PHYS 325 Assignment 4: Realistic Projectile Motion

Author: A. J. Ogle
Date: May 8th, 2015
PHYS 325 Computational Physics

1 Introduction

In the previous section, we showed how the Euler method was a useful computational method for solving 1st order differential equations. In this section, projectile motion will be approached from a more realistic standpoint. Using the Euler method, projectile motion with 2nd order differential equations is considered.

2 Background

Macroscopic objects behave according to Newton's Laws. In most introductory texts, important factors such as friction and air resistance are generally ignored. This is because 1st order differential equations are easier to solve than 2nd order, and the 1st order equations tend to give a sufficient enough idea for the general behavior of macroscopic objects. Nonetheless, it is known that falling objects experience some force proportional to their velocity. This force creates what is known as a terminal velocity, which is the maximum velocity that can be achieved by an object in free fall. This paper is concerned with modeling real world scenarios such as bicycle racing and the trajectories of cannon shells. For bicycle racing, the power exerted by the bicyclist can be rewritten in terms of the energy output over some interval of time. For a situation such as a flat road, where the energy is completely kinetic, the equation for power can be rewritten as

$$\frac{dv}{dt} = \frac{P}{m * v} \tag{1}$$

Where P is the power, m the mass, and v the velocity of the bicyclist.

The term for the air resistance can be written as

$$F_{drag} = \frac{-1}{2} * C * p * A * v^2 \tag{2}$$

C is the drag coefficient, and must be determined experimentally with the use of an air tunnel. A is the frontal area of the object which pushes against the air molecules as the object moves forward, and p is the density of the air. Using the Euler method, the algorithm for calculating the velocity after some time interval will be:

$$v_{i+1} = v_i + \frac{P}{m * v_i} * \delta * t - \frac{C * p * A * v_i^2}{2 * m} * \delta * t \tag{3}$$

The term due to the air resistance makes the equation 2nd order. As will be later shown, this is the term which develops the terminal velocity of the bicyclist, since it will increase much faster than the term for power as velocity increases.

In a more complicated case, the trajectory of cannon shells in 2 dimensions presents a more complex problem. In this case, the shell is fired with some initial velocity, after which it is acted on by the gravitational and drag force. Accounting for these two forces produces a more accurate prediction for where the shell will land. This is important when firing shells, because they are heavy and could hurt someone if not aimed properly. For this case, the forces are decomposed into x and y components. The force due to gravity acts only in the y direction, so the equation of motion for x is more simple than y. For the Euler method, the kinematic equations are written as:

$$x_{i+1} = x_i + v_{x,i} * \delta * t \tag{4}$$

$$v_{x,i+1} = v_{x,i} - \frac{B_2 * v * v_{x,i}}{m} * \delta * t \tag{5}$$

$$y_{i+1} = y_i + v_{y,i} * \delta * t \tag{6}$$

$$v_{y,i+1} = v_{y,i} - g * \delta * t - \frac{B_2 * v * v_{x,i}}{m} * \delta * t \tag{7}$$

For shells fired sufficiently high, it is possible that two factors may alter the shell's trajectory. Gravity and air density are both variables of height. It is possible that by accounting for these two factors, an even more accurate model for the trajectory of the shells can be made. Assuming an ideal gas, the density can be determined as a function of the air density at sea level and the ratio of the projectile height from that of sea level:

$$p = p_0 * e^(\frac{-y}{y_0}) \tag{8}$$

The acceleration due to gravity also varies according to height, and can be written as:

$$g = \frac{G * M_e}{(y + r_0)^2} \tag{9}$$

G is the gravitational constant, M_e the mass of the earth, y the height of the projectile, and r_0 the distance from the center of the earth to sea level.

It will later be explored to what extent these factors affect the trajectory of the shell.

3 Problem 2.1: Bicycle Velocity Without Air Resistance

To begin, the velocity of a bicycle moving without air resistance will be found numerically and analytically. The two results are plotted in the following figure.

