1. Lab 0 - Part 1

Table of Contents

1.1 - Variables

1. Execute and explain the following commands (pay attention to what variable ans points to):

This MATLAB command calculates 6^2, finding the result of 36 and storing it into the ans variable.

```
6^2
```

```
ans =

    36
```

This MATLAB command prints the value of the ans variable to the console.

```
ans
```

```
ans =

    36
```

This MATLAB command takes the value of the ans variable, divides it by 6, stores it back into the ans variable, and prints it to the console

```
ans/6
```

```
ans =

     6
```

This MATLAB command prints the value of the ans variable to the console (here, ans is thetresult of the last computation performed).

```
ans
```

```
ans =

     6
```

2. MATLAB can be used as a very expensive calculator. Execute and explain the following commands:

This MATLAB equation makes use of the pi constant to evaluate pi^2 -10. It then stores the result into ans. I will omit this part about storing the result into ans heirinafter.

```
pi*pi-10
```

```
ans =

   -0.1304
```

This calculates the sine of pi/4, ising the sine trigonometric function included with MATLAB and the pi constant.

```
sin(pi/4)
```

```
ans =

    0.7071
```

This calcuales the square of ans, where ans is the sin(pi/4).

```
ans^2
```

```
ans =

    0.5000
```

3. Just like in other programming languages, you can assign variables in MATLAB. Execute and explain the following commands and watch the area labeled ?workspace? on the side of your screen. The variables will appear as they are assigned.

This calculates the sin(pi/5) and assigns it to the variable *x* for later use.

```
x = sin(pi/5)
```

```
x =

    0.5878
```

This calculates the cos(pi/5).

```
cos(pi/5)
```

```
ans =

    0.8090
```

This calculates the sqrt(1-x^2) and assigns it to *y*, where *x* was assigned to be the result of sin(pi/5) in a previous step.

```
y = sqrt(1-x*x);
```

This prints ans to the console, where ans was assigned to be cos(pi/5) in a previous step. Note that it was not the result of sqrt(1-x^2) in the previous step, as that result was stored in *y*.

```
ans
```

```
ans =

    0.8090
```

Question 1: What is the numerical value of *x* and *y* ?

We can see from the below that *x* = 0.5878 and *y* = 0.8090

```
x
y
```

```
x =

    0.5878


y =

    0.8090
```

1.2 - Vectors and Colon Operator

1. Execute and explain the following commands:

This command creates a series of values from 0 to 6, inclusive, with a step size of 1 and assigns it to *x*.

```
x = 0:6;
```

This command creates a series of values bounded by 2 and 17 with a step size of 4 and assigns it to *y*.

```
y = 2:4:17;
```

This command creates a series of values bounded by 2 and 4 with a step size of 1/9 (~.1111) and assigns it to *z*.

```
z = 2:(1/9):4;
```

This command first creates a series of values bounded by 0 and 2 with a step size of .1, then multiplies pi into each term, and finally assigns it to *t*.

```
t = pi*[0:0.1:2];
```

2. With this colon notation in MATLAB, extracting/inserting numbers into a vector is made easy. Execute and explain the following commands:

zeros(1,3) creates a vector of length 3 with all zeroes. linspace(0,1,5) creates a list of length 5 of evenly spaced elements that range from 0 -> 1 ones(1,4) creates a vector of length 4 with all ones. The square bracketed comma delimited list concatenates the three vectors together.

```
xx = [zeros(1,3), linspace(0,1,5), ones(1,4)];
```

This command returns the 4th through the 6th elements, inclusive, from xx.

```
xx(4:6)
```

```
ans =

         0    0.2500    0.5000
```

This command returns the size of the rows and columns in this matrix. Here, the amount of rows is 1 and the amount of columns is 12.

```
size(xx)
```

```
ans =

     1    12
```

This command returns the amount of columsn in this matrix. As discussed above, that result is 12.

```
length(xx)
```

```
ans =

    12
```

This commmad slices into the xx vector starting at the second element and continuing until the end, and uses a step size of 2.

```
xx(2:2:length(xx))
```

```
ans =

         0         0    0.5000    1.0000    1.0000    1.0000
```

Execute and explain the following commands:

This command assigns xx to yy

```
yy = xx;
```

This command assigns yy from 4 to 6 inclusive to be pi multiplied into the vector pi*[1,2,3]

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

3. Now, after learning how the above code works, Write one line of code (using vectorization) that will take the vector xx and take the elements with an even index {xx(2), xx(4), ...} and replace them with pi^pi (thats pi to the exponent pi).

```
xx(2:2:length(xx)) = pi^pi;
```

1.3 - Loops

1. Read help for page. Execute and explain the following commands:

The command x=x+1 iterates 4 times, as k=1 the first iteration, k=2 the second interation, and so forth until k=4 the final iteration. The loop is bounded from 1 -> n so x = n at the commencement of the loop.

```
x = 0;
for k = 1:4
    x = x+1;
end
```

2. Using for loop, generate a vector containing the following elements: 1^1, 2^2, 3^3, ..., 9^9. Write code and explain.

This code first initializes a vector of length 9 to be all 0. Then, we have a for loop iterate *i* from 1->9 that sets the ith element to be i^i.

```
exponentiated = zeros(1,9);
for i = 1:9
    exponentiated(i) = i^i;
end
exponentiated
```

```
exponentiated =

  Columns 1 through 6

           1           4          27         256        3125
 46656

  Columns 7 through 9

      823543    16777216   387420489
```

3. In many cases, code with loops can be optimized by vectorization. Functions like exp() and cos() are defined for vector inputs. Example:

```
M = 200;
for k = 1:M
    x(k) = k;
    y(k) = cos(0.001*pi*x(k)*x(k));
end
```

The piece of code above can be optimized as:

