Lab Report No 2

Digital Signal Processing

Submitted By:

Registration No:

Section:

"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))	Fair (Meet Expectation (2-3))	Good (Exceeds Expectation (4-5)	Score
	The student failed to demonstrate a clear understanding of the assignment concepts	The student demonstrated a clear understanding of some of the assignment concepts	The student demonstrated a clear understanding of the assignment concepts	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.

1.1 Playing with MATLAB

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

(a) 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.

(b) 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
```

(c) Run the MATLAB demos: type `demo` and explore a variety of basic MATLAB commands and plots.

(d) Use MATLAB as a calculator. Try the following:

```
pi*pi - 10
sin(pi/4)
ans ˆ 2           %<--- "ans" holds the last result
```

(e) 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
```

(f) 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 ])
```

2 Warm-Up

2.1 MATLAB Array Indexing

(a) 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 ];
```

D: ▸ Uni ▸ DSP Lab ▸

Command Window

```
>> jkl = 0 : 6

jkl =

     0     1     2     3     4     5     6

>> jkl = 2:4:17

jkl =

     2     6    10    14

>> jkl = 99 : -1 : 88

jkl =

    99    98    97    96    95    94    93    92    91    90    89    88

>> ttt = 2 : (1/9) : 4

ttt =

  Columns 1 through 8

    2.0000    2.1111    2.2222    2.3333    2.4444    2.5556    2.6667    2.7778
```

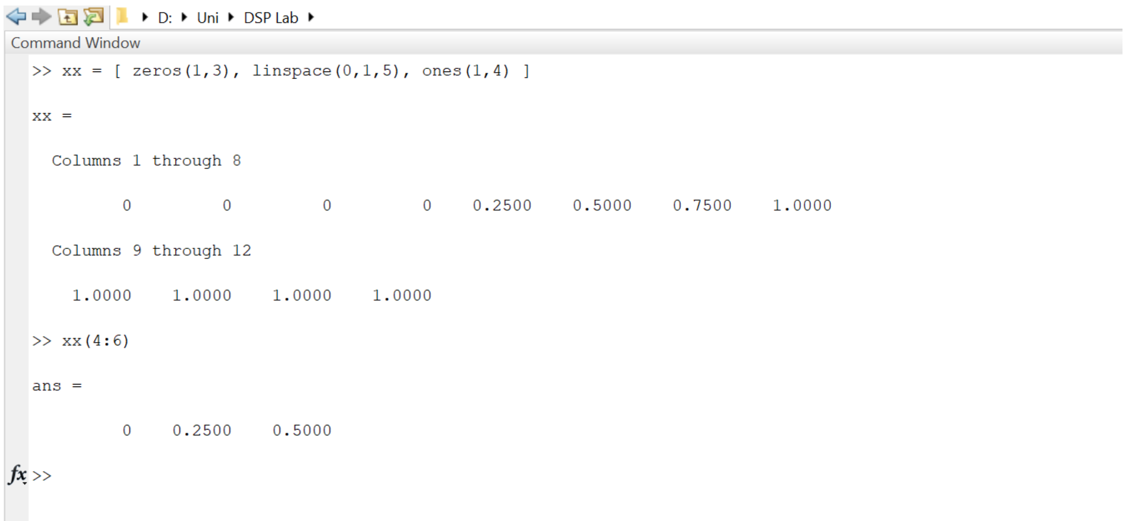
Command Window

```
    2.8889    3.0000    3.1111    3.2222    3.3333    3.4444    3.5556    3.6667

  Columns 17 through 19

    3.7778    3.8889    4.0000

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

tpi =

  Columns 1 through 8

         0    0.3142    0.6283    0.9425    1.2566    1.5708    1.8850    2.1991

  Columns 9 through 16

    2.5133    2.8274    3.1416    3.4558    3.7699    4.0841    4.3982    4.7124

  Columns 17 through 21

    5.0265    5.3407    5.6549    5.9690    6.2832

fx >>
```

Remarks: In this section, I learned how to use colon notation to generate any sequence of numbers.

