MATH 320 – HW 1

NAME: ROHAN SARAOGI

(1)

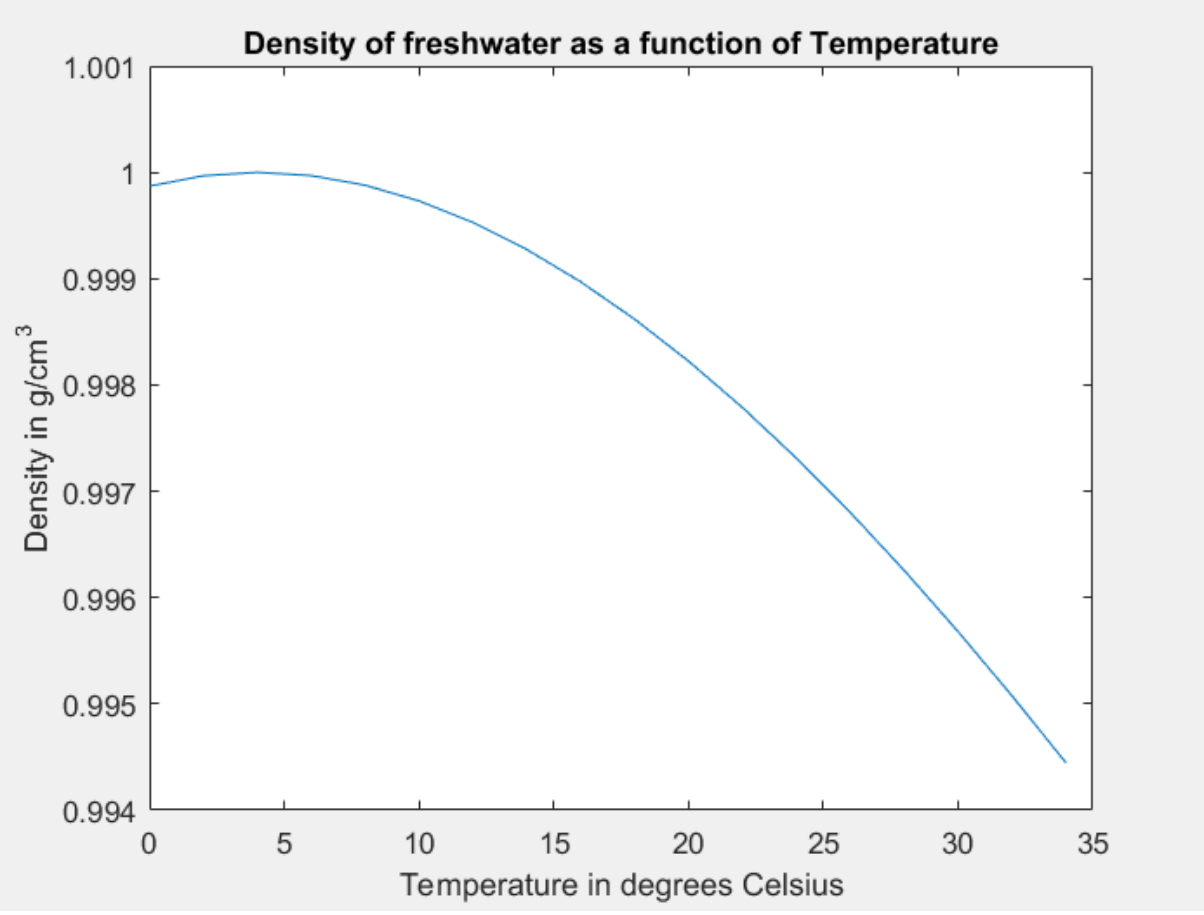
CODE

```
tf = 32:3.6:93.2;
tc = (tf - 32) * 5 / 9;
p = 5.5289 * (10^-8) * (tc.^3) - 8.5016 * (10^-6) * (tc.^2) + ...
   6.5622 * (10^-5) * tc + 0.99987;
plot(tc, p);
title('Density of freshwater as a function of Temperature');
xlabel('Temperature in degrees Celsius');
ylabel('Density in g/cm^3');
```

NOTES

tf = Temperature vector (°F)
tc = Temperature vector (°C)
p = density (g / cm^3)
The code converts tf to tc and then applies the formula to calculate p. The p vs. tc graph is then plotted.