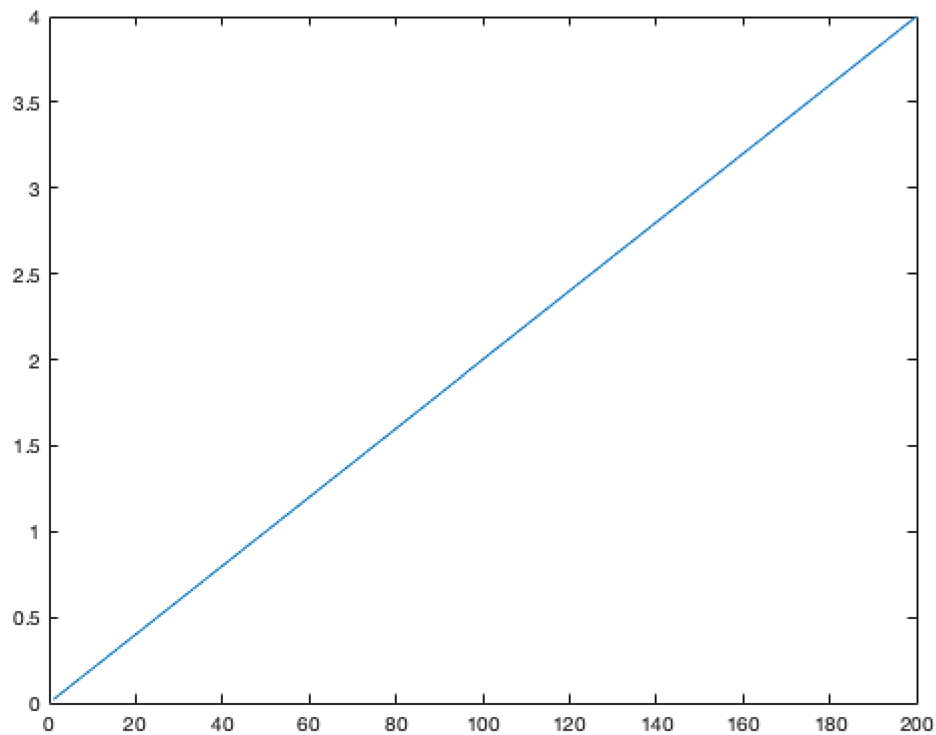
```
M = 200;
y = cos(0.001*pi*(1:M).*(1:M));
```

Now use this same idea to optimize the following code by removing the for loop. Write 2 or 3 lines of code, plot, and explain:

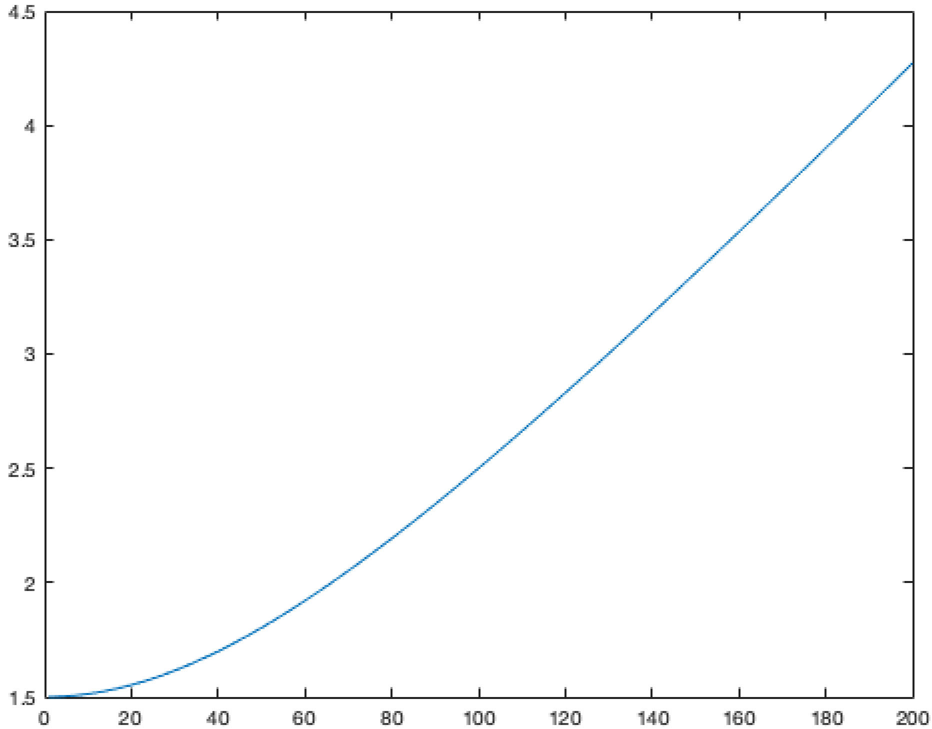
```
N = 200;
for k = 1:N
    xk(k) = k/50;
    rk(k) = sqrt(xk(k)*xk(k) + 2.25);
    sig(k) = exp(j*2*pi*rk(k));
end
```

As each iteration of the previous for loop was not dependent on the other iterations of the for loop, it was quite easy to parallelize these operations with the use of matrices. Simply create a vector ranging form 1->200 and divide it by 50 for xk. For rk, perform element wise multiplication on rk*rk and then add 2.25 to each eleemtn as well. Finally, take the square root. To calculate sig, just calculate j*2*pi*rk and exponentiate it. This performs element wise operations for each element of sig.

