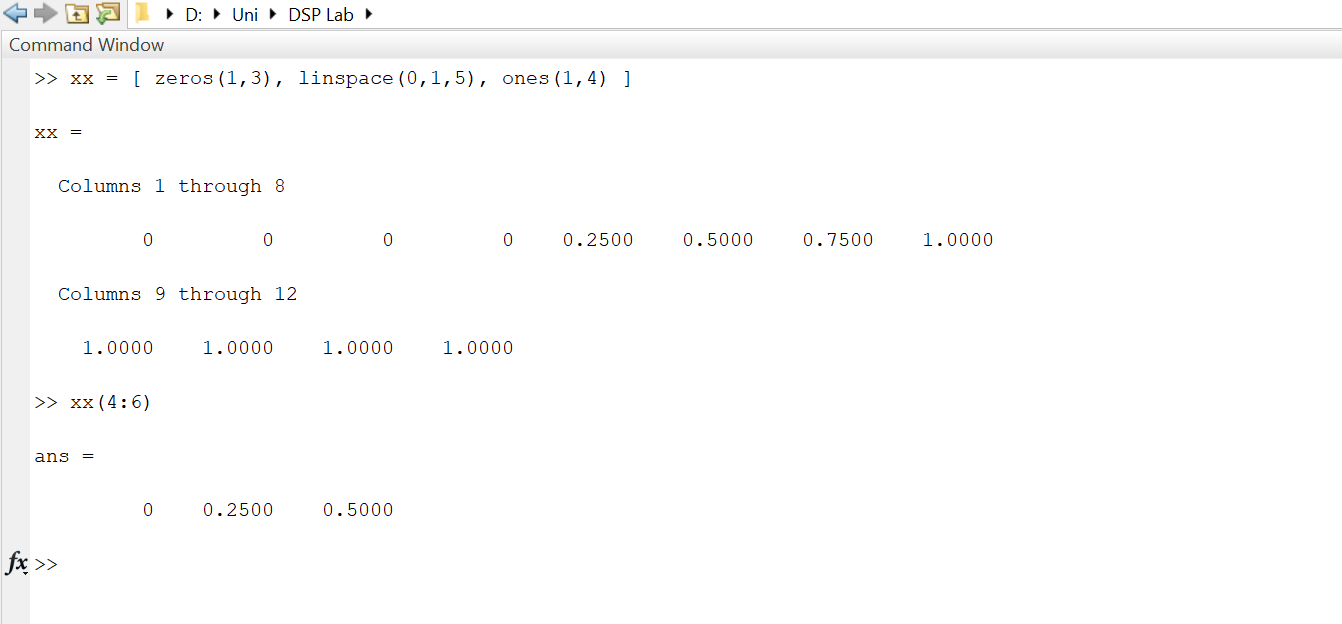
xx(4:6)

size(xx)

length(xx)

xx(2:2:length(xx))

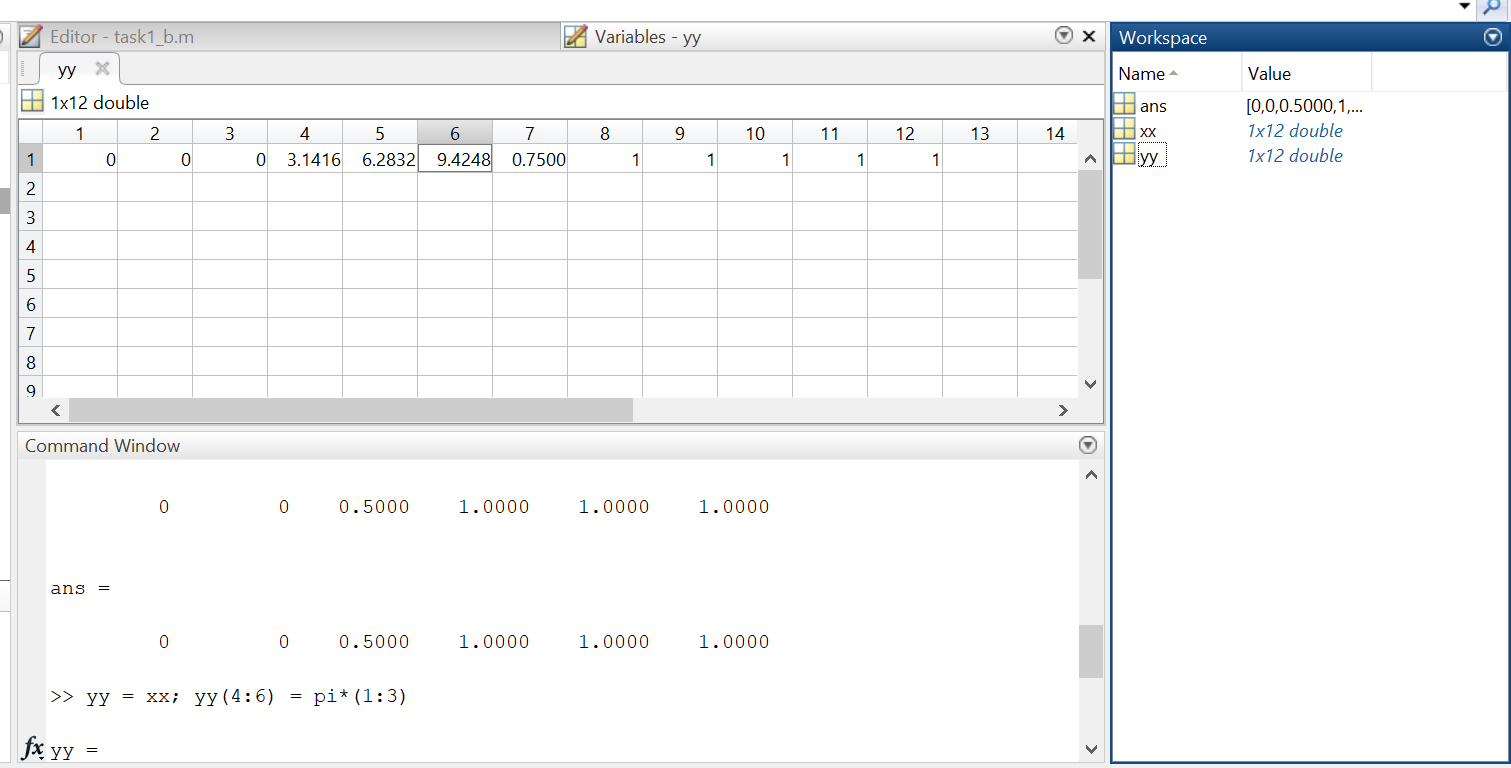
xx(2:2:end)