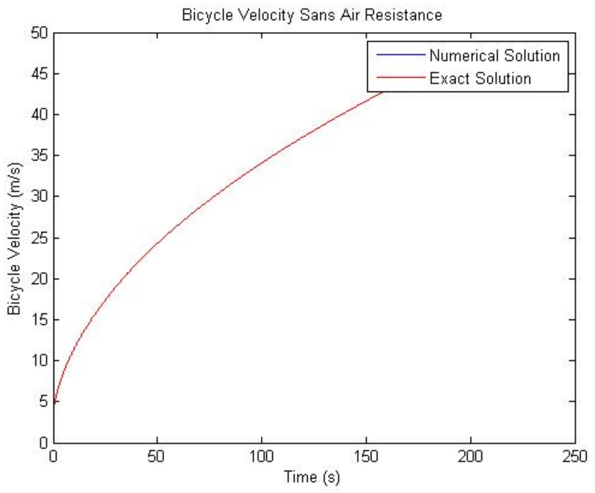

Figure 1: The analytic and numerical results showing the velocity of a bicycle with constant power. The mass was 70kg, with initial velocity 4 m/s and a time step of 0.1 s

As can be seen, the numerical and analytic results agree completely, since the two lines are indistinguishable in the figure. The figure shows that as time goes on, the velocity of the bicyclist increases indefinitely. This is a situation which is unrealistic, of course. The force due to air drag has not yet been considered.

4 Problem 2.2: Effects of Frontal Area and Power on Velocity of Bicyclist

Considering a more realistic situation which incorporates the drag force, the effects of varying frontal area and power on the velocity of the bicyclist are explored. It is expected that a greater frontal area would increase the drag force, since there is more surface area pushing against air molecules. An increase in power should result in a higher possible velocity, since this is what determines the acceleration of the bicyclist.

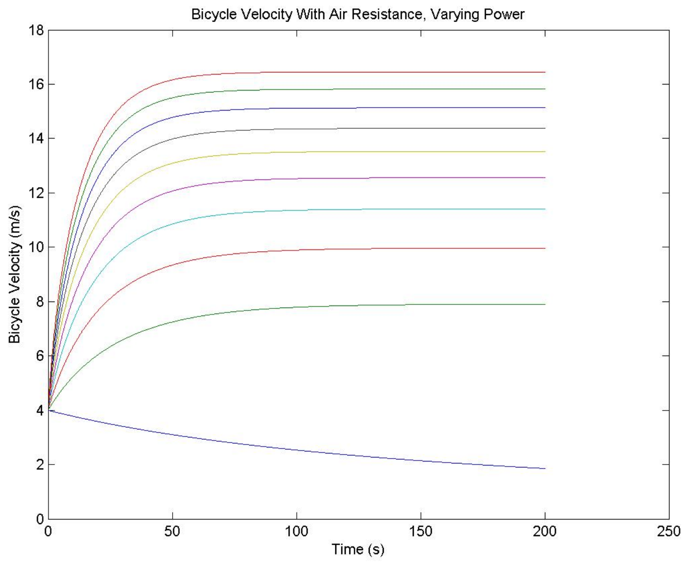

Figure 2: The effects of varying power on the velocity of the bicyclist. The maximum power is 400 W, and all subsequent lines below it are in increments of 50 W. The greater the power, the less the spacing between the velocity lines.

As predicted, a greater power results in a greater possible velocity, even with wind resistance. It is interesting to note that the difference between terminal velocity increases less the greater the power. This indicates even a limit for which power is able to propel the bicyclists, due to the drag force term of **??**.

Looking at figure 3, we see that varying the frontal area also affects the velocity of the bicyclist, but in a way opposite to that of power. By increasing the frontal area of the bicyclist, the terminal velocity is decreased, as expected. With no frontal area, the bicyclist is able to increase their velocity indefinitely. This was the case for the model in problem 1.1.