OUTPUT



(2)

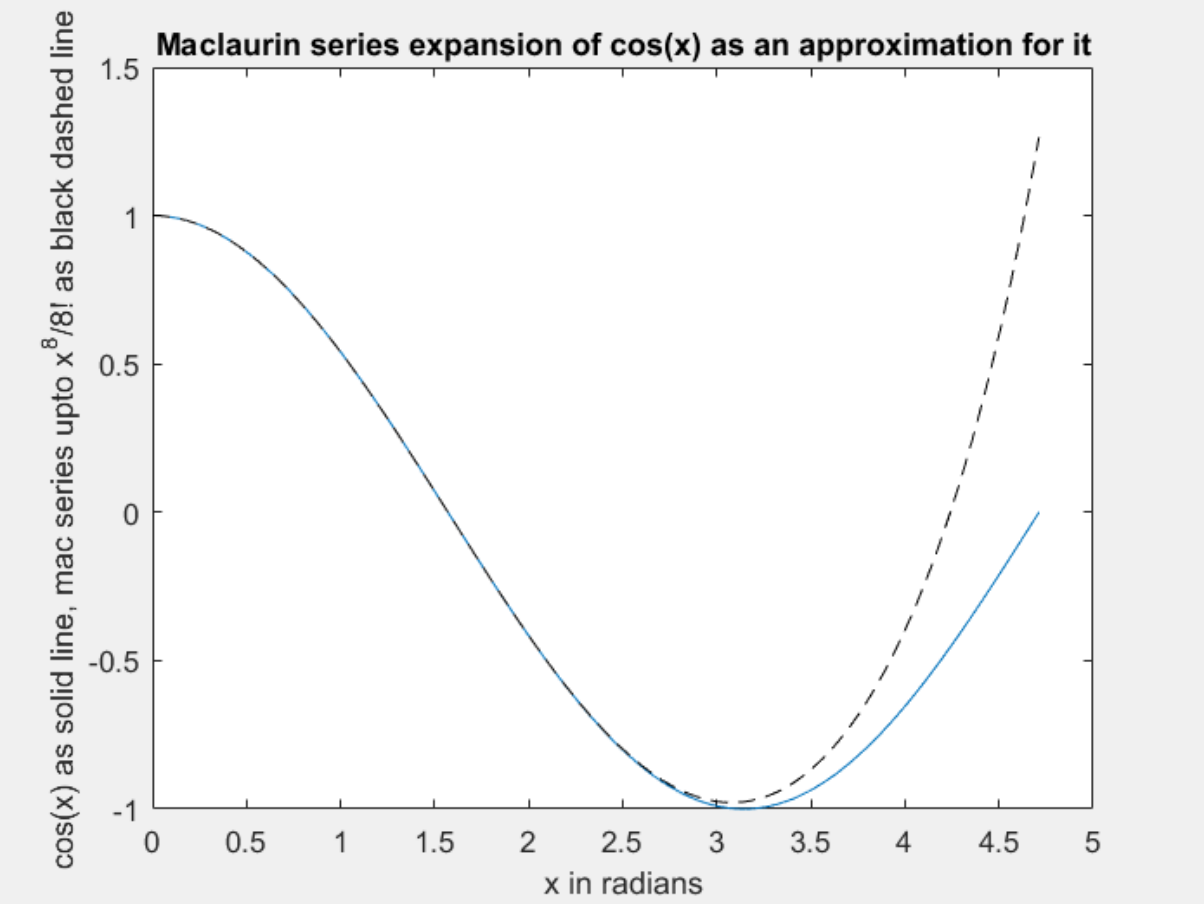
CODE

```
x = 0:0.001:3 * pi / 2;
y = cos(x);
z = 1 - (x.^2)/factorial(2) + (x.^4)/factorial(4) - ...
   (x.^6)/factorial(6) + (x.^8)/factorial(8);
plot (x, y, '-', x, z, 'k--');
title('Maclaurin series expansion of cos(x) as an approximation for it')
xlabel('x in radians');
ylabel('cos(x) as solid line, mac series upto x^8/8! as black dashed line');
```

NOTES

x = Radian vector between 0 and 3π/2 with a spacing of 0.001.
y = Corresponding cosine vector
z = Vector composed of an approximation of the maclaurin series upto to $x^8/8!$ term.
The code calculates the values of the y and z vectors and plots them for comparison.