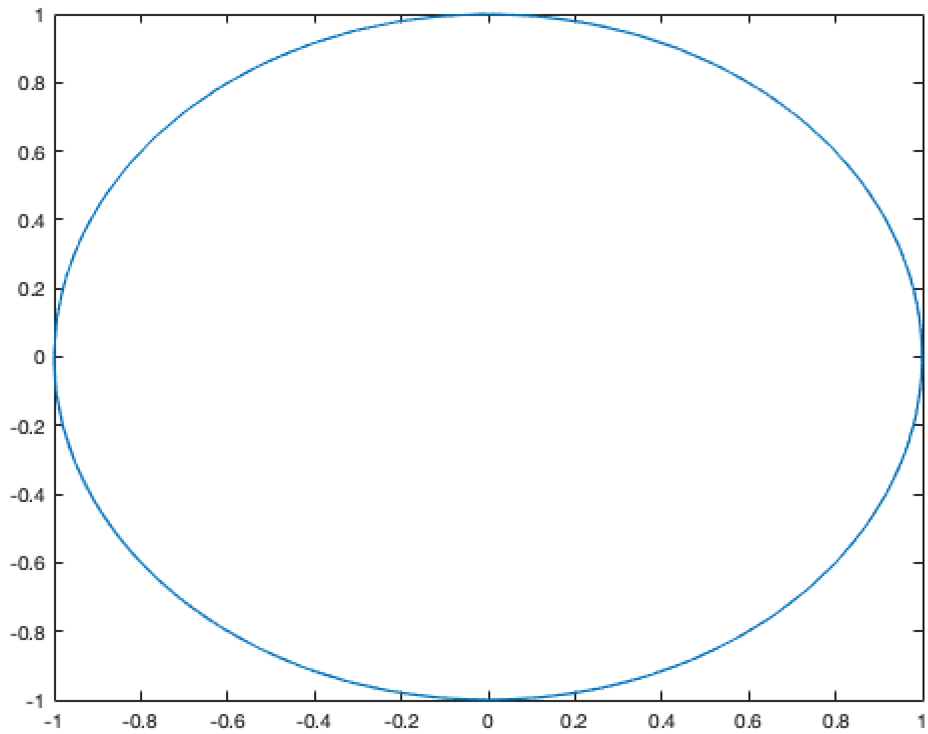
```
N = 200;
xk = [1:200]/50;
rk = sqrt(xk.*xk + 2.25);
sig = exp(j*2*pi*rk);

plot(xk);
```




```
plot(rk);
```




```
plot(sig);
```



Question 2: What?s the value of x after the code executes?

As seen below, the value of x is equal to:

```
x


x =

  Columns 1 through 13

     1      2      3      4      5      6      7      8      9     10     11
 12     13

  Columns 14 through 26

    14     15     16     17     18     19     20     21     22     23     24
 25     26

  Columns 27 through 39

    27     28     29     30     31     32     33     34     35     36     37
 38     39

  Columns 40 through 52
```

```
    40     41     42     43     44     45     46     47     48     49     50
51     52

  Columns 53 through 65

    53     54     55     56     57     58     59     60     61     62     63
64     65

  Columns 66 through 78

    66     67     68     69     70     71     72     73     74     75     76
77     78

  Columns 79 through 91

    79     80     81     82     83     84     85     86     87     88     89
90     91

  Columns 92 through 104

    92     93     94     95     96     97     98     99    100    101    102
103    104

  Columns 105 through 117

   105    106    107    108    109    110    111    112    113    114    115
116    117

  Columns 118 through 130

   118    119    120    121    122    123    124    125    126    127    128
129    130

  Columns 131 through 143

   131    132    133    134    135    136    137    138    139    140    141
142    143

  Columns 144 through 156

   144    145    146    147    148    149    150    151    152    153    154
155    156

  Columns 157 through 169

   157    158    159    160    161    162    163    164    165    166    167
168    169

  Columns 170 through 182

   170    171    172    173    174    175    176    177    178    179    180
181    182

  Columns 183 through 195
```

```
  183   184   185   186   187   188   189   190   191   192   193
194   195

 Columns 196 through 200

  196   197   198   199   200
```

Question 3: What is the purpose of the dot before the asterisk in line two of the above?

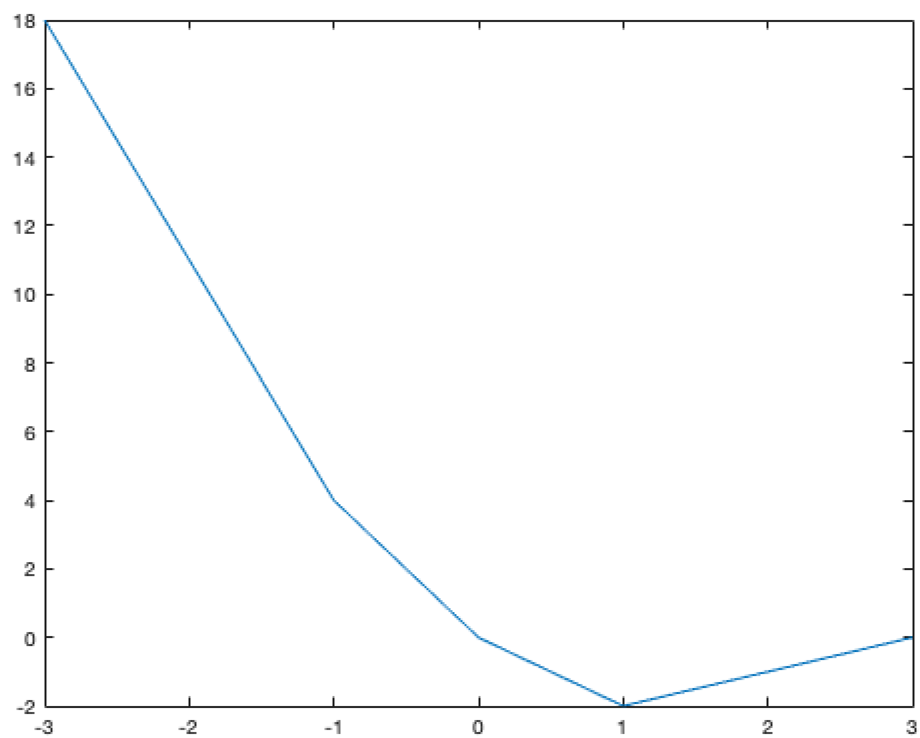
The purpose of the dot before the asterisk in line two of the above means that we want to perform element wise multiplication, NOT matrix multiplication. The dot is commonly used to denote this in many programming contexts.