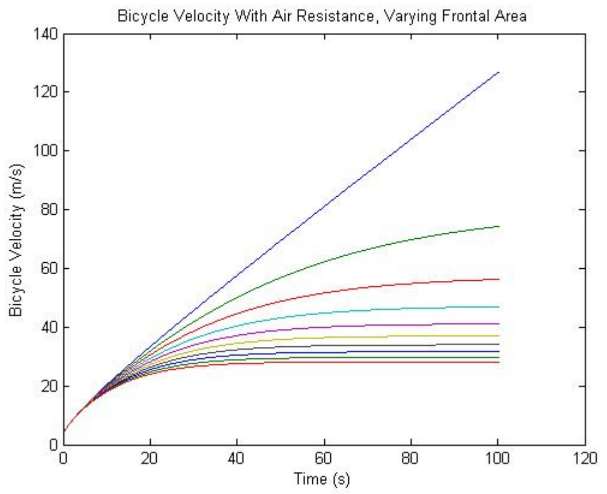

Figure 3: The effects of varying frontal area on the velocity of the bicyclist. The top most line is the velocity of the bicyclist without any frontal area, which effectively removes the drag force term. The bottom most line is the velocity of the bicyclist with the most frontal area, A = 0.33 m^2

5 Problem 2.4: Bicyclists Through Mountainous Terrain

The case for which a bicyclist is either riding up or down a hill is now considered. The only part that changes in the consideration of the forces becomes how the force components are decomposed, since there will be some offset due to the angle θ of the terrain.

For bicyclists traveling downhill, the downward angle allows more of the gravitational force to add to the velocity in the x direction. The following figure summarizes the effects of varying power for downhill terrain.

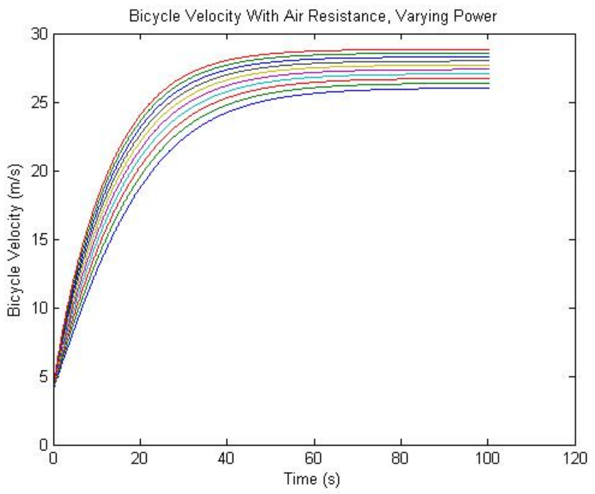

Figure 4: The effects of varying power on the velocity of a bicyclist going downhill. The maximum velocity of $31\ \frac{m}{s}$ was achieved with a downward angle of $\theta = 0.111$. This θ produces a velocity similar to that achieved by some professional bicyclists. The frontal area, A, was 0.33. Maximum power was 400 W, decreasing in intervals of 50 W.

For a bicyclist moving downhill, the terminal velocity is increased overall. This is because it is easier to travel downhill. Even for a bicyclist outputting no power (the lowest line in the figure), their velocity is increased due simply to gravitational potential.

Conversely, we expect that for a bicyclist moving uphill, the achievable velocity is decreased for all levels of power. Looking at the figure 5, we see that this is indeed the case.

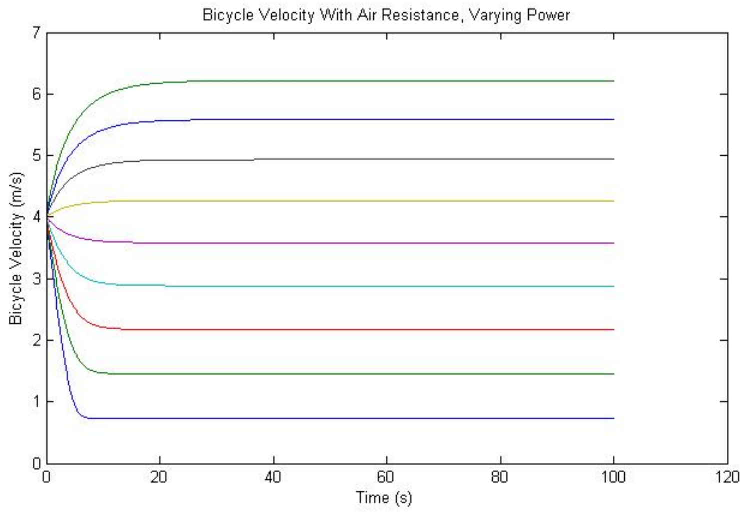

Figure 5: The effects of varying power on the velocity of a bicyclist going uphill. The maximum velocity of $6\ \frac{m}{s}$ was achieved with a downward angle of $\theta = -0.111$.

