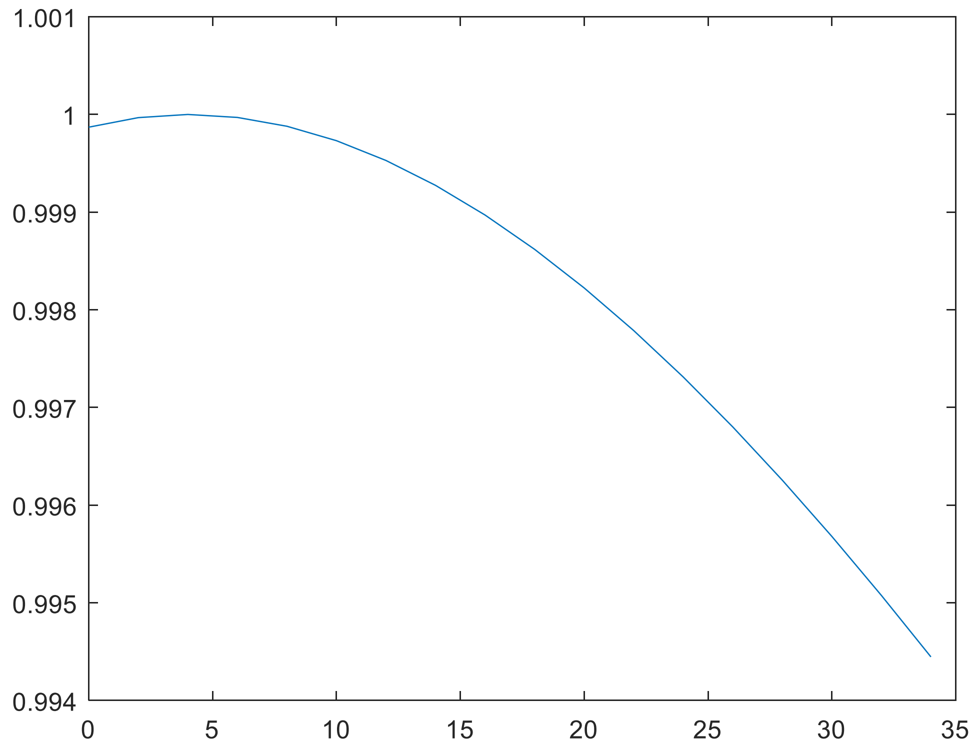
Edward Zhao, HW1

1.

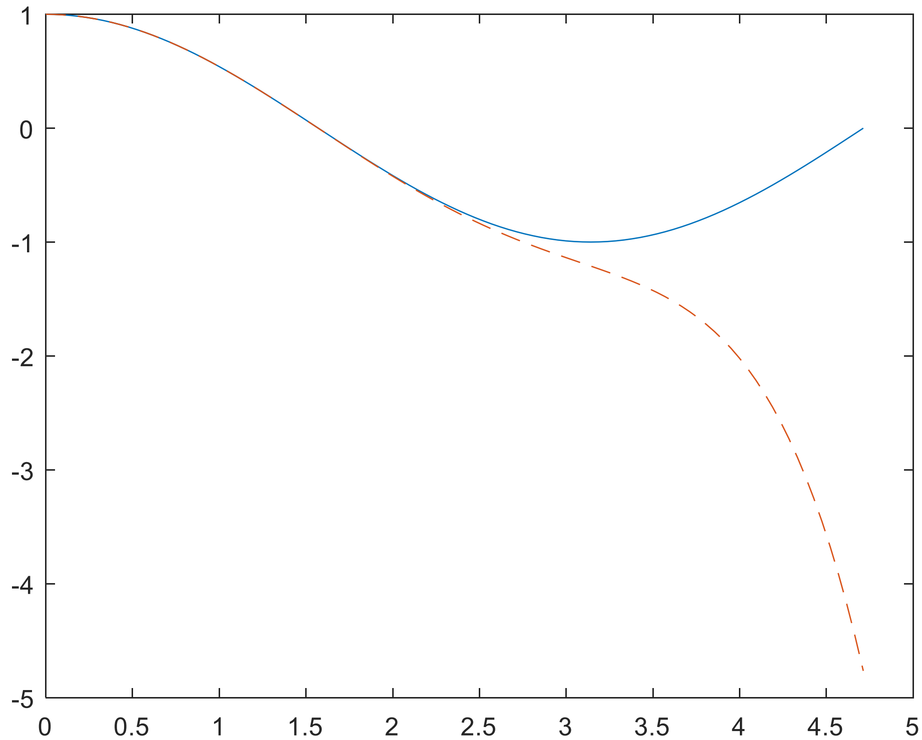
```
temp_range=32:3.6:93.2
temp_rangeC=5/9*(temp_range-32)
density=5.5289*10^-8 * temp_rangeC.^3 - 8.5016*10^-6*temp_rangeC.^2 +
6.5622*10^-5 * temp_rangeC+0.99987
plot(temp_rangeC,density)
```




The vector of temperature ranges is first given in Fahrenheit and then converted to Celsius. The density is computed for each value of temperature in the vector. Then the plot is drawn so that temperature in C is on the x axis and density in g/cm^3 is on the y axis.