1.4 - Plot

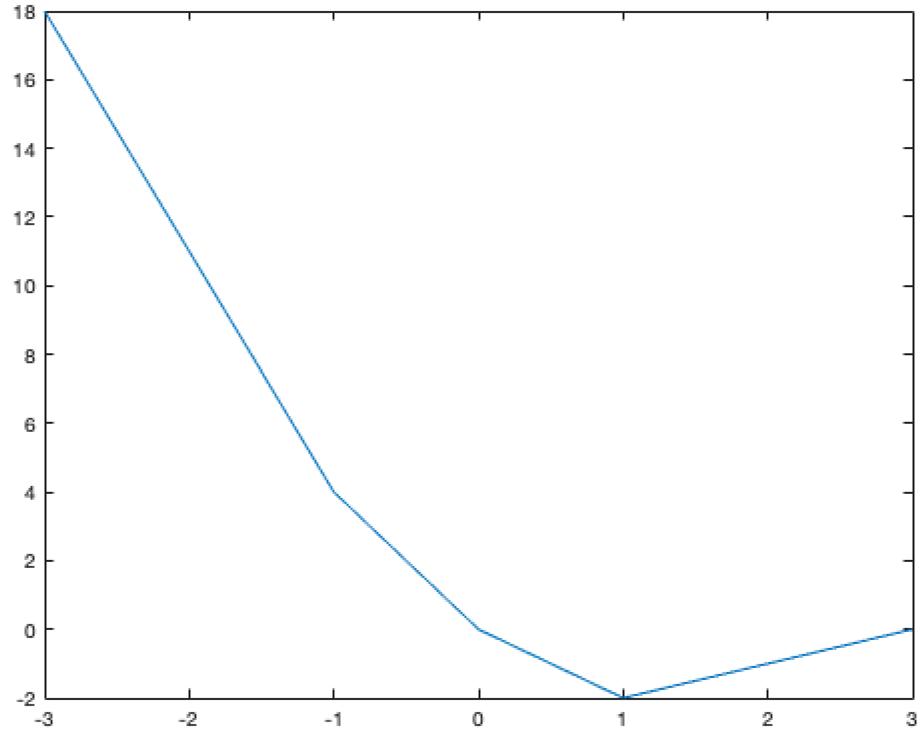
1. Execute and explain the following commands

The following code: Declares a vector x, equal to [-3 -1 0 1 3] Maps the vector x to a vector y using the equation x.*x - 3*x Creates a new plot in figure 4, plotting x on the x axis and y on the y axis. Assigns z to be the sum of x and jy, and then plots it on figure 5.

```
x = [-3 -1 0 1 3];
y = x.*x - 3*x;
plot(x, y);
```



```
z = x + y * sqrt(-1);
plot(z);
```



Question 4: What is the difference between a*b and a.*b, where a and b are matrices?

The difference between a*b and a.*b, where a and b are matrices, is as follows. a*b means matrix multiplication, a special operation that can be performed on only matrices. This operation is very different than element wise multiplication, where each corresponding i and j value is multiplied together.

1.5 - Functions

1. Write a function called oddsummer that takes a positive integer n as its argument and returns the sum of all odd numbers between 1 and n.

```
oddsummer(3)
```

```
ans =

     4
```

```
oddsummer(5)
```

```
ans =
```

```
     9


oddsummer(99)


ans =

        2500
```

2. Write a function called hellos, taking a positive integer n as its argument and displays the word ?hello? n times.

```
hellos(1)

hello

hellos(5)

hello
hello
hello
hello
hello

hellos(10)

hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
```

1.6 - Comments

No work to be completed.

1.7 - Complex Numbers in MATLAB

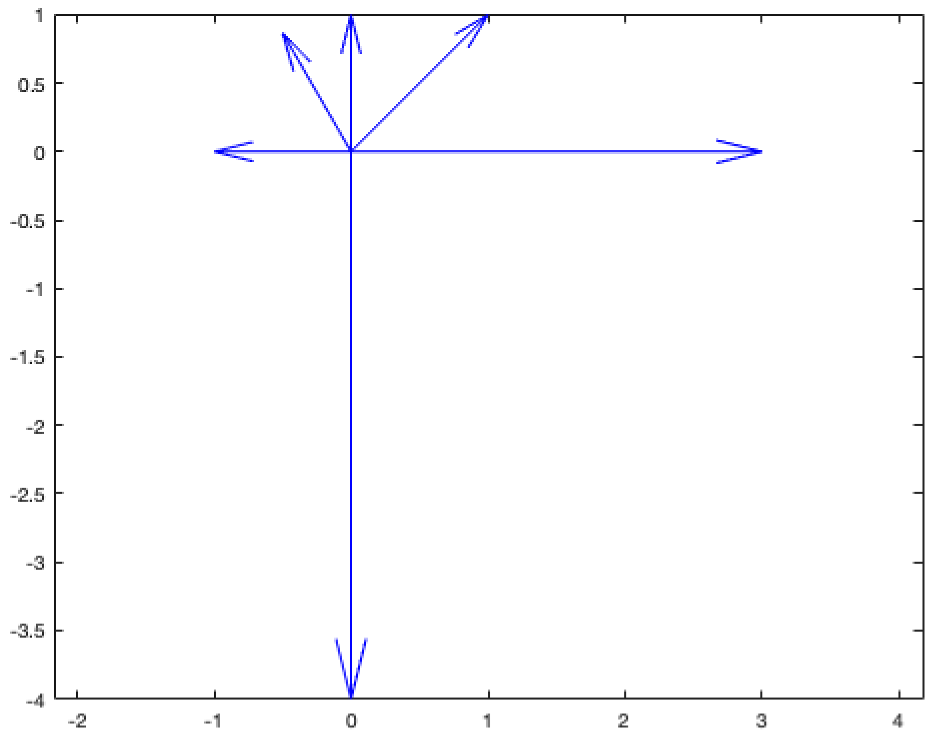
1. MATLAB reserves the letter i or j as sqrt(?1) . In this class, the letter i is usually used as iterator of loops. Hence, the letter j will mostly be reserved as sqrt(?1).

```
z = 3 + 4*j, w = -3 +4*j % Declaring complex numbers
real(z), imag(z) % Real and imaginary components
abs([z,w]) % Computes magnitudes
conj(z+w) % Computes complex conjugate of the sum
angle(z) % Computes the phase of the complex number
```

```
% Also written as atan2(imag(z)/real(z))
exp(j*pi) % Using Euler?s formula, cos(pi)+j*sin(pi)
exp(j*[pi/4, 0, -pi/4])


z =

   3.0000 + 4.0000i


w =

  -3.0000 + 4.0000i


ans =

     3


ans =

     4


ans =

     5     5


ans =

   0.0000 - 8.0000i


ans =

    0.9273


ans =

  -1.0000 + 0.0000i


ans =

   0.7071 + 0.7071i   1.0000 + 0.0000i   0.7071 - 0.7071i
```