As expected, it is harder for bicyclists to move fast while going uphill, since most of their power must be used to push against the force of gravity.

6 Problem 2.8: Shell Trajectories

Newton's equations of motion help us to determine the trajectory of objects well. Without accounting for all of the forces and variables in the equations of motion, however, the equations can give unrealistic results. This was the case in problem 1.1, where the bicyclist was able to increase their velocity without bounds. The effects of varying gravity and air density will now be considered in the context of shell trajectories. Gravity varies as with the distance between two bodies of mass. The closer masses are to each other, the stronger the gravitational force. This is part of why the air density is non-uniform in the earth's atmosphere, as the air furthest away from the center of mass is least bound. To begin, a program was written which calculates the trajectory of a shell at different firing angles, θ. After running the simulation without making gravity a variable with respect to height, figure 6 was found.

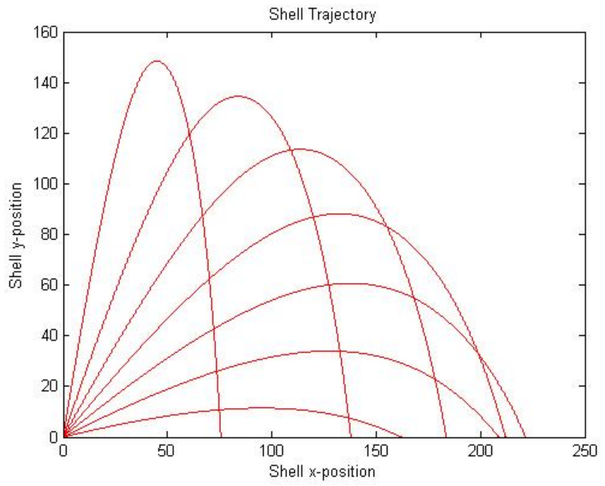

Figure 6: Trajectories of a shell fired at varying angles with constant gravity. These results were simultaneously compared with that of the figure with gravity and were found to be nearly the same. The effect of gravity accumulated a maximum of $\delta x = 1.2988 * 10^-13$ from all data points. It is safe to assume that for a typical shell, the effect due to changing gravity is minimal.

When investigating the effects of varying air density, a much greater difference was seen.

7 Problem 2.9: Effects of Varying Air Density According to Height

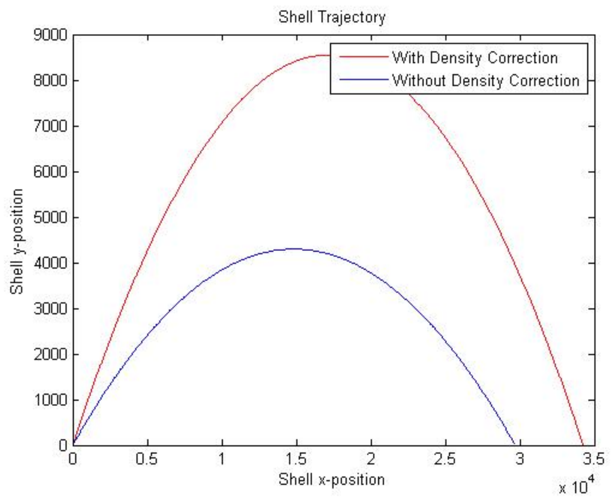

Figure 7: Trajectories of a fired shell both with and without air density correction. With air density correction, the shell was found to travel much farther. With air density correction, the maximum achievable distance was found to be $x_{max} = 3.4260 * 10^4$. Without air density correction, $x_{max} = 2.9643 * 10^4$. This represents a $15.58 percent$ increase in the x_{max}. The initial velocity was 700 $\frac{m}{s}$.

After accounting for the air density, the maximum achievable distance of the shell was increased by $15.58 percent$, with an initial velocity of 700 $\frac{m}{s}$ each trial. The angle $\theta = \frac{pi}{4.01}$ was found to give the greatest distance for either case. As the projectile is shot higher into the atmosphere, the air thins, thus reducing the drag force and allowing the shell to travel further. For shells that travel significantly high (greater than 3 km), this effect is pronounced, allowing a shell to travel almost an entire kilometer more than a shell traveling through uniform air density.