(b) 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)
```



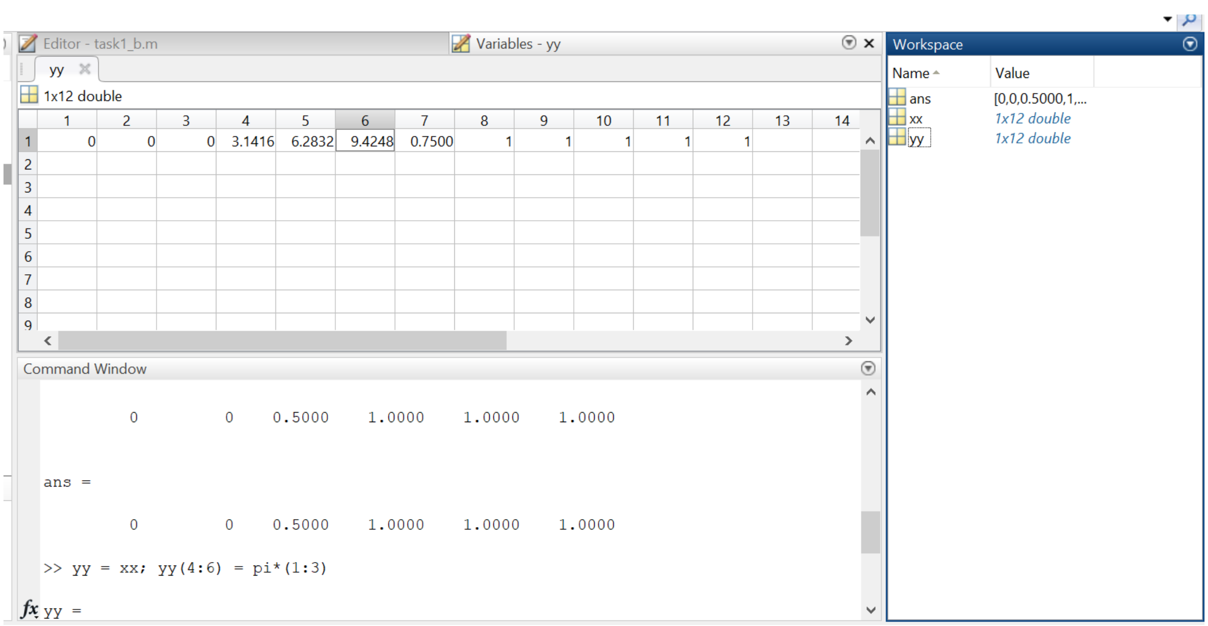
Explain the results echoed from the last four lines of the above code.

Remarks:

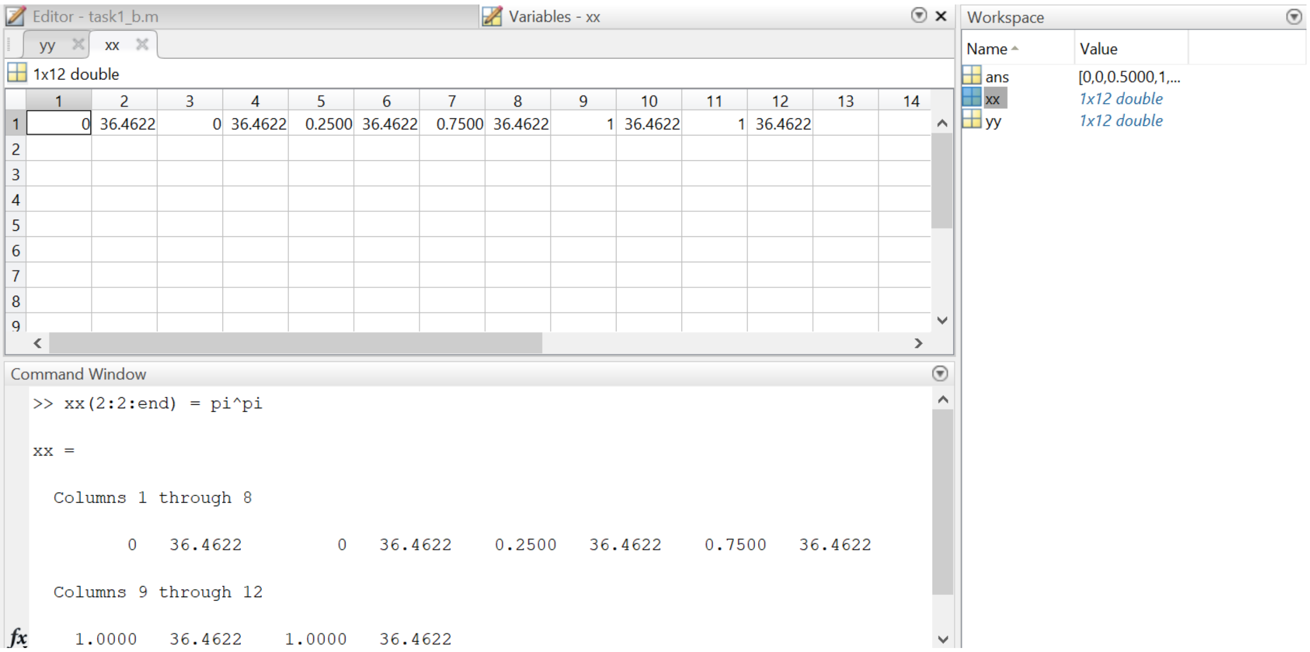
- size() function returns the size of an m-by-n matrix. So in this case, we passed xx into size as argument and it shows us the size of matrix(vector in this case).
- length() shows us the total entries in a matrix.
- xx(2:2:length(xx)) will shows us the even entries in vector xx
- xx(2:2:end) works the same as above code line.

(c) 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 π^{π}. Use a vector replacement, not a loop.