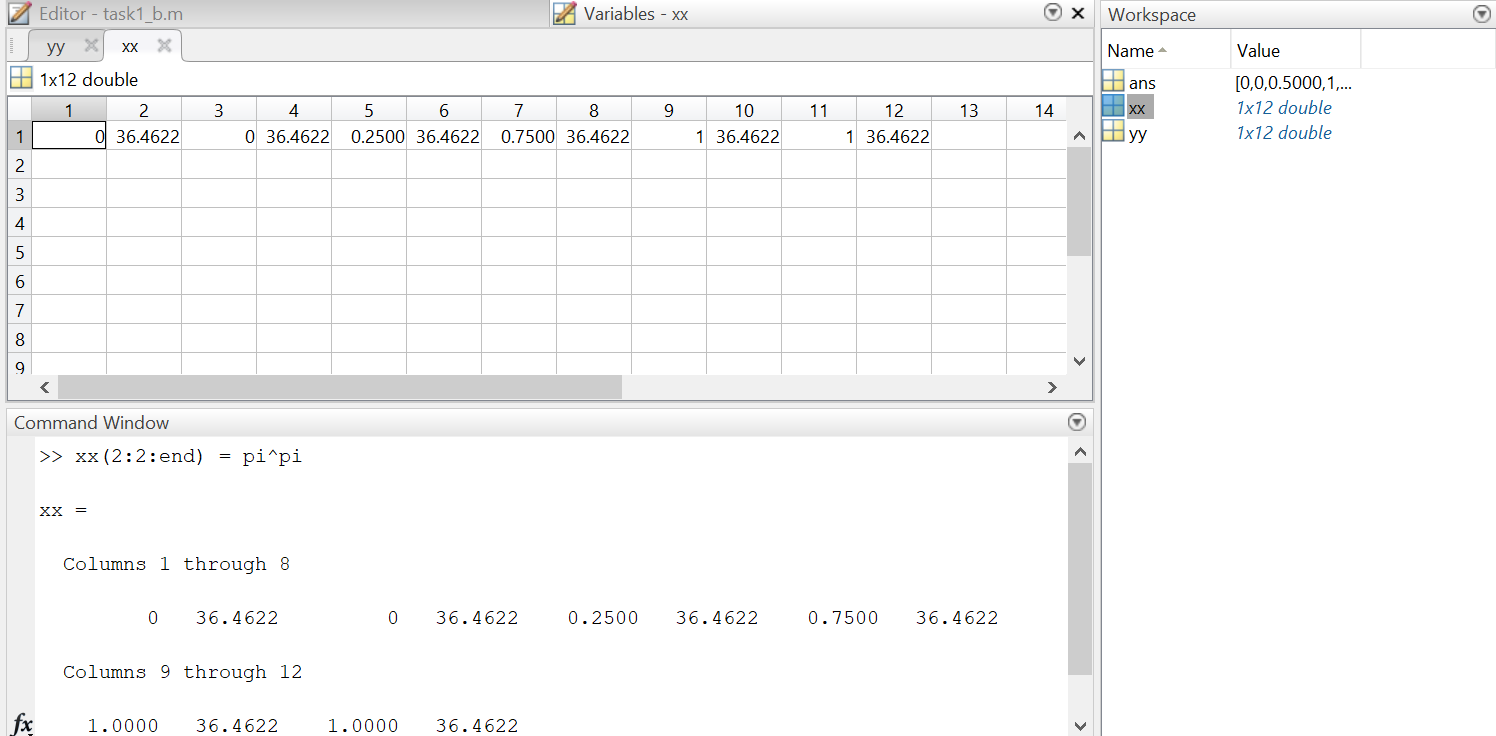
Explain the results echoed from the last four lines of the above code.