8 Problem 2.10: Firing Shells to Locations at Different Altitudes

As previously shown, a shell is able to travel farther given non-uniformly dense air. For someone trying to fire a shell at a specific location, this factor becomes important. For this section, the model is extended further to incorporate a target height, stopping the simulation once the fired shell has dropped on to the desired height. This allows users of the simulation to determine a few things. The user will be able to tell the angle, initial velocity, and the maximum possible distance that a target of a specific height can be hit. A target height of 1km and 4km is specified in **??** and **??**, respectively.

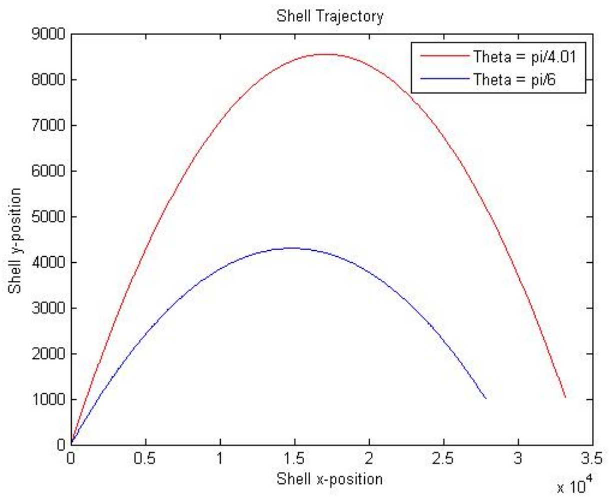

Figure 8: A shell is fired at a target with a height of 1km. Two different angles are shown, $\theta = pi/4.01$ and $\theta = pi/6$. These angles give the maximum possible distances for the two different models. For $\theta = \pi/4.01$, aid-density is considered variable. For $\theta = \pi/6$, air density is considered uniform.

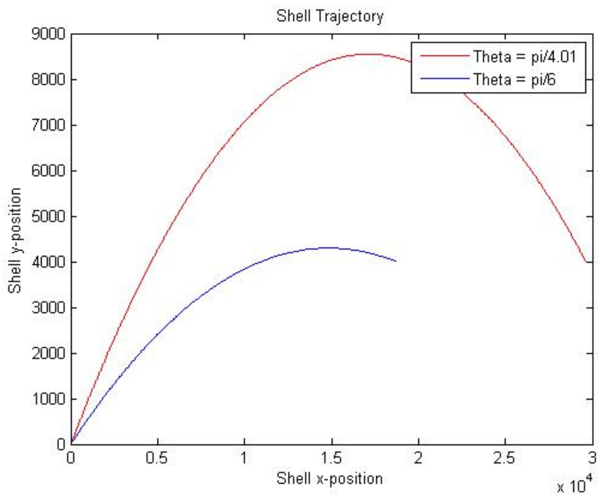

Figure 9: A shell is fired at a target with a height of 4km. Two different angles are shown, $\theta = \pi/4.01$ and $\theta = \pi/6$. The percent difference between the achievable distance for a target at this height for the two angles is $\delta_x = 58.312 percent$. The angle of the projectile for $\theta = \pi/4.01$ still provides the maximum distance, as expected.

Once again, a model which incorporates air-density as a variable produces a result with $\delta_x = 58.312 percent$, a significant difference for such applications as military.

9 Conclusions

When modeling realistic physical situations, it is important to account for such factors as friction and air resistance. These are the factors which produce such phenomena as terminal velocities. Not accounting for these forces in kinematic equations can lead to differences in projections of $\delta_x = 58.312 percent$, a significant amount of error when firing a shell at a target 1 km away. For the situation of the bicyclists, accounting for air friction results in a terminal velocity, not allowing bicyclists to continue accelerating to infinity. Although 2nd order differential equations are more complicated, they are more representative of the tangible phenomena in our life. For situations such as projectile motion, Euler's method suffices to predict the behavior of the system.

References

[1] Giordano, Nicholas J., and Hisao Nakanishi. "2. Realistic Projectile Motion" Computational Physics. Upper Saddle River, NJ: Pearson/Prentice Hall, 2006. N. pag. Print.