2. MATLAB can compute complex valued formulas and display the results as phasor diagrams. Use the following zvect function to plot five vectors all on one graph. Execute and plot:

```
zvect([1 + j, j, 3 -4*j, exp(j*pi), exp(2j*pi/3)]);
```



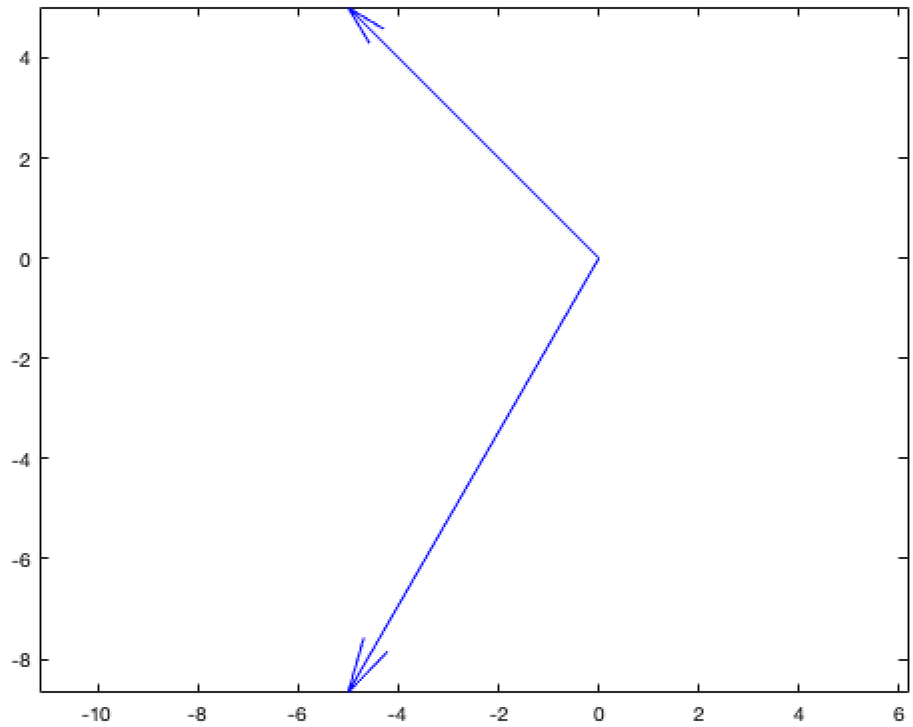
3. Use z1 = 10e^(-j*2*pi/3) and z2 = -5 + 5j for all parts of this section.

```
z1 = 10*exp(-j*2*pi/3);
z2 = -5 + 5j;
```

a. Plot z1 and z2 using zvect and print them with zprint.

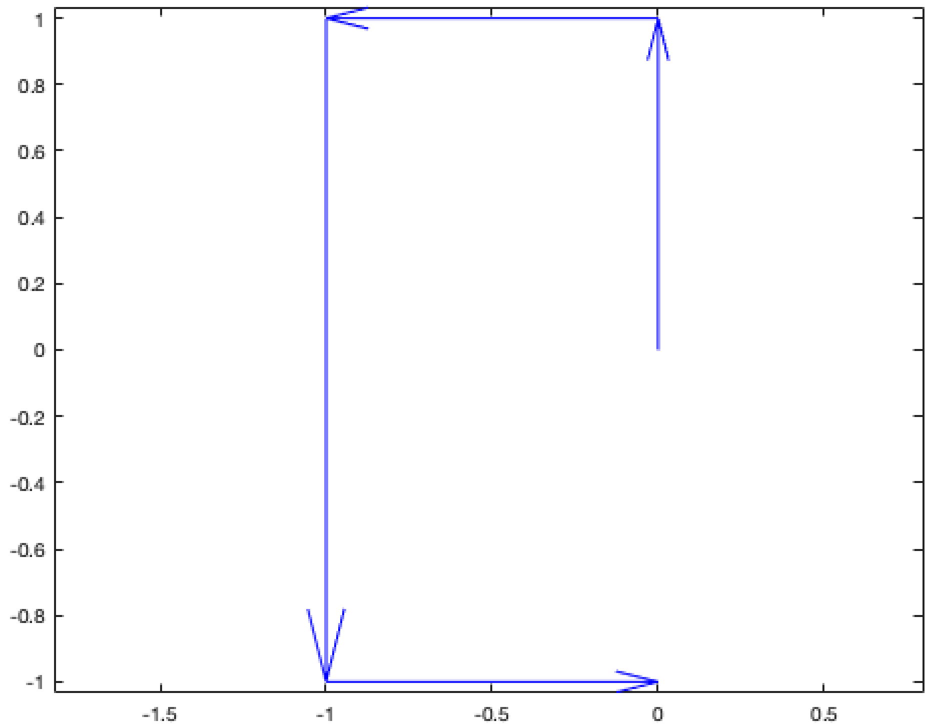
```
zvect([z1,z2]);
zprint([z1,z2])
```

```
 Z =      X     +       jY     Magnitude    Phase     Ph/pi    Ph(deg)
           -5         -8.66           10    -2.094    -0.667   -120.00
           -5             5        7.071     2.356     0.750    135.00
```



b. Execute and explain: The following adds the vectors together and plots each step of the way.