1. Observe the result of the following assignments:

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



Now write a statement that will take the vector xx defined in part (b) and replace the even indexed elements (i.e., xx(2), xx(4), etc) with the constant ππ. Use a vector replacement, not a loop.



Section (a) introduces the utilization of colon notation for generating diverse sequences of numbers.

Regarding section (b):

* The size() function is employed to determine the dimensions of an m-by-n matrix. In this context, it was applied to the matrix/vector 'xx' to reveal its size.
* The length() function provides the total number of entries in a matrix.
* The expression xx(2:2:length(xx)) identifies and displays the even entries in the vector 'xx.'
* The notation xx(2:2:end) achieves the same result as the aforementioned code line.

In section (c), the vector 'yy' is created by replicating the entire vector 'xx,' and subsequent modifications involve replacing elements at indices 4, 5, and 6 with 1π, 2π, and 3π, respectively.

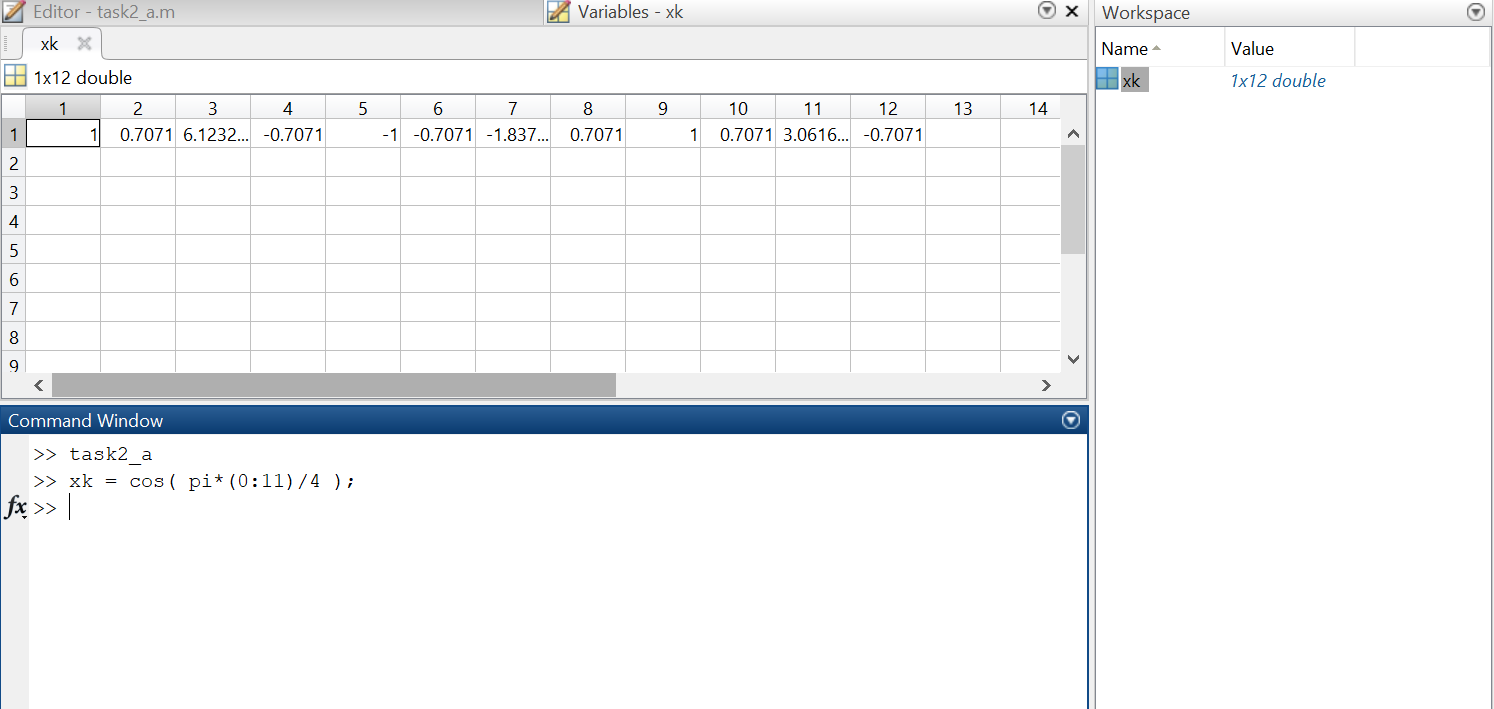
Moving on to section (d), the even elements of vector 'xx' are replaced with the value 'pipi.'

## 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

Explain how the different values of cosine are stored in the vector xk. What is xk(1)? Is xk(0) defined?



NOTES: the semicolon at the end of a statement will suppress the echo to the screen. The text following the % is a comment; it may be omitted.