[2] Barker, Christopher A. "Euler's Method." Numerical Methods–Euler's Method. San Joaquin Delta College, 2009. Web. 28 Feb. 2015. ¡http://calculuslab.deltacollege.edu/ODE/7-C-1/7-C-1-h-b.html¿.

10 MATLAB Code

```
clear all;
%% Problems 2.1, 2.2, and 2.4

%Run with or without air resistance? 1 is with and 0 is without
air = 1;



%Define initial variables
dt = 0.1; %time interval, s
ft = 100; %size of the time interval desired, s
P = 400; %Power of elite bicyclist, W
v_i = 4; %initial velocity of the bicyclist, m/s
m = 70; %mass of bicyclist and bicycle, kg
C = 0.5; %air resistance constant
p = 1.225; %density of air, kg/m^3
A_n = 0.33; %frontal area, m^2
A_mid = 0.231; %frontal area for rider in the middle of a pack, m^2
theta = 0.111;

%Create an array for the time variable (independent variable)
t = (0.0:dt:ft+dt); %0 to ft seconds
%Create an array for the velocity variable (dependent variable)
v(1) = v_i;
vt(1) = v_i;

if(~air)
%Loop for calculating the change in velocity of the bicycle
for i = 2:length(t)
      v(i) = v(i-1) + (P/(m*v(i-1)))*dt;
      vt(i) = (v(i-1)^2 + (2*P*dt)/m)^(0.5) + 1;
end

clc
disp('Simulation 2.1 completed.');

clf
plot(t,v,'-');
hold on;
plot(t,vt,'-r');
hold on;
legend('Numerical Solution','Exact Solution');
xlabel('Time (s)')
ylabel('Bicycle Velocity (m/s)')
title('Bicycle Velocity Sans Air Resistance')

else
```

```
%Initial conditions, 5 different powers, 5 different frontal areas
dp = 50;
da = 0.04125;
v_p1(1) = v_i;
v_p2(1) = v_i;
v_p3(1) = v_i;
v_p4(1) = v_i;
v_p5(1) = v_i;
v_p6(1) = v_i;
v_p7(1) = v_i;
v_p8(1) = v_i;
v_p9(1) = v_i;
v_p10(1) = v_i;

P_t = (0:dp:P+dp);
A_t = (0:da:A_n+dt);
%Loop for varying power and frontal area, with air resistance
for j = 1:length(A_t)
for i = 2:length(t)
       v_p1(i) = v_p1(i-1) + (P/(m*v_p1(i-1)))*dt + 9.8*sin(theta)*dt - ((C*p*A_t(1)*
end
for i = 2:length(t)
       v_p2(i) = v_p2(i-1) + (P/(m*v_p2(i-1)))*dt + 9.8*sin(theta)*dt - ((C*p*A_t(2)*
end
for i = 2:length(t)
       v_p3(i) = v_p3(i-1) + (P/(m*v_p3(i-1)))*dt + 9.8*sin(theta)*dt - ((C*p*A_t(3)*
end
for i = 2:length(t)
       v_p4(i) = v_p4(i-1) + (P/(m*v_p4(i-1)))*dt + 9.8*sin(theta)*dt - ((C*p*A_t(4)*
end
for i = 2:length(t)
       v_p5(i) = v_p5(i-1) + (P/(m*v_p5(i-1)))*dt + 9.8*sin(theta)*dt - ((C*p*A_t(5)*
end
for i = 2:length(t)
       v_p6(i) = v_p6(i-1) + (P/(m*v_p6(i-1)))*dt + 9.8*sin(theta)*dt - ((C*p*A_t(6)*
end
for i = 2:length(t)
       v_p7(i) = v_p7(i-1) + (P/(m*v_p7(i-1)))*dt + 9.8*sin(theta)*dt - ((C*p*A_t(7)*
end
for i = 2:length(t)
       v_p8(i) = v_p8(i-1) + (P/(m*v_p8(i-1)))*dt + 9.8*sin(theta)*dt - ((C*p*A_t(8)*
end
for i = 2:length(t)
       v_p9(i) = v_p9(i-1) + (P/(m*v_p9(i-1)))*dt + 9.8*sin(theta)*dt - ((C*p*A_t(9)*
end
for i = 2:length(t)
       v_p10(i) = v_p10(i-1) + (P/(m*v_p10(i-1)))*dt + 9.8*sin(theta)*dt - ((C*p*A_t(
end
end
```

```
clc
disp('Simulation 2.2 completed');
clf