OUTPUT



(3)

CODE

```
function [ r, a ] = cartesian_to_polar( x, y )
r = sqrt(x^2 + y^2);
if x > 0
   a = atan(y / x);
elseif x < 0 && y > 0
   a = atan(y / x) + pi;
elseif x < 0 && y < 0
   a = atan(y / x) - pi;
elseif x < 0 && y == 0
   a = pi;
elseif x == 0 && y > 0
   a = pi / 2;
elseif x == 0 && y < 0
   a = -pi / 2;
elseif x == 0 && y == 0
   a = 0;
else
   error('Please enter real arguments.');
end
a = radtodeg(a);
end
```

NOTES

r = radius

a = angle about positive x axis

The code accepts the Cartesian coordinates x and y of a point as input. It begins by computing the radius. It then steps through the else if else loop to set the 'a' value depending on the location of the point on the Cartesian plane. If the user does not enter real values for x and y, the function terminates prematurely with an error message. Otherwise, it converts the 'a' value from radians to degrees and returns r and a as a vector.

OUTPUT

x	**y**	**r**	**Θ (Degrees)**
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

(4)

CODE

```
function [ o, c, m_c ] = cross_product( a, b )
dot_product = a * b';
m_a = sqrt(a * a');
m_b = sqrt(b * b');
o = acos(dot_product / (m_a * m_b));
c = [(a(2) * b(3) - a(3) * b(2)) (a(3) * b(1) - a(1) * b(3)) (a(1) * b(2) - a(2) * b(1))];
m_c = sqrt(c * c');
plot3([0 a(1)], [0 a(2)], [0 a(3)], '--', [0 b(1)], [0 b(2)], [0 b(3)], '--', ...
   [0 c(1)], [0 c(2)], [0 c(3)], '-')
xlabel('x'); ylabel('y'); zlabel('z');
end
```

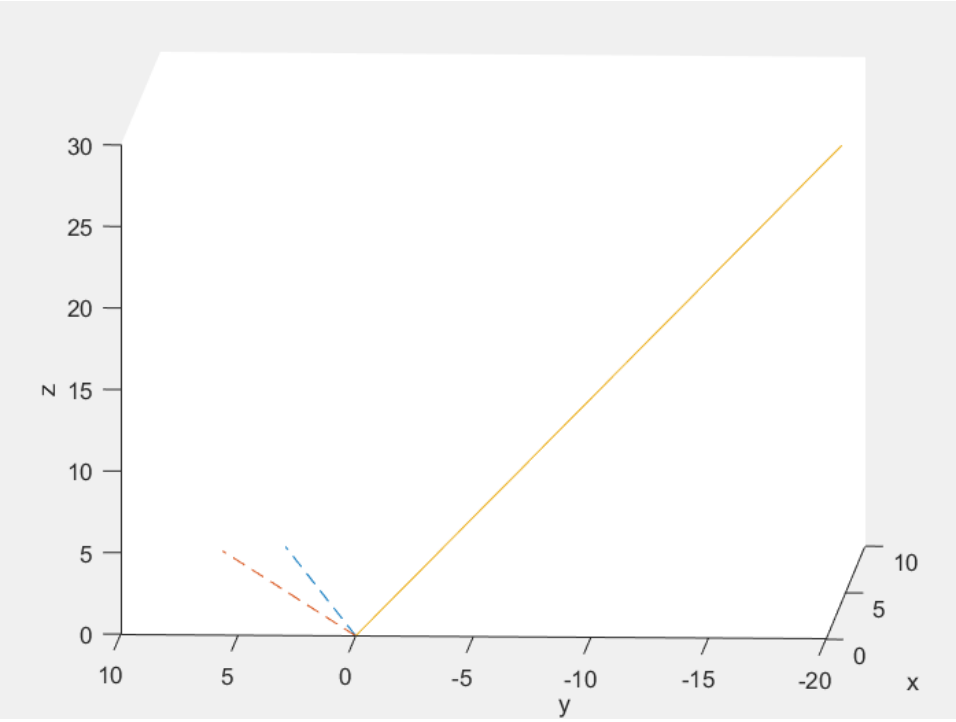
NOTES

o = angle between a and b
c = perpendicular vector
m_c = magnitude of c
The code accepts two vectors in R^3 with Cartesian coordinates as input. It computes o, c and m_c. c is calculated using the component form of the cross product. The three vectors a, b, and c are then plotted in 3 dimensional Cartesian space.

