1. First, I generated a vector of the temperatures in Fahrenheit from 32 to 93.2 in increments of 3.6. Then, I applied the scalar conversion of Fahrenheit to Celsius to that vector since Matlab does that for each element of the vector. I then generated a new vector for the densities for each temperature by passing in each element of the Celsius temperatures to the density formula by using arrayfun. Finally, I plotted it versus the Celsius temperatures.

```
% Vector of temperatures in fahrenheit
tempFahrenheit = 32:3.6:93.2;

% Vector of temperatures in celsius
tempCelsius = 5/9*(tempFahrenheit - 32);

% Densities for each temperature in celsius
densities = arrayfun(@(x) 5.5289*10^(-8)*x^3 ...
  - 8.5016*10^(-6)*x^2 ...
  + 6.5622*10^(-5)*x ...
+ 0.99987, tempCelsius);

% Plot of density versus temperature in celsius
plot(tempCelsius, densities)
title('Density of freshwater for temperature(C)')
```

2. First, I generated a vector of the potential values of x from 0 to $3\pi/2$. I arbitrarily made x increment by $\pi/100$ since it was probably sufficiently accurate. I then generated a vector of the approximations of cosine by using the Maclaurin series expansion by passing each element of the vector of x to the series by using arrayfun. Finally, I plotted x versus y with a dashed black line.

```
% Define x from 0 to 3pi/2 with arbitrary steps of pi/100
x = 0:pi/100:3*pi/2;

% Define y as the Maclaurin series for cosine up to x^8
y = arrayfun(@(x) 1 - x^2/factorial(2) ...
               + x^4/factorial(4) ...
               - x^6/factorial(6) ...
               + x^8/factorial(8), x);

% plot y versus x
plot(x, y, '--k')
title('Approximation of cosine using the Maclaurin series')
```

3. In the function that I used to convert (x, y) to (r, θ), I calculated r right away since it would be the same regardless of what x and y were. Then, in the first branch of an if statement, I checked to see if x ¿ 0 since θ was calculated the same way. Then, in the next branch, I checked if x ¡ 0. From there, I checked further if y ¿ 0, y ¡ 0, or y ==