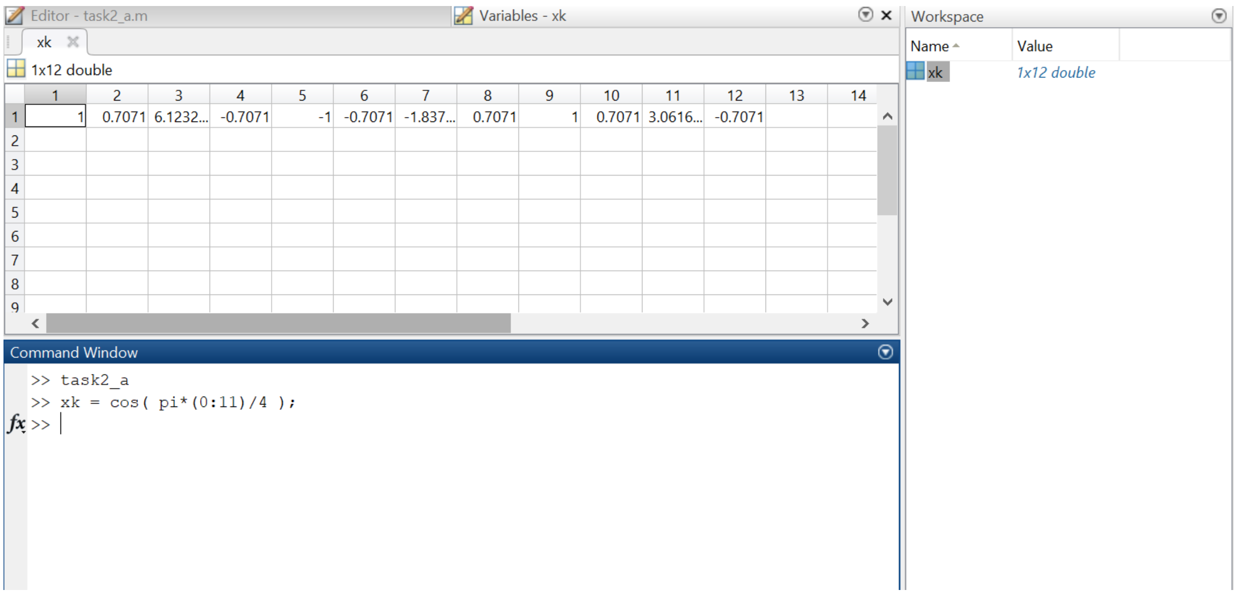
Remarks: In part(c),first I replicated the whole vector xx and assigned it to yy. Then I changed the elements at index 4,5,6 with 1 pi, 2 pi and 3 pi respectively. After that, in part(d), I changed the even elements of vector xx with pi$^{\mathrm{pi}}$.

2.2 MATLAB Script Files

(a) 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.

Remarks: Different values of cos are generated using the colon notation in the cos function. xk(0) is not defined in MATLAB when we are referring to the index of vector zk. Mathematically, xk(0) is defined for 0 which is at index 1, xk(1).