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

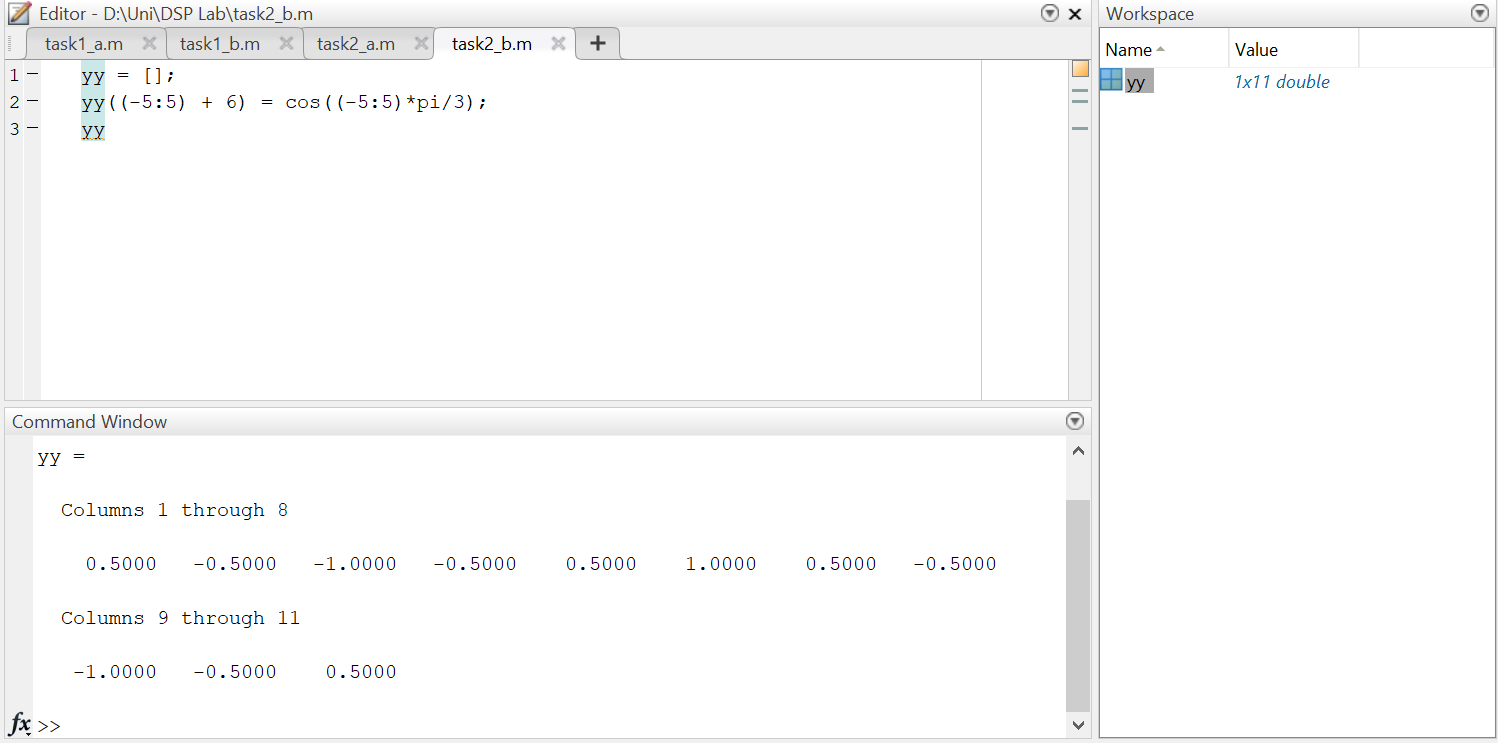
for k=-5:5

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

end

yy

Explain why it is necessary to write yy(k+6). What happens if you use yy(k) instead?



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:

x = [-3 -1 0 1 3];

y = x.\*x - 3\*x;