2.

```
x=linspace(0,3*pi/2)
y=cos(x)
y2=1-x.^2/2+x.^4/factorial(4)-x.^6/factorial(6)
plot(x,y,x,y2,'--')
```




The x-coordinates are generated using linspace to get 100 equaly spaced points between 0 and 3pi/2. Cos(x) is stored in y, which is a vector with cos applied to the points in x. The Maclaurin approximation is stored in y2.  The first 2 arguments in plot draw cos(x) vs x and the next 2 draw the approximation vs x. The last argument specifies that the Maclaurin approximation is a dashed line.

3.

```
function [r, theta] = fun(x,y)
r=sqrt(x .^2 + y .^2);
if x<0
    if y>0
        theta=atan(y/x)+pi;
    elseif y<0
        theta=atan(y/x)-pi;
    else
        theta=pi;
    end
elseif x>0
    theta=atan(y/x);
else
    if y>0
        theta=pi/2;
    elseif y<0
        theta=-pi/2;
    else
        theta=0;
    end
```

```
end
theta=theta*180/pi;
```

| *x* | *y* | *r* | *theta* |
|---|---|---|---|
| 2 | 0 | 2 | 0 |
| 2 | 1 | 2.2361 | 26.5651 |
| 0 | 3 | 3 | 90 |
| -3 | 1 | 3.1623 | 161.5651 |
| -2 | 0 | 2 | 180 |
| -1 | -2 | 2.2361 | -116.5651 |
| 0 | 0 | 0 | 0 |
| 0 | -2 | 2 | -90 |
| 2 | 2 | 2.8284 | 45 |

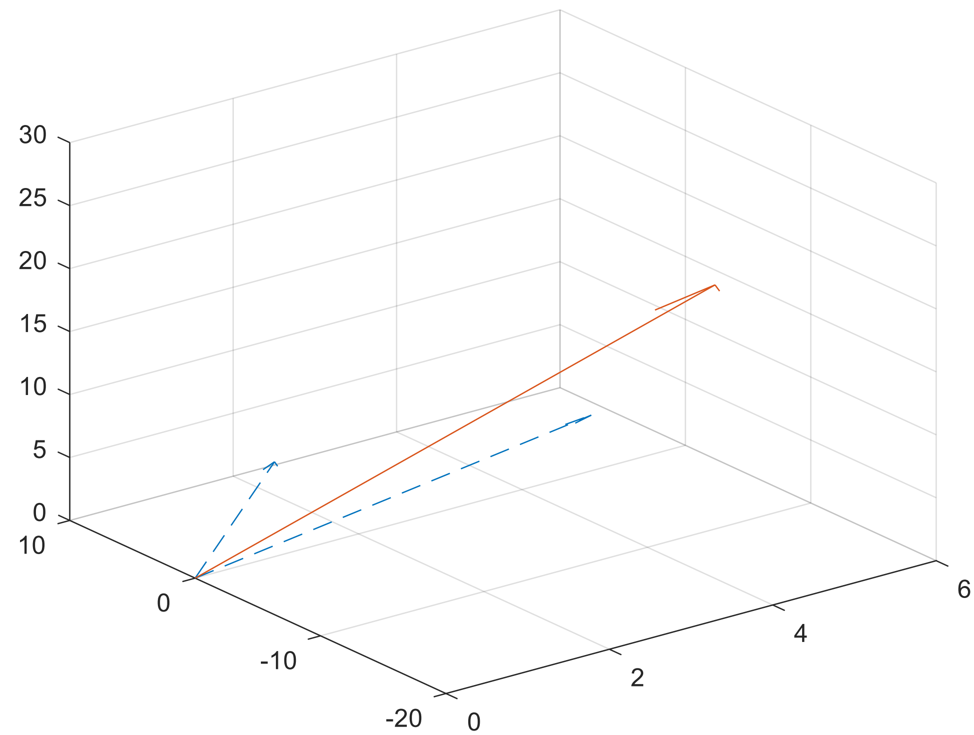
The function has outputs r and theta, and inputs x and y. First r is calculated using the formula given. The if/else statements first test if x is less than 0, after which it tests if y is greater than 0, then y<0, and then the only remaining case is if y=0. The elseif case is for when x >0. Then the next else is for the only remaining case which is x=0, where we again go through the same process for y as for when x<0. For each case, the formula from the given table is used to find theta. At the end, theta is converted to degrees from radians. Evaluating the cases involves running [r, theta]=fun(x,y) where the values for x and y are inputted.

