ENSC 180 - Assignment 3

Shengcong Zhou
Lab group 9
sza111@sfu.ca

1.
Function 1

```
%This function calculates the depth of a sphere submerged in fluid by
%following steps
%1.prompting for radius of sphere, the densities of sphere and fluid
%2.calculate the volume submerged using the inputed values and with
%different values of depth
%3.Take the difference of the volumes from two different calculations done
%in 2. to find the root (difference <= eps)
%variables are listed below
%depth is the depth the sphere sinks
%radius is the radius of the sphere
%densityS and densityF are the densities of sphere and fluid respectively
%volumeSub is the volume of sphere below fluid calculated using Eqn1
function depthSub
%Prompt for inputs
radius=input('Enter the radius of sphere (mm)\n');
densityS=input('Enter the density of sphere (g/mm^3)\n');
densityF=input('Enter the density of fluid (g/mm^3)\n');
if densityS > densityF
   error('This program is not designed for cases when the sphere keeps sinking; density of sphere <= density of fluid');
end
%Calculations
volumeSub=(densityS/densityF)*((4/3)*pi*(radius^3)); %Eqn1

h=0:0.01:(radius*2);
difference=(pi.*(3.*radius.*(h.^2)-(h.^3)))./3-(volumeSub); %Eqn2 - Eqn1

for i=1:(radius*2/0.01)
   if densityS==densityF
      depth=radius*2;
   elseif difference(i)*difference(i+1)<0
      depth=h(i);
   end
end
fprintf('The sphere depth below the fluid surface is: %.2f (mm)\n',depth);
end
```

Output when r=40mm, ρ_S =0.6 g/mm^3 and ρ_0 =1.0 g/mm^3

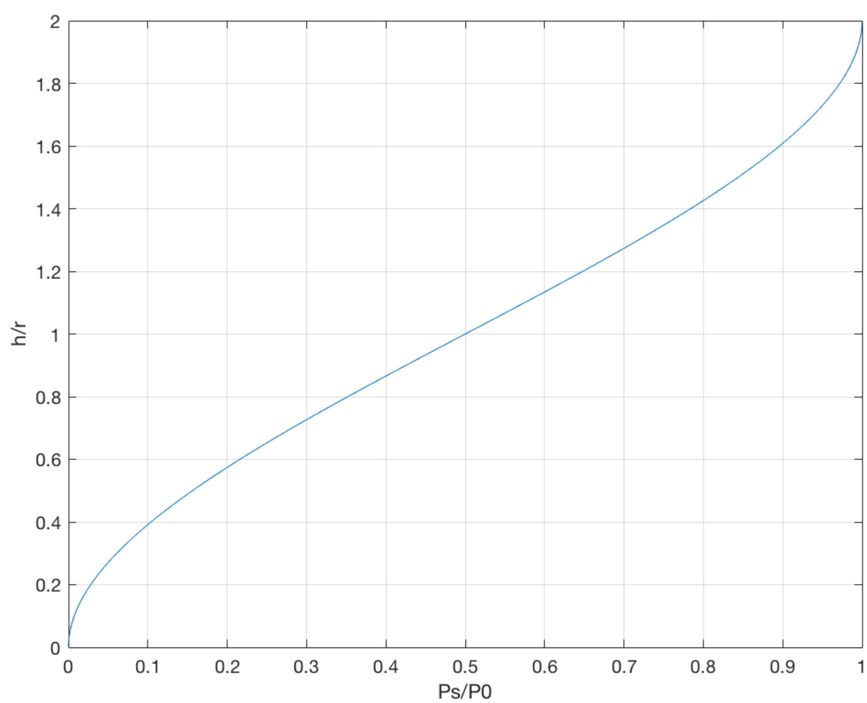
```
>> depthSub
Enter the radius of sphere (mm)
40
```

Enter the density of sphere (g/mm^3)
0.6
Enter the density of fluid (g/mm^3)
1
The sphere depth below the fluid surface is: 45.36 (mm)

Function 2

```
%This function calculates the rate of h/r VS Ps/P0
%The function can use an arbitrary radius in this case 4
%The maximum of the depth is the diameter of the sphere
%variables are listed below
%radius is the radius of the sphere
%h is the depth submerged
%ration_PsP0 is Ps/P0
%ration_hr is h/r
function ratio
radius=4; %An arbitrary radius
h=0:0.01:(radius*2);
ratio_PsP0=(3.*radius.*(h.^2)-(h.^3))./(4.*(radius.^3)); %Ps/P0 in terms of r and h
ratio_hr=h./radius;

plot(ratio_PsP0,ratio_hr)
grid
xlabel('Ps/P0')
ylabel('h/r')
end
```

Output



When the density ratio is 1 h/r=2 meaning h is the radius of the sphere and the whole sphere is submerged. When density ratio is 0.5 only half of the sphere is submerged and as the density ratio decreases h/r decreases meaning less part of sphere sinks.