```
>> xk(0)
Subscript indices must either be real positive integers or logicals.

>>
```

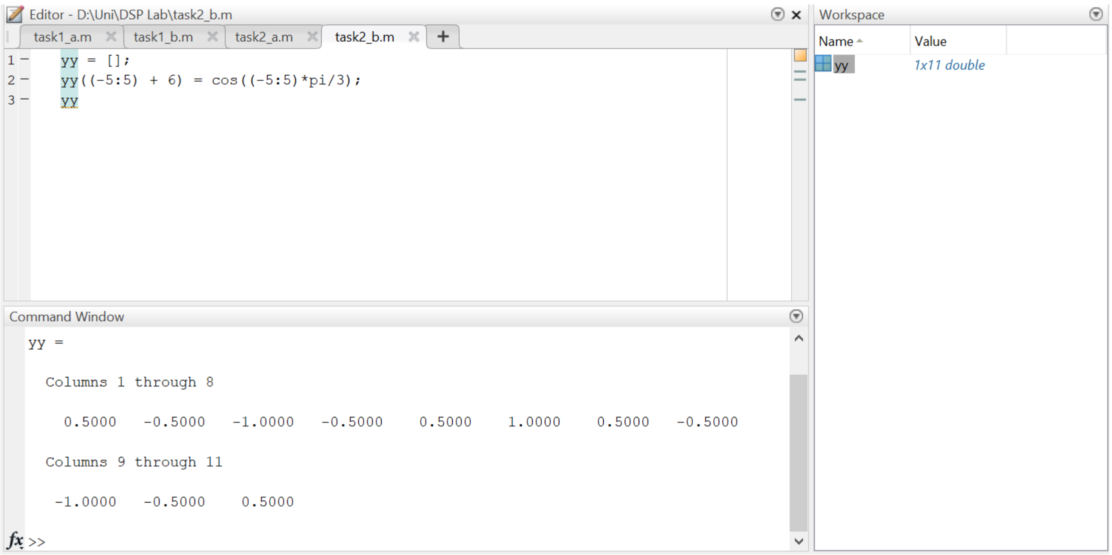
(b) (A taste of vectorization) Loops can be written in MATLAB, but they are NOT the most efficient way to get things done. It's better to always avoid loops and use the colon notation instead. The following code has a loop that computes values of the cosine function. (The index of `yy()` must start at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

Remarks: Because MATLAB generates an error for negative index. Hence, we start at index 1 by adding 6 to k.

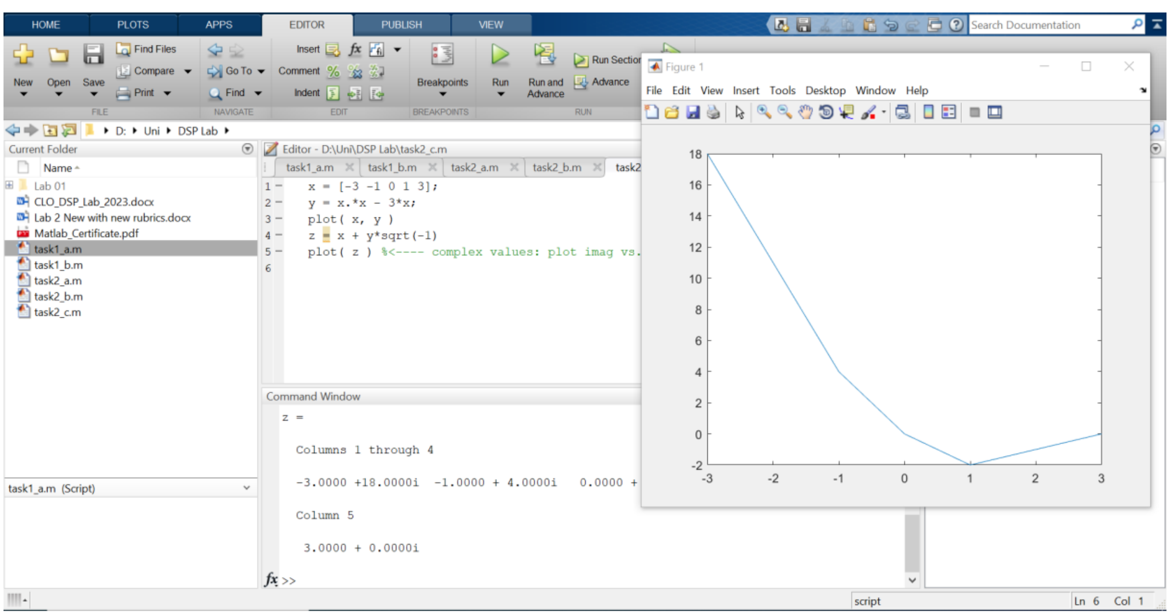


(c) Plotting is easy in MATLAB for both real and complex numbers. The basic plot command will plot a vector `y` versus a vector `x` connecting successive points by straight lines. Try the following:

```
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`.