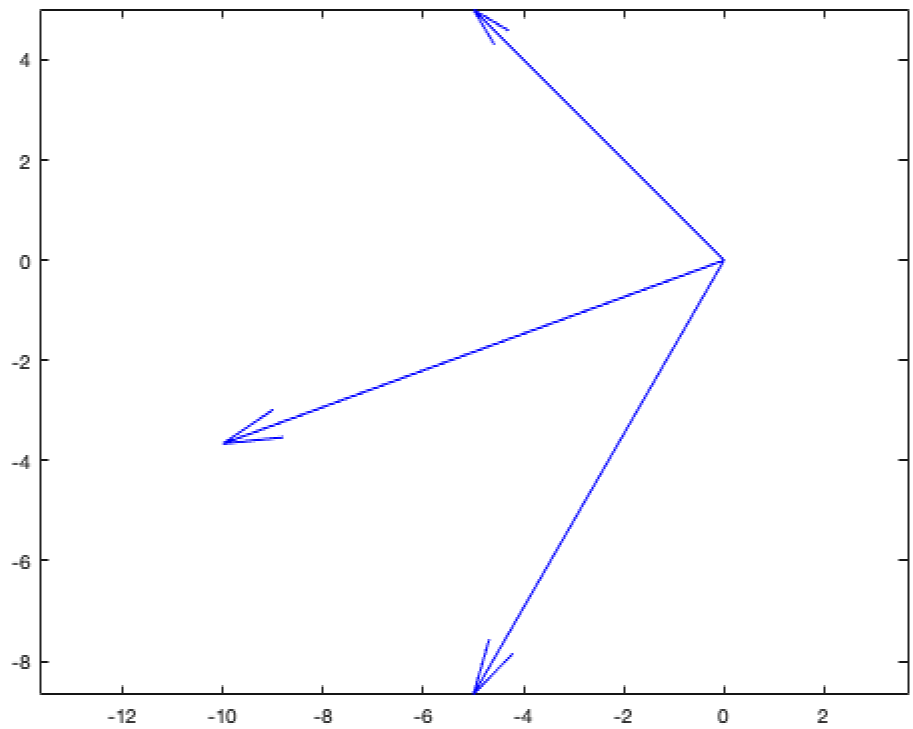
```
zcat([1j, -1, -2j, 1]);
```



c. Compute z1 + z2, and plot the sum using zvect. Using hold on, Plot all three vectors (,,) on the same plot where and are concatenated using zcat. Display the numerical values of z1, z2, z1 + z2

```
zsum = z1+z2;
zprint(z1)
zprint(z2)
zprint(zsum)
zvect([zsum,z1,z2])
```

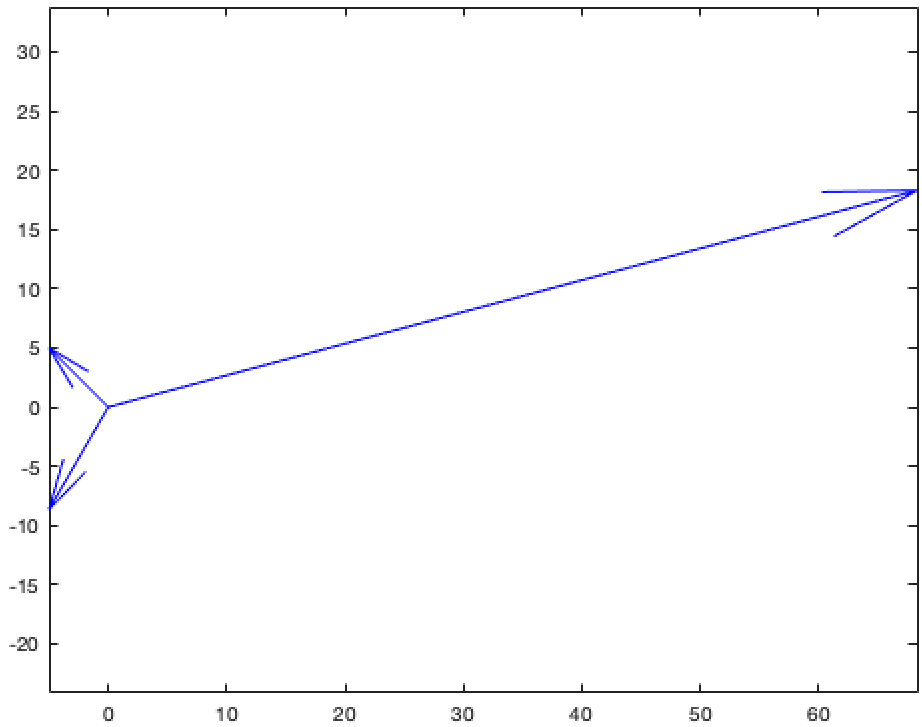
```
Z =      X     +       jY     Magnitude    Phase     Ph/pi    Ph(deg)
          -5         -8.66           10   -2.094    -0.667   -120.00

Z =      X     +       jY     Magnitude    Phase     Ph/pi    Ph(deg)
          -5             5        7.071    2.356     0.750    135.00

Z =      X     +       jY     Magnitude    Phase     Ph/pi    Ph(deg)
         -10         -3.66        10.65   -2.791    -0.888   -159.90
```



d. Compute the product and display the numerical result. Plot z1, z2, and z1z2 on the same plot.

```
zprod = z1*z2;
zprint(z1)
zprint(z2)
zprint(zprod)
zvect([zprod,z1,z2])
```

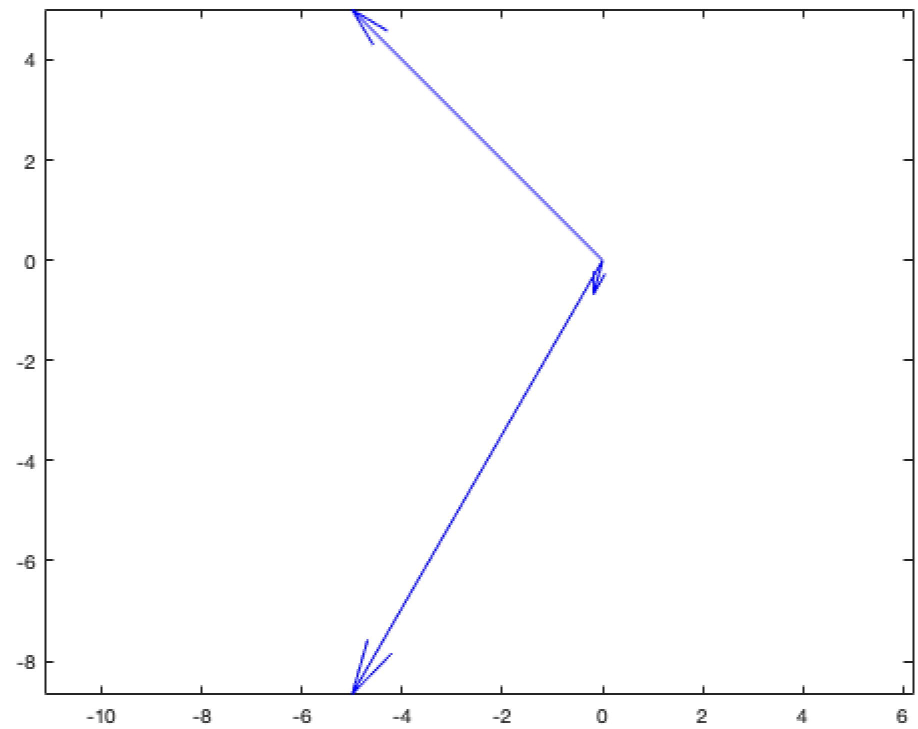
```
 Z =       X     +       jY     Magnitude    Phase     Ph/pi    Ph(deg)
            -5          -8.66          10   -2.094    -0.667   -120.00

 Z =       X     +       jY     Magnitude    Phase     Ph/pi    Ph(deg)
            -5              5       7.071    2.356     0.750    135.00

 Z =       X     +       jY     Magnitude    Phase     Ph/pi    Ph(deg)
          68.3           18.3       70.71    0.262     0.083     15.00
```