plot(t,v_p1,t,v_p2,t,v_p3,t,v_p4,t,v_p5,t,v_p6,t,v_p7,t,v_p8,t,v_p9,t,v_p10);
hold on;
xlabel('Time (s)')
ylabel('Bicycle Velocity (m/s)')
title('Bicycle Velocity With Air Resistance, Varying Frontal Area')

end

%% Problems 2.8, 2.9, 2.10

%Define initial variables
dt = 0.1; %time interval, s
ft = 10; %size of the time interval desired, s
G = 6.67384e-11; %gravitational constant
Me = 5.972e24; %mass of the earth
theta = pi/4; %angle at which the shell is fired
p_i = 0; %initial position of the shell
p_ix = p_i*cos(theta); %component forms
p_iy = p_i*sin(theta); %component forms
v_i = 100; %initial velocity of the shell
v_ix = v_i*cos(theta); %velocity in x direction
v_iy = v_i*sin(theta); %velocity in y direction
m = 70; %mass of shell
C = 0.5; %air resistance constant
p = 1.225; %density of air, kg/m^3
A_n = 0.33; %frontal area, m^2


%Create an array for the time variable (independent variable)
t = (0.0:dt:ft+dt); %0 to ft seconds
%Create an array for the velocity variable (dependent variable)
p(1) = p_i;
p_x(1) = p_ix;
p_y(1) = p_iy;
v(1) = v_i;
v_x(1) = v_ix;
v_y(1) = v_iy;

%Loop for calculating the change in velocity of the bicycle
for i = 2:length(t)
      p_x(i) = p_x(i-1) + v_x(i-1)*dt;
      p_y(i) = p_y(i-1) + v_y(i-1)*dt;

      v_x(i) = v_x(i-1) - ((C*v(i-1)*v_x(i-1))/m)*dt;
      v_y(i) = v_y(i-1) - ((G*Me)/(p_y(i-1) + 6371000))*dt - ((C*v(i-1)*v_y(i-1))/m)*
      v(i) = sqrt(v_x(i)^2 + v_y(i)^2);
      if(p_y <= 0)
          i = length(t);
      end
end
clc
disp('Simulation 2.8 completed.');

clf
plot(p_x,p_y,'-');
hold on;
legend('Trajectory 1');
xlabel('Time (s)')
ylabel('Bicycle Velocity (m/s)')
title('Bicycle Velocity Sans Air Resistance')

%%Problem 2.8
%% Problems 2.8

clear all;

%Define initial variables
dt = 0.0001; %time interval, s
ft = 20; %size of the time interval desired, s
G = 6.67384e-11; %gravitational constant
Me = 5.972e24; %mass of the earth
theta = 45; %angle at which the shell is fired
p_i = 0; %initial position of the shell
p_ix = p_i*cos(theta); %component forms
p_iy = p_i*sin(theta); %component forms
v_i = 100; %initial velocity of the shell,m/s
v_ix = v_i*cos(theta);%v_i*cos(theta); %velocity in x direction
v_iy = v_i*sin(theta);%v_i*sin(theta); %velocity in y direction
m = 70; %mass of shell
B = 0.47; %drag coefficient of sphere
g = 9.8; % acceleration due to gravity, m/s^2
O = 0:(pi/16):(pi/2);
color = ['y','m','c','r','g','b','w','k'];
grav_diff = 0;
grav_stand_dev = 0;

%Create an array for the time variable (independent variable)
t = (0.0:dt:ft+dt); %0 to ft seconds

clf
%Loop calculating the change in velocity of the bicycle
for j = 2:length(O)
%Create an array for the velocity variable (dependent variable)
```

```
p_ix = p_i*cos(O(j)); %component forms
p_iy = p_i*sin(O(j)); %component forms
v_ix = v_i*cos(O(j));%velocity in x direction
v_iy = v_i*sin(O(j));%velocity in y direction
p(1) = p_i;
p_x(1) = p_ix;
p_y(1) = p_iy;
v(1) = v_i;