2.
a)
Function

```
%This function calculates the height and speed of a rocket at a given time
%Where height is given by 2.13*time^2 - 0.0013*time^4 + 0.000034*time^4.751
%and speed is given by the absolute value of the 1st order derivative of
%height with respect to time
function R_motion(time)
if time > 63.01
   error('The rocket lands approximately at t=63.01s. Time must be less than 63.01s');
end
height=2.13*time^2 - 0.0013*time^4 + 0.000034*time^4.751;
speed=abs(4.26*time - 0.0052*time^3 + 0.000161534*time^3.751);

fprintf('Height=%f (m)\nSpeed=%f (m/s)\n', height, speed);
end
```

Output when time = 50

```
>> R_motion(50)
Height=1211.302439 (m)
Speed=55.846042 (m/s)
```

---

b)

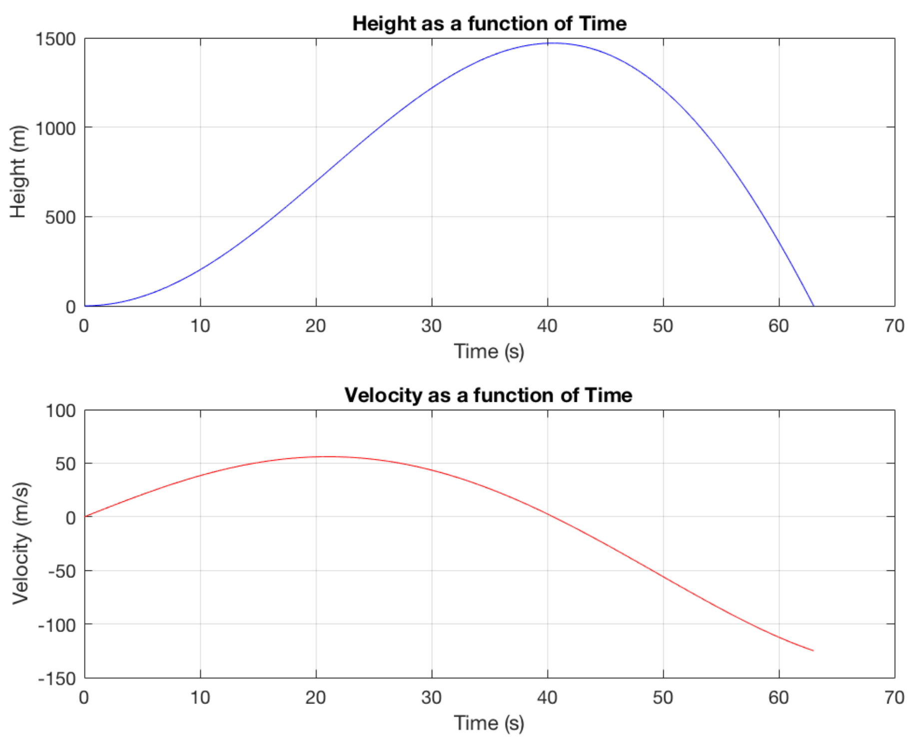
```
>> f=@R_motion

f =

  function_handle with value:

    @R_motion

>> f(60) %Test the function handle with time=60
Height=358.399488 (m)
Speed=112.317734 (m/s)
```

c)
Function

```
%This function plot height and velocity as a function of time from t=0 to
%t=63.01(the landing time)
%height is given by 2.13*time^2 - 0.0013*time^4 + 0.000034*time^4.751
%speed is given by the absolute value of the 1st order derivative of height with respect to time
function R_motionplot
time=0:0.01:63.01;

height=2.13.*time.^2 - 0.0013.*time.^4 + 0.000034.*time.^4.751;
```

```
velocity=4.26.*time - 0.0052.*time.^3 + 0.000161534.*time.^3.751;

s(1) = subplot(2,1,1);
height=2.13.*time.^2 - 0.0013.*time.^4 + 0.000034.*time.^4.751;
plot(time,height,'b')
grid
title('Height as a function of Time')
xlabel('Time (s)')
ylabel(s(1),'Height (m)')

s(2) = subplot(2,1,2);
velocity=4.26.*time - 0.0052.*time.^3 + 0.000161534.*time.^3.751;
plot(time,velocity,'r')
grid
title('Velocity as a function of Time')
xlabel('Time (s)')
ylabel(s(2),'Velocity (m/s)')
end
```