e. Compute the quotient z2/z1 and display the numerical result. Plot z1, z2, and z2/z1 on the same plot.

```
zquot = z2/z1;
zprint(z1)
zprint(z2)
zprint(zquot)
zvect([zquot,z1,z2])
```

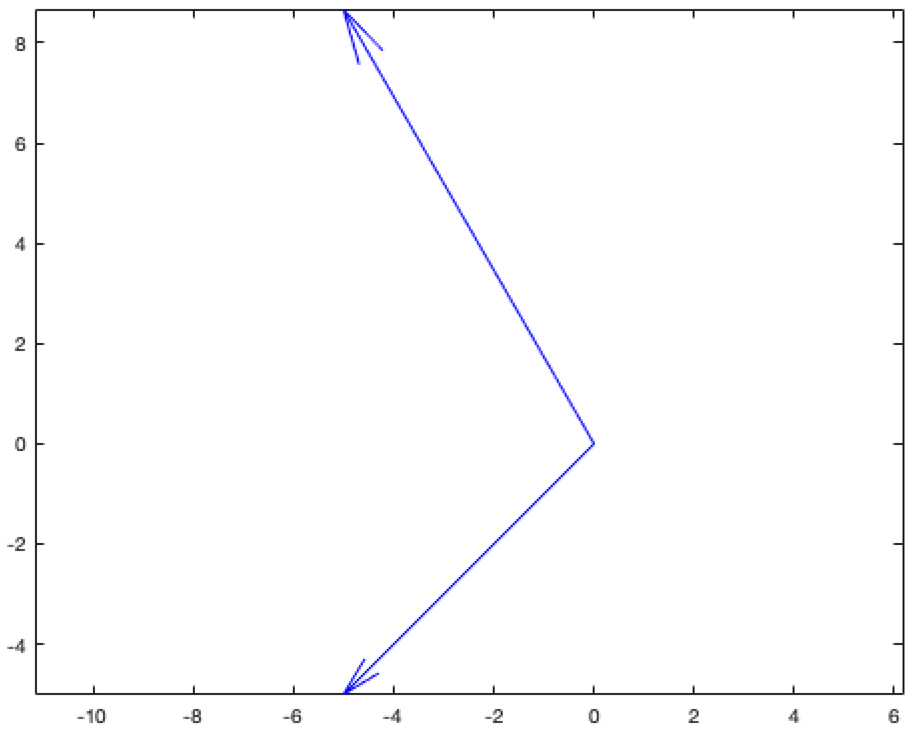
```
 Z =      X     +     jY     Magnitude    Phase    Ph/pi   Ph(deg)
         -5         -8.66          10    -2.094   -0.667  -120.00

 Z =      X     +     jY     Magnitude    Phase    Ph/pi   Ph(deg)
         -5             5       7.071     2.356    0.750   135.00

 Z =      X     +     jY     Magnitude    Phase    Ph/pi   Ph(deg)
     -0.183        -0.683      0.7071    -1.833   -0.583  -105.00
```



f. Compute the conjugates z1 and z2 and display the numerical result. Plot them on the same graph.

```
z1conj = conj(z1);
z2conj = conj(z2);
zprint(z1conj)
zprint(z2conj)
zvect([z1conj,z2conj])
```

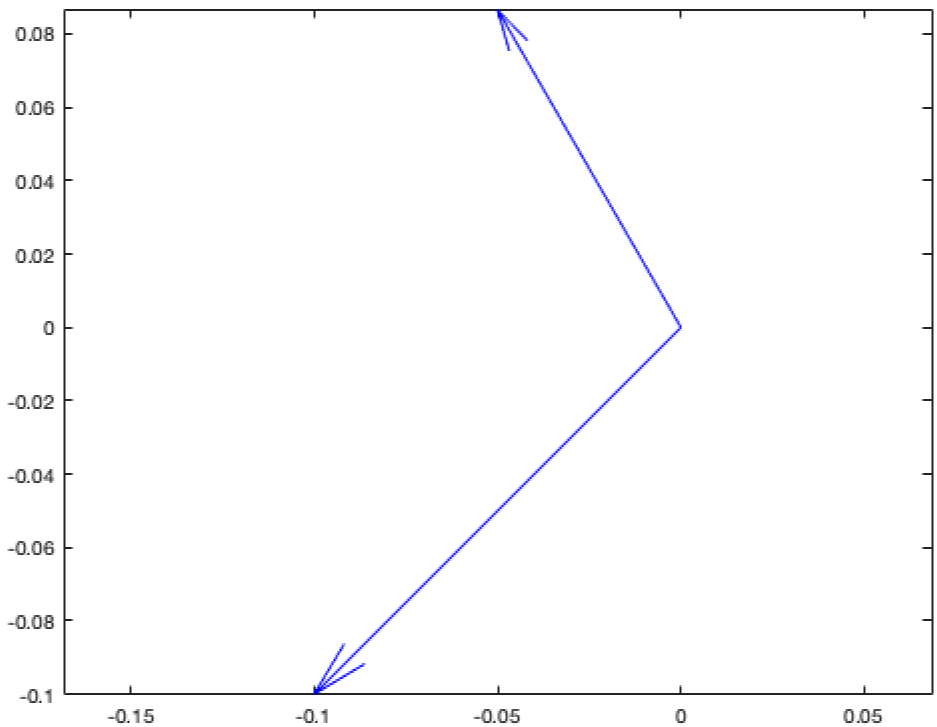
```
Z =      X     +      jY     Magnitude     Phase     Ph/pi    Ph(deg)
          -5          8.66          10     2.094     0.667    120.00

Z =      X     +      jY     Magnitude     Phase     Ph/pi    Ph(deg)
          -5            -5       7.071    -2.356    -0.750   -135.00
```