4.

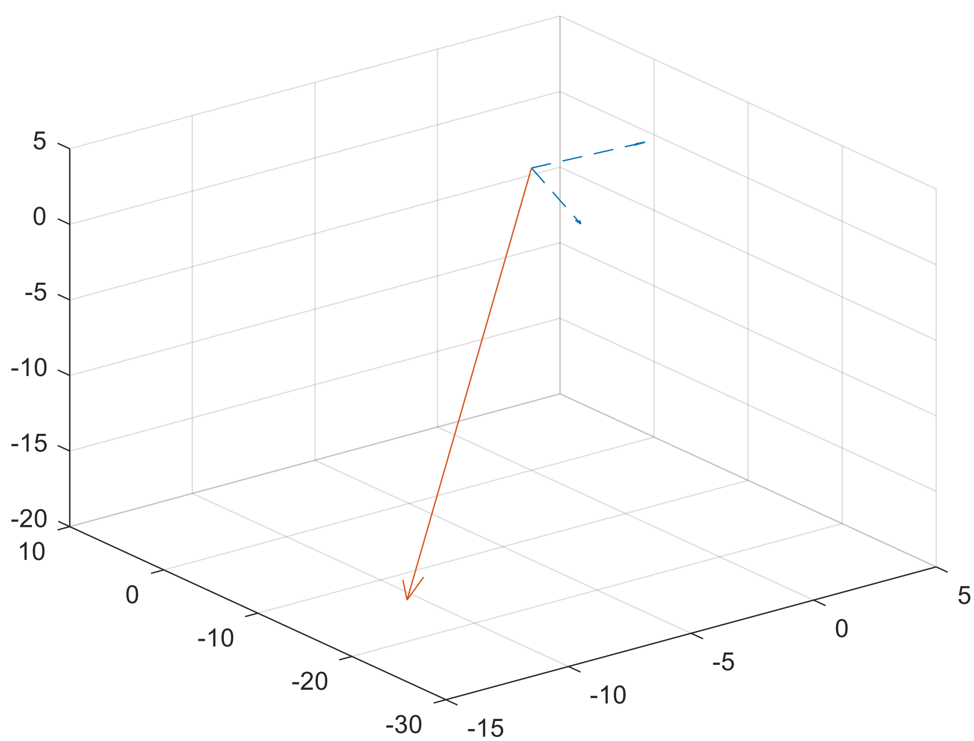
```
function [theta, c, magnitude] = fun2(a,b)
theta=acos(dot(a,b)/(norm(a)*norm(b)))
c=cross(a,b);
theta=theta*180/pi;
magnitude=sqrt(c(1)^2+c(2)^2+c(3)^2);
m=[a ; b];
quiver3([0;0],[0 ;0],[0 ;0],m(:,1),m(:,2),m(:,3),'--');
hold on;
quiver3(0,0,0,c(1),c(2),c(3),'-');
```

The function has outputs theta, c, and the magnitude of c, and takes inputs x, y which are vectors. First theta is calculated as the inverse cosine of the dot product of the vectors. The denominator is to normalize the vector. Theta needs to be converted to degrees. The cross product can be found using the cross function. Finally the magnitude of c is just using the 3 dimensional version of the Pythagorean theorem. To evaluate, we would do [theta, c, magnitude]= fun2( [x1 x2 x3], [y1 y2 y3]. m is a matrix that stores a and b with each vector being a row. Quiver3 allows us to plot a and b using 0 as the starting point and the columns of m for the x, y, and z of the end point of the vector. This first one is dashed. Then the hold makes sure we keep the current plot, and then plot c using similar logic except for a solid instead of dashed line.