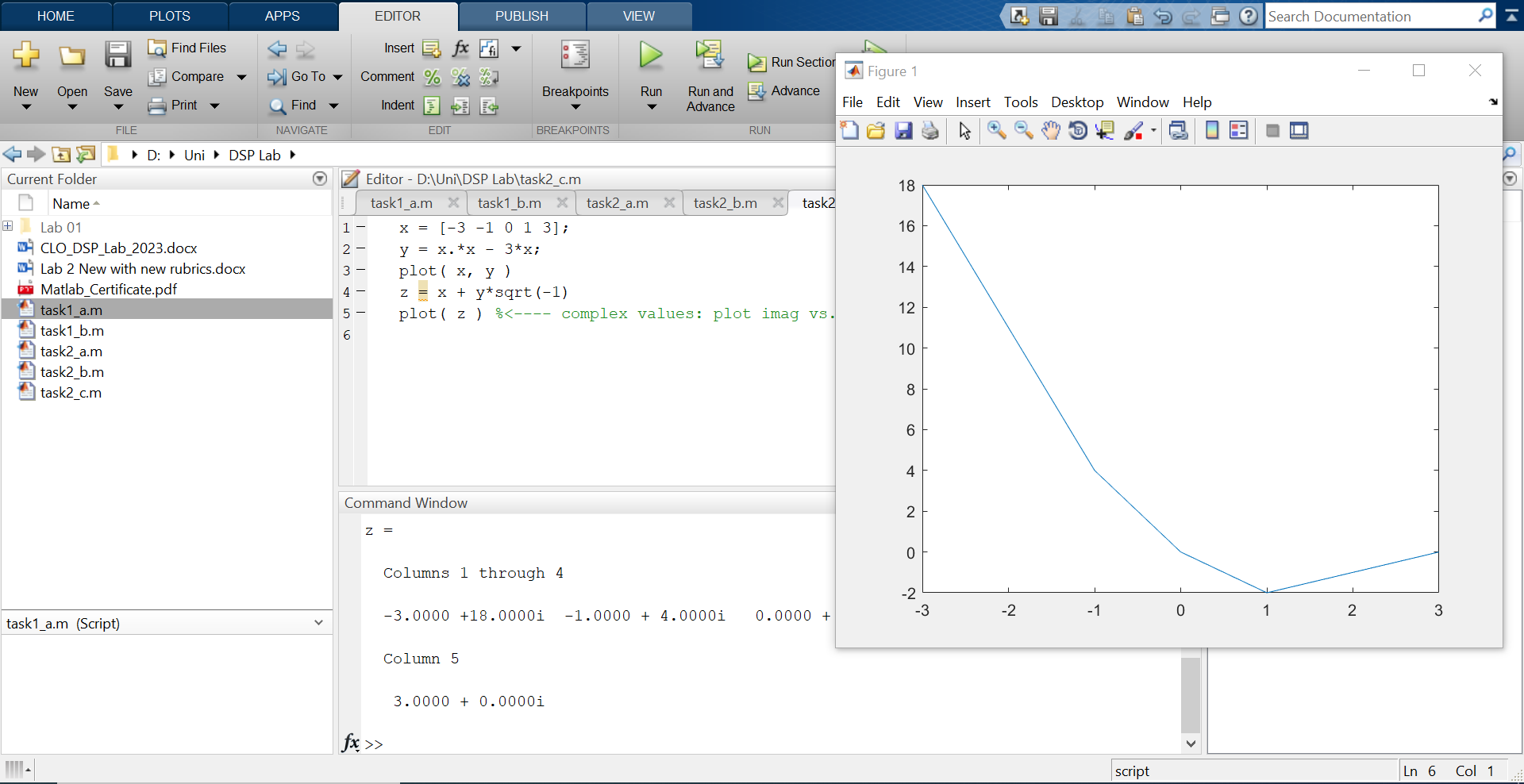
plot( x, y )

z = x + y\*sqrt(-1)

plot( z ) %<---- complex values: plot imag vs. real

Use helpwin arith to learn how the operation xx.\*xx works when xx is a vector; compare to matrix multiply.

When unsure about a command, use helpwin.



1. 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)’)

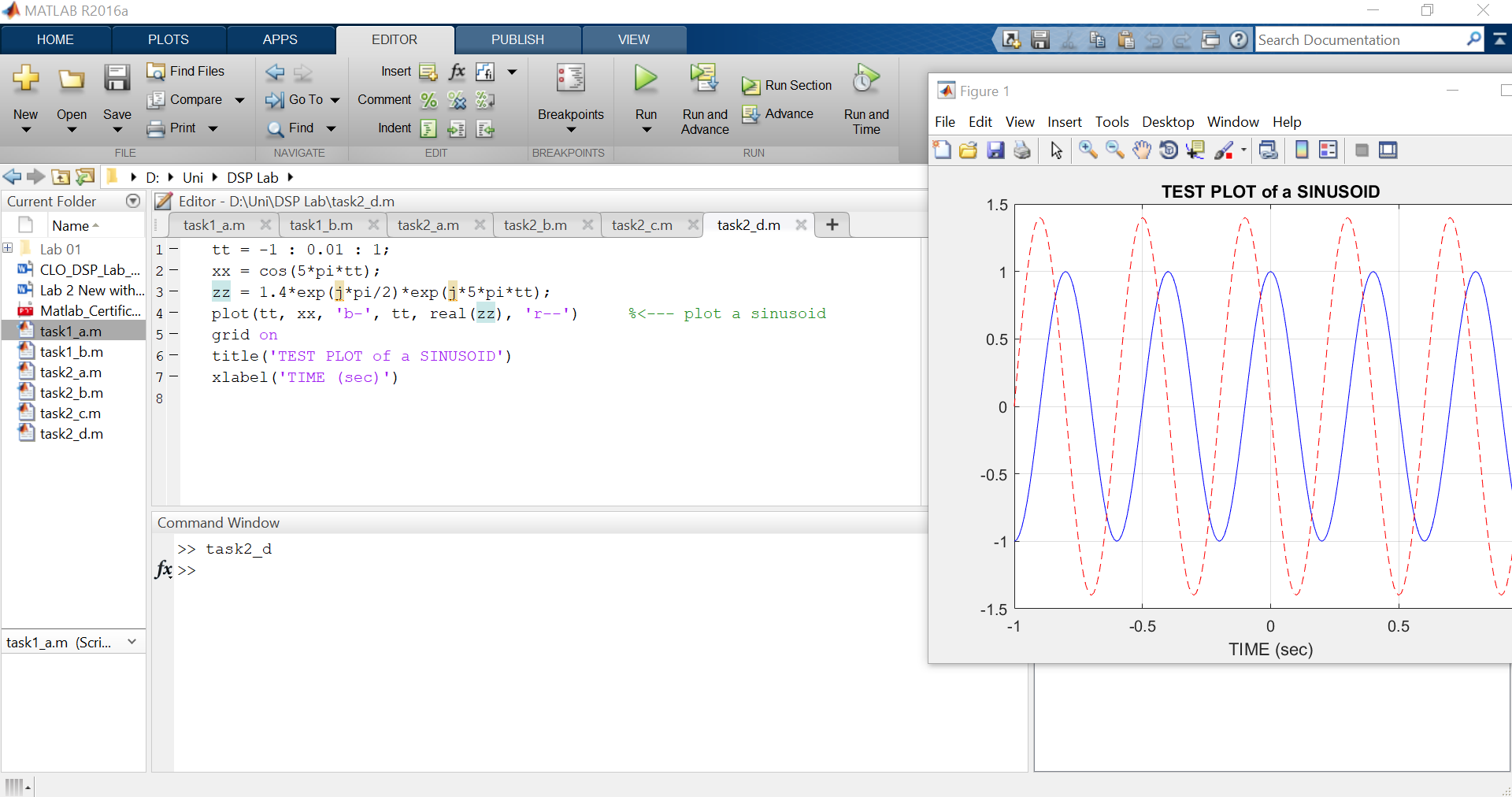
Explain why the plot of real(zz) is a sinusoid. What is its phase and amplitude? Make a calculation of the phase from a time-shift measured on the plot.