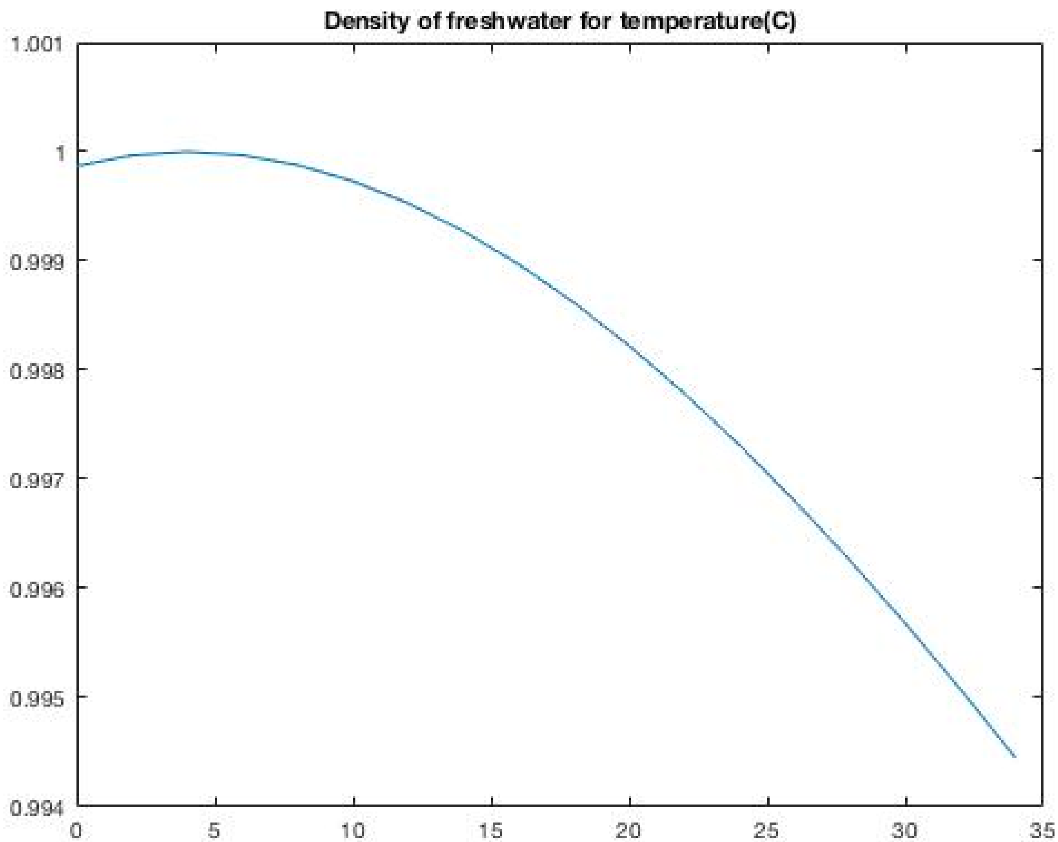

Figure 1: Problem 1

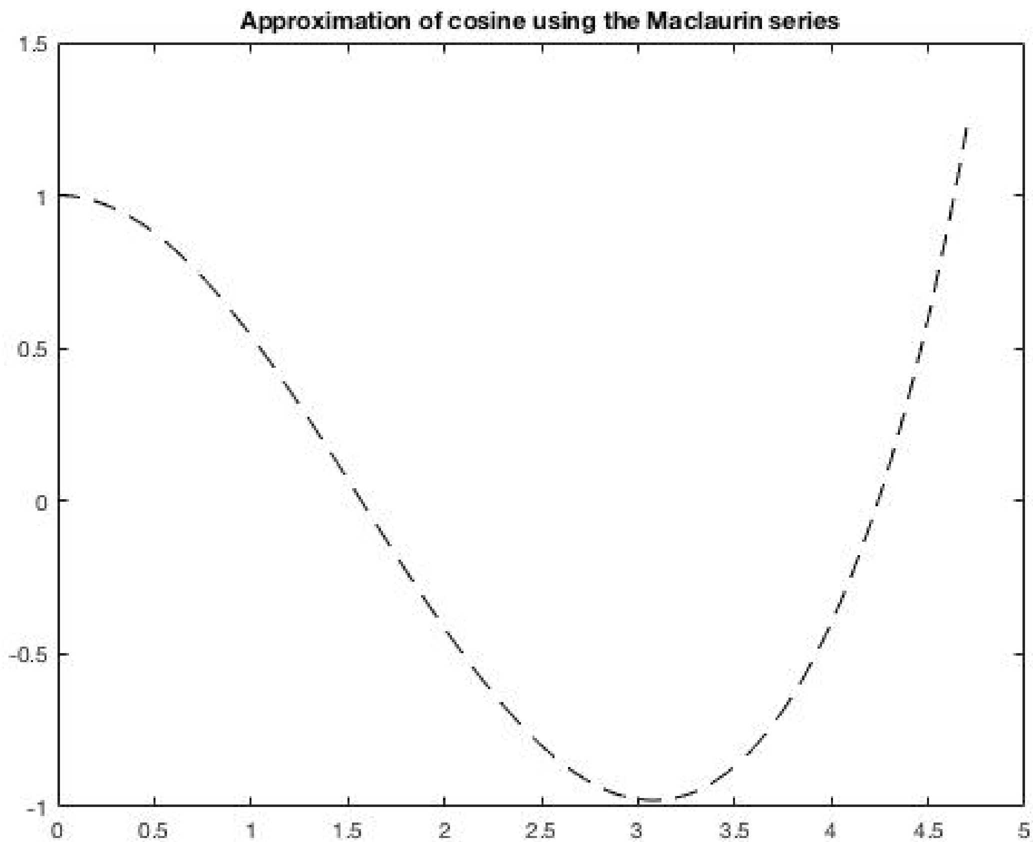

Figure 2: Problem 2

0, and I calculated θ accordingly. I checked for y in the same way in the last branch of the if statement where x == 0.