g. Compute 1/z1 and 1/z2 and display the numerical result. Plot them on the same graph.

```
z1inv = 1/z1;
z2inv = 1/z2;
zprint(z1inv)
zprint(z2inv)
zvect([z1inv,z2inv])
```

```
 Z =      X     +      jY      Magnitude    Phase     Ph/pi    Ph(deg)
        -0.05        0.0866          0.1    2.094     0.667    120.00

 Z =      X     +      jY      Magnitude    Phase     Ph/pi    Ph(deg)
         -0.1          -0.1       0.1414   -2.356    -0.750   -135.00
```



Question 5: What does zcat() do with a vector of complex numbers?

It sums them together as usual, adding real components and complex components of each of the numbers together and then plotting them.

Question 6: What is the relationship between the two initial angles and the angle of the product?

The relationship between the two initial angles and the angle of the product is that the two initial angles are summed together to get the angle of the product.

Question 7: Why is the plot of real(zz) a sinusoid even though no cos or sin is present in its equation?

Although there is no cos or sin present in the equation of zz, we know that the form that it is written in is equivalent to some Acos(wt)+jAsin(wt), so it makes sense that its plot is sinusoidal.

Question 8: What are its phase and amplitude? Calculate the phase based on a time-shift measured from the plot. Take a screenshot of the plot.

From the equation zz = 1.4*exp(j*pi/2)*exp(j*5*pi*tt), we can see that the amplitude is equal to 1.4, as the amplitude is always equal to the simplified coefficient of the exponential. One can also clearly see this amplitude on the screenshot of the plot I have included. Additionally, I have extrapolated two x coordinates with the same height (1.3828). These data points occur at x2 = 0.29 and x1 = -0.09. From these points, we can find omega (2.632) and tau (closest peak to origin, occurring at -0.09). From this, we can easily find phi, our phase shift, which is equal to ~0.237.

```
x2 = 0.29;
x1 = -0.09;
```

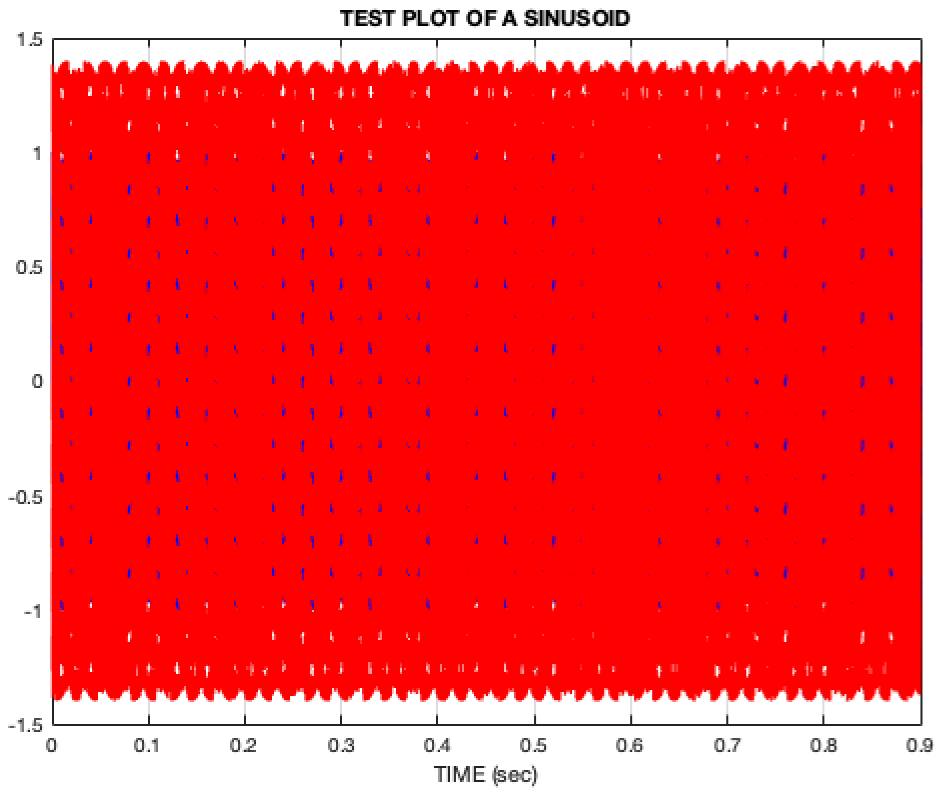
```
omega = 1/(x2-x1)
tau = x1
phi = -omega*tau
```

```
omega =

    2.6316


tau =

   -0.0900


phi =

    0.2368
```

1.8 - Sounds

1. Run the MATLAB sound demo by typing xpsound. Sounds are represented by sinusoids, so to create sound using MATLAB, you need to create a sinusoid. Your script mylab1.m creates a sinusoid with frequency 2.5 Hz. Modify your script so that it produces a 2000 Hz sound, with sampling frequency 11025 Hz, which is 0.9 seconds long. The appropriate time vector is therefore tt = 0:1/11025:0.9; Now use soundsc() in order play the sound. Play your generated sound for your TA, who will check you off on canvas.

```
mylab1
soundsc(real(zz))
```



Question 9: What is the length of your tt vector?

The length of the tt vector can be seen below, it is 9923.

```
length(tt)
```

```
ans =

        9923
```

Published with MATLAB® R2019a