Section (a) explores the generation of various cosine values through colon notation within the cos function. It notes that xk(0) is not defined in MATLAB when referencing the index of vector zk. The mathematical representation xk(0) corresponds to 0 at index 1, xk(1).

In part (b), the discussion revolves around MATLAB generating an error for negative indices, necessitating the initiation of the index at 1 by adding 6 to k.

Moving on to part (c), the narrative describes the creation of two vectors, x and y. Subsequently, an imaginary vector z is formed by combining the real part x and the imaginary part y. Finally, a plot of the vector z is generated.

In part(d): Amplitude is 1.4 and Phase is pi/2 or 90 degrees.

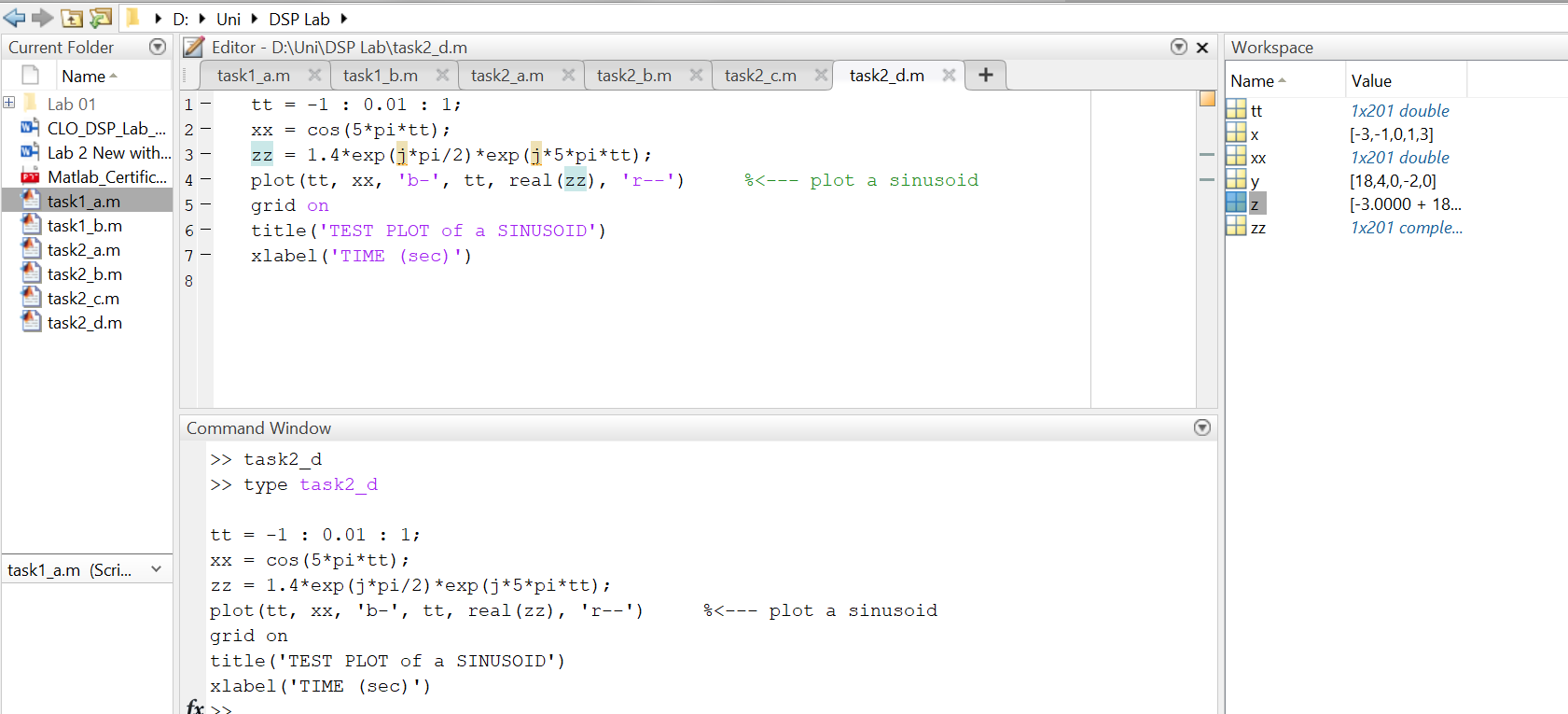


1. Run your script from MATLAB. To run the file mylab1 that you created previously, try

mylab1 %<---will run the commands in the file

type mylab1 %<---will type out the contents of

% mylab1.m to the screen



## 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.

When unsure about a command, use helpwin.

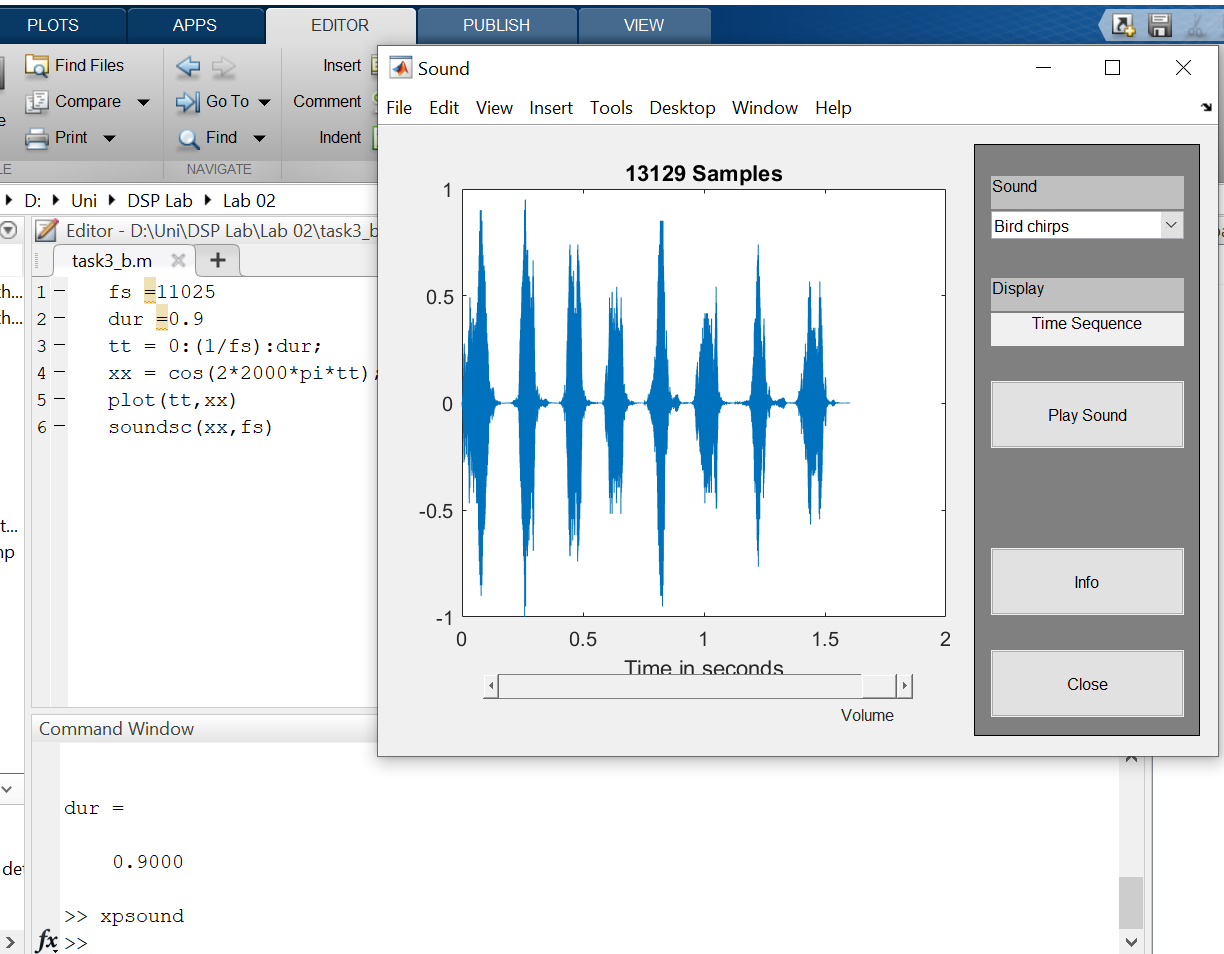
1. Now generate a tone (i.e., a sinusoid) in MATLAB and listen to it with the soundsc() command.[[1]](#footnote-1)