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

4. In the funtion that I used to find the cross product between a and b, I first plotted both a and b with a helper function, which also set them both at the origin, which I did by appending a row of zeroes to the top of the matrix representing each vector. This let me also plot vector c easily later on. I calculated c by using the built-in cross function, I calculated the magnitude with the norm function, and I calculated the angle between a and b with atan2d. The plot for a = $[-1 \quad 0 \quad 0]$ and b = $[0 \quad -1 \quad 0]$ is shown below.

```
function [theta, c, magC] = crossProd(a, b)
  hold on
  view(3);
  displayVector(a, '--');
  displayVector(b, '--');
  c = cross(a, b);
  displayVector(c, '-');
  magC = norm(c);
  theta = atan2d(magC, dot(a, b));
end

function displayVector(vector, option)
  origin = [0 0 0];
```

```
    originVector = [origin;vector];
    plot3(originVector(:,1)...
          , originVector(:,2)...
          , originVector(:,3), option);
end
```

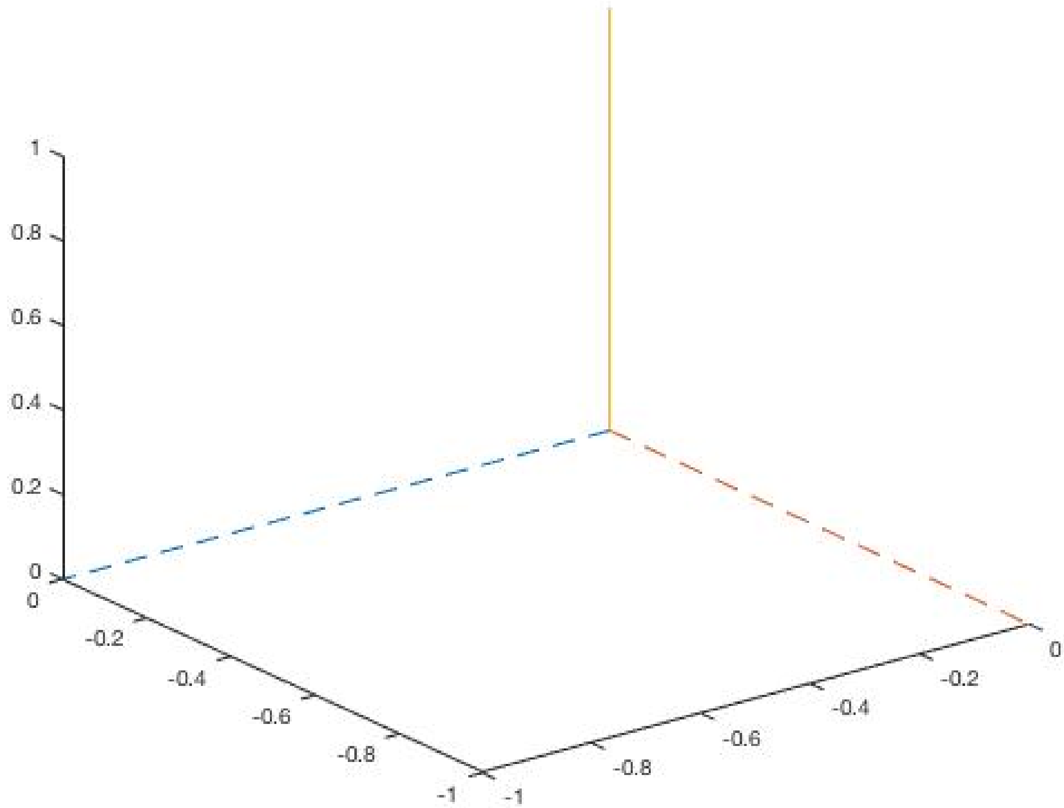

Figure 3: Problem 4