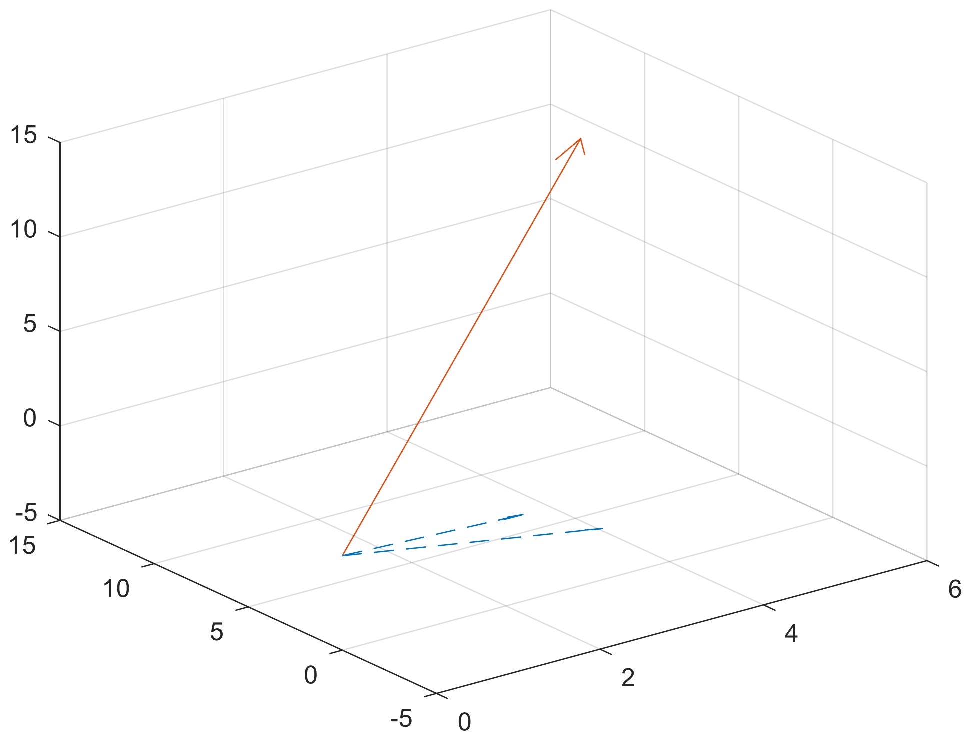
a) theta = 38.2132, c = [4 -20 28], magnitude= 34.6410



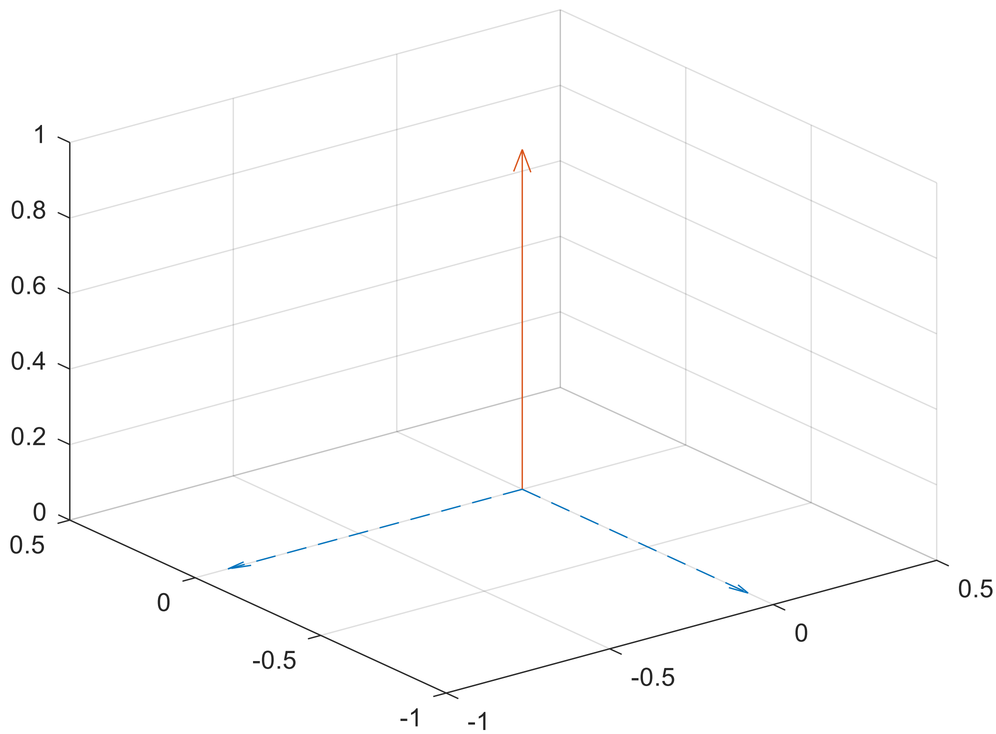

b) theta = 90, c= [-16 -27 -17], magnitude=35.6931



c) theta = 90, c= [6 12 12], magnitude= 18




d) theta = 90, c= [0 0 1], magnitude = 1


1
0.8
0.6
0.4
0.2
0
0.5
0
-0.5
-1
-1
-0.5
0
0.5