OUTPUT

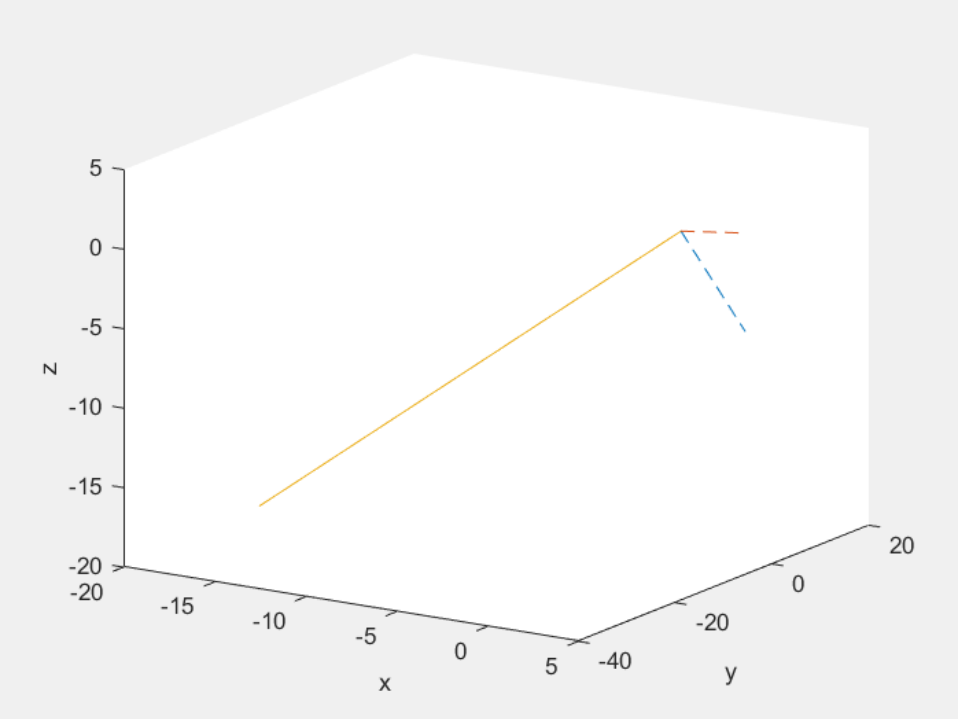
(a)

Θ = 0.6669 radians, c = [4 -20 28], magnitude = 34.6410



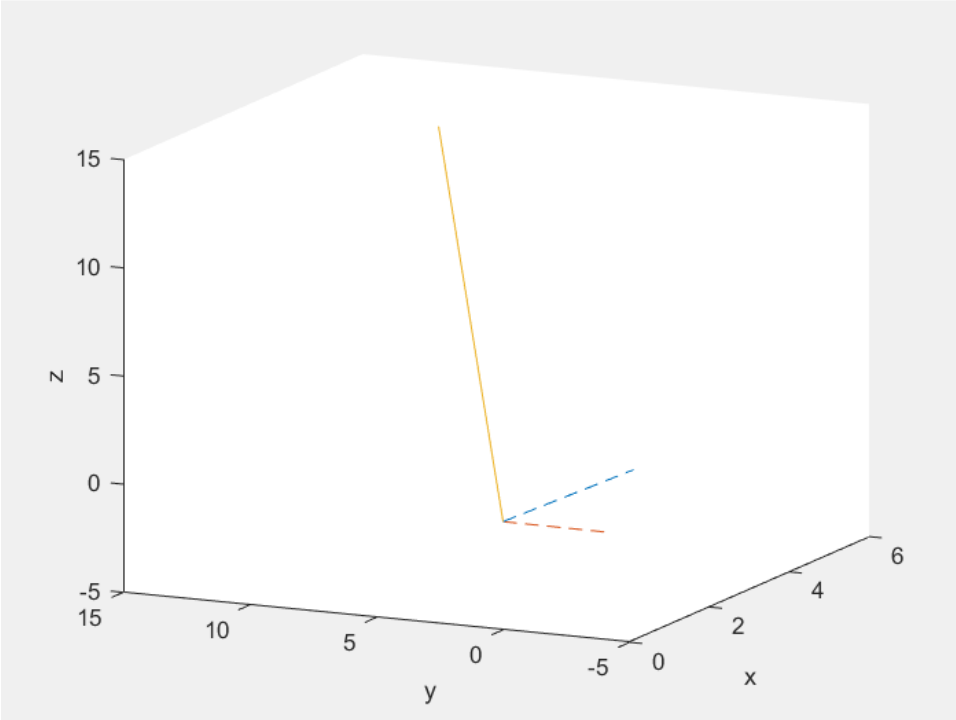
(b)

Θ = 1.5708 radians, c = [-16 -27 -17], magnitude = 35.6931



(c)

Θ = 1.5708 radians, c = [6 12 12], magnitude = 18



(d)

Θ = 1.5708 radians, c = [0 0 1], magnitude = 1