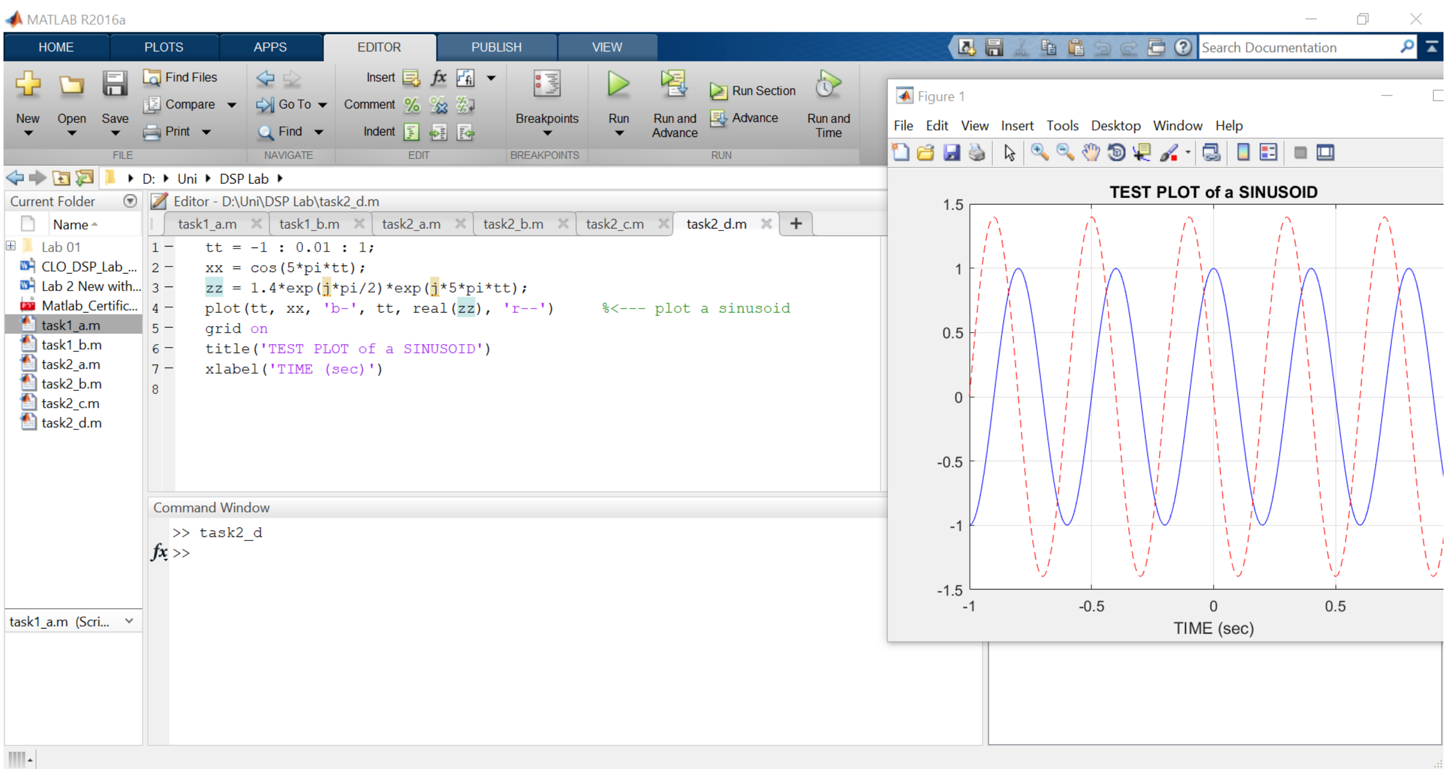
Remarks: In this section, I created two vectors x and y. Then I created an imaginary vector z using the real part x and imaginary part y. Lastly, I plot the vector z.