Output



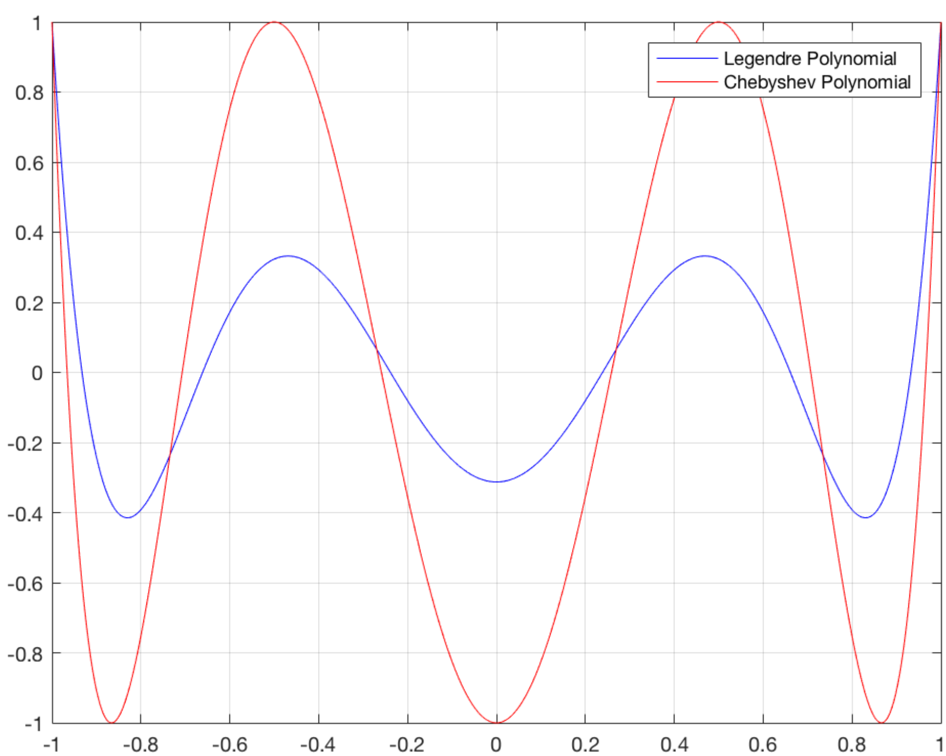
3.

```
x=-1:0.001:1;
LegendreP=(693.*x.^6-945.*x.^4+315.*x.^2-15)./48;
ChebyshevP=32.*x.^6-48.*x.^4+18.*x.^2-1;
fprintf('Roots\tLegendre\tChebyshev\n');
```

```
plot(x,LegendreP,'b',x,ChebyshevP,'r')
grid
legend('Legendre Polynomial','Chebyshev Polynomial')

for i=1:2000
    if LegendreP(i)*LegendreP(i+1)<0 %Product is negative if function crosses x-axis
        fprintf('\t%.3f\n',x(i));
    elseif ChebyshevP(i)*ChebyshevP(i+1)<0
        fprintf('\t\t\t%f.3\n',x(i));
    end
end
```

Output

```
Roots   Legendre        Chebyshev
                        -0.966000.3
        -0.933
                        -0.708000.3
        -0.662
                        -0.259000.3
        -0.239
        0.238
                        0.258000.3
        0.661
                        0.707000.3
        0.932
                        0.965000.3
```