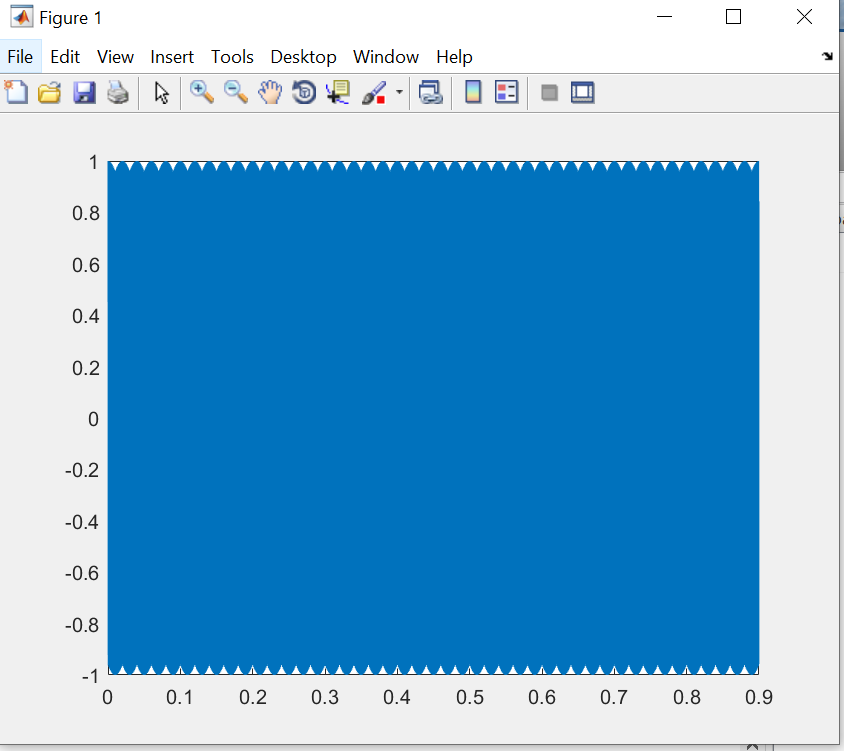
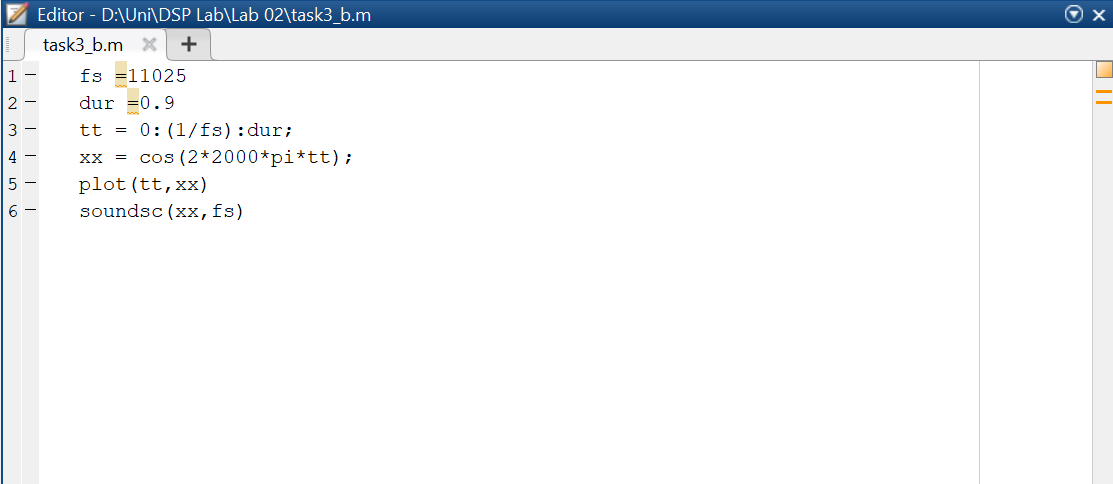
The first two lines of code in part 3.2(d) create a vector xx of values of a 2.5 Hz sinusoid. The frequency of your sinusoidal tone should be 2000 Hz and its duration should be 0.9 sec. Use a sampling rate (fs) equal to 11025 samples/sec. The sampling rate dictates the time interval between time points, so the time-vector should be defined as follows:

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?

Part a Output



Part b Output

1. The soundsc(xx,fs) function requires two arguments: the first one (xx) contains the vector of data to be played, the second argument (fs) is the sampling rate for playing the samples. In addition, soundsc(xx,fs) does automatic scaling and then calls sound(xx,fs) to actually play the signal. [↑](#footnote-ref-1)