(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)')
```

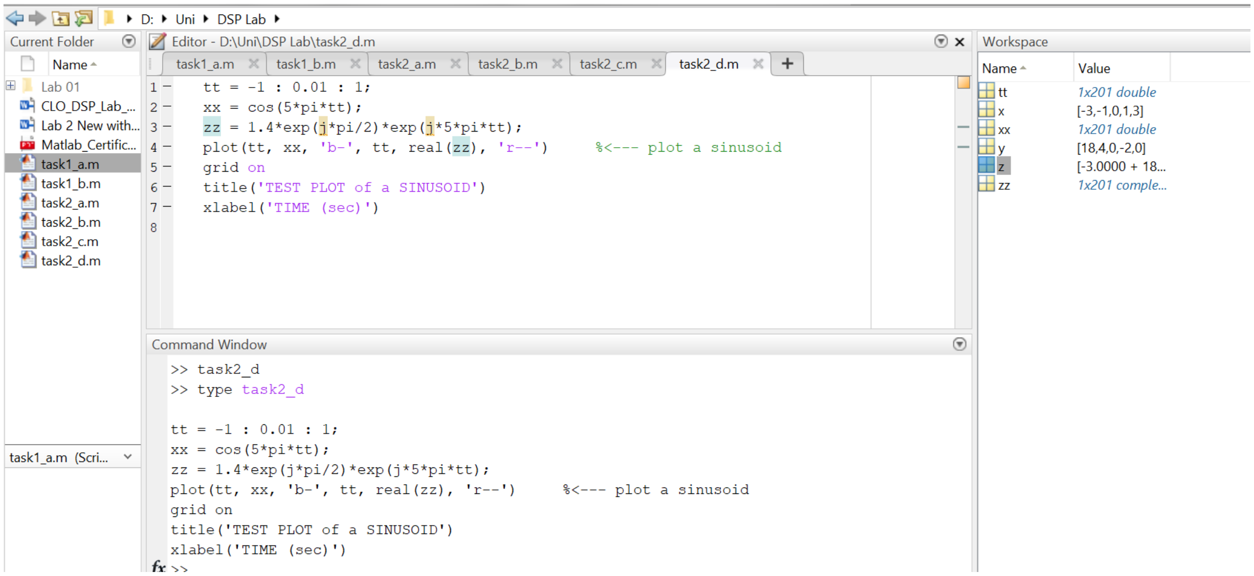
Explain why the plot of `real(zz)` is a sinusoid. What is its phase and amplitude? Make a calculation of the phase from a time-shift measured on the plot.

Remarks: Amplitude is 1.4 and Phase is pi/2 or 90 degrees.



(e) 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
```



2.3 MATLAB Sound

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

(a) 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`.

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

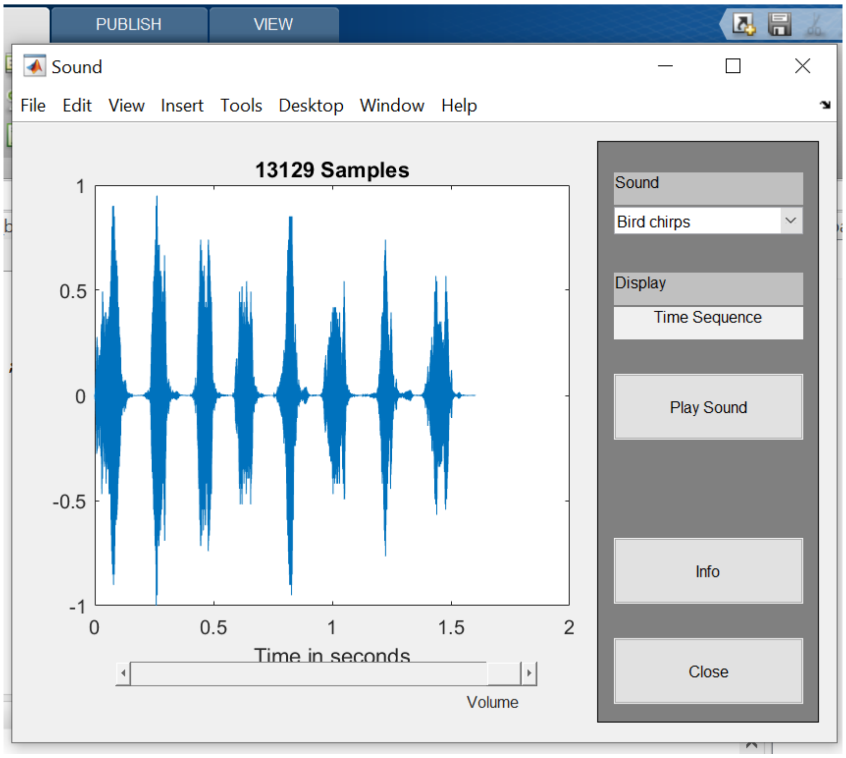
The first two lines of code in part 2.2(d) create a vector xx of values of a 2.5 Hz sinusoid. The frequency of your sinusoidal tone should be 2000 Hz and its duration should be 0.9 sec. Use a sampling rate (`fs`) equal to 11025 samples/sec. The sampling rate dictates the time interval between time points,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

```
tt = 0:(1/fs):dur;
```

[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.

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)



Part(b)