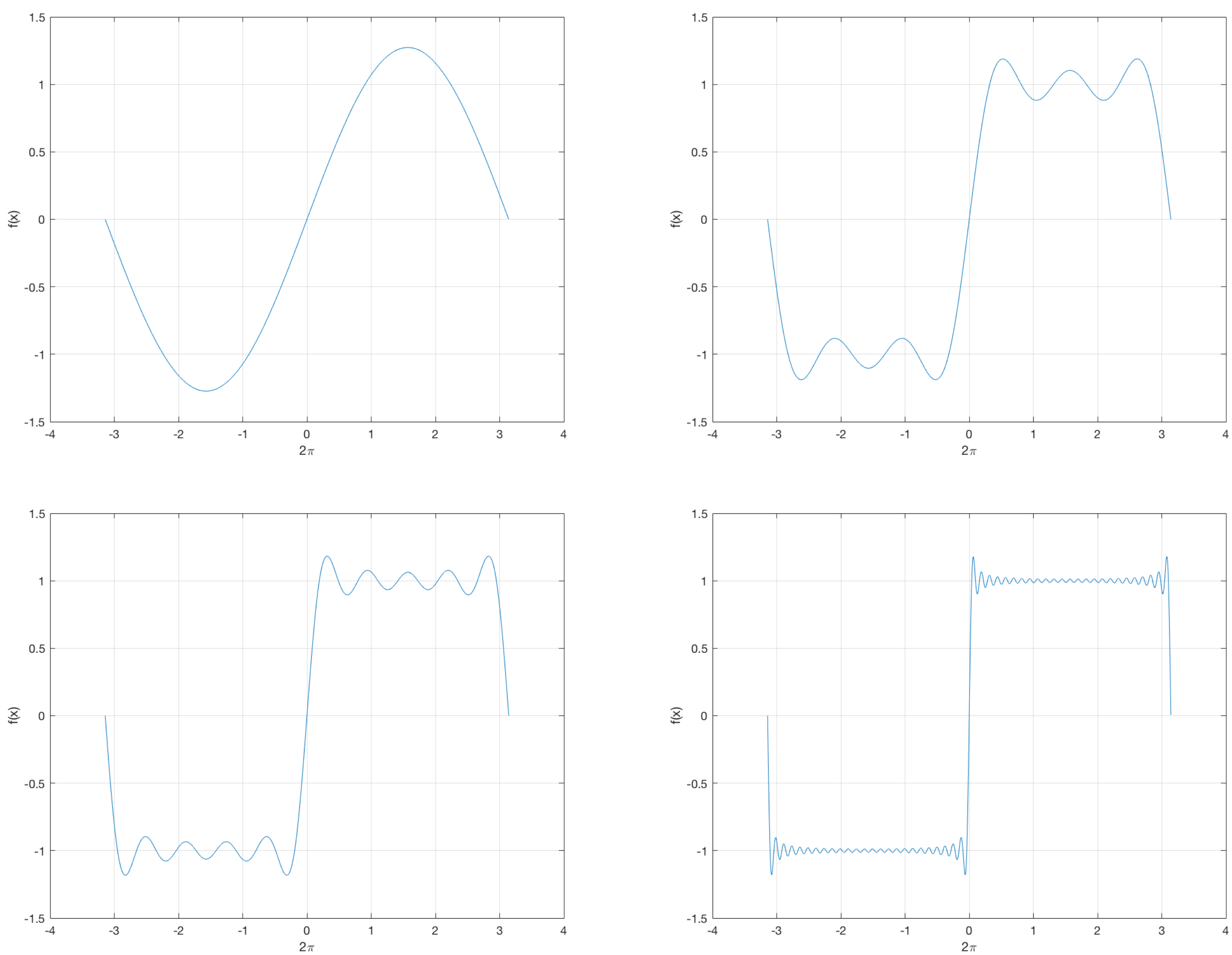


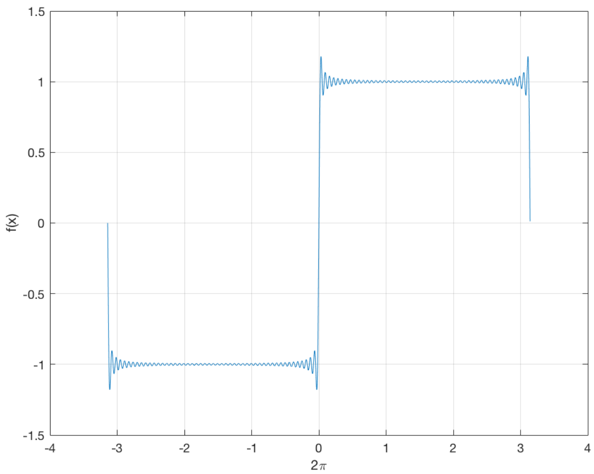
Both polynomials are oscillating and are symmetrical over the y-axis, thus the absolute values of the roots with the same distance of x from 0 are the same.

4.

```
k=1;
series = 0;
x = -pi:0.001:pi;

for n=1:100 %sum of series from n=1 to n=100
   series = series + sin(n.*x).*(2.*k.*(1-cos(n.*pi))./(n.*pi));
end

plot(x, series)
grid
xlabel('2\pi')
ylabel(‘f(x)')
```

Output N=2,5,10,50,100 respectively





As N becomes larger value the solution graph looks more like f(x) = -k when -pi<x<0 =k when 0<x<pi. The portion of the solution at k and -k in this case 1 and -1 becomes flatter as well. The biggest error is the glitch close to the ends of those flat lines, namely: around pi, 0, -pi.