v_x(1) = v_ix;
v_y(1) = v_iy;
check(1) = p_i + v_i;

for i = 2:length(t)
      p_x(i) = p_x(i-1) + v_x(i-1)*dt;
      p_y(i) = p_y(i-1) + v_y(i-1)*dt;

      v(i) = sqrt(v_x(i-1)^2 + v_y(i-1)^2);

      v_x(i) = v_x(i-1) - ((B*v(i-1)*v_x(i-1))/m)*dt;
      v_y(i) = v_y(i-1) - g*dt - ((B*v(i-1)*v_y(i-1))/m)*dt; %use
      %((G*Me)/(p_y(i-1) + 6371000))*dt for gravity wrt altitude
      check(i) = p_y(i) + v_y(i)*dt;

      if(check(i) <= 0)
          break
      end

end

for k = length(O)
plot(p_x,p_y,'-r');
hold on;
end

end


for j = 2:length(O)
%Create an array for the velocity variable (dependent variable)

p_ix = p_i*cos(O(j)); %component forms
p_iy = p_i*sin(O(j)); %component forms
v_ix = v_i*cos(O(j));%velocity in x direction
v_iy = v_i*sin(O(j));%velocity in y direction
p(1) = p_i;
p_xg(1) = p_ix;
p_yg(1) = p_iy;
v(1) = v_i;
v_x(1) = v_ix;
v_y(1) = v_iy;
```

```
check(1) = p_i + v_i;

for i = 2:length(t)
      p_xg(i) = p_xg(i-1) + v_x(i-1)*dt;
      p_yg(i) = p_yg(i-1) + v_y(i-1)*dt;

      v(i) = sqrt(v_x(i-1)^2 + v_y(i-1)^2);

      v_x(i) = v_x(i-1) - ((B*v(i-1)*v_x(i-1))/m)*dt;
      v_y(i) = v_y(i-1) - ((G*Me)/((p_yg(i-1) + 6371000)^2))*dt - ((B*v(i-1)*v_y(i-1)
      %((G*Me)/((p_yg(i-1) + 6371000)^2))*dt for gravity wrt altitude
      check(i) = p_yg(i) + v_y(i)*dt;

      if(check(i) <= 0)
          break
      end
end

for k = length(O)
plot(p_x,p_y,'-r');
hold on;
end

end

for h = 1:length(p_xg)
    diff = p_x(i) - p_xg(i);
    grav_diff = grav_diff + diff;
end

clc
xlabel('Shell x-position')
ylabel('Shell y-position')
title('Shell Trajectory')
disp(grav_diff);

disp('Simulation 2.8 completed.');

%%Problem 2.9
%% Problems 2.9

clear all;

%Define initial variables
obj_alt = 10000;
above_alt = false;
miss_1 = true;
miss_2 = true;
dt = 0.1; %time interval, s
ft = 300000; %size of the time interval desired, s
```

```
G = 6.67384e-11; %gravitational constant
Me = 5.972e24; %mass of the earth
theta = pi/4.01; %angle at which the shell is fired
rho_i = 1.225; %kg/(m^3)
p_i = 0.0000001; %initial position of the shell
p_ix = p_i*cos(theta); %component forms
p_iy = p_i*sin(theta); %component forms
v_i = 700; %initial velocity of the shell,m/s
v_ix = v_i*cos(theta);%v_i*cos(theta); %velocity in x direction
v_iy = v_i*sin(theta);%v_i*sin(theta); %velocity in y direction
m = 70; %mass of shell
B = 0.47; %drag coefficient of sphere
g = 9.8; % acceleration due to gravity, m/s^2
O = 0:(theta):(theta);
color = ['y','m','c','r','g','b','w','k'];
grav_diff = 0;
grav_stand_dev = 0;

%Create an array for the time variable (independent variable)
t = (0.0:dt:ft+dt); %0 to ft seconds

clf

for j = 2:length(O)
%Create an array for the velocity variable (dependent variable)

p_ix = p_i*cos(O(j)); %component forms
p_iy = p_i*sin(O(j)); %component forms
v_ix = v_i*cos(O(j));%velocity in x direction
v_iy = v_i*sin(O(j));%velocity in y direction
p(1) = p_i;
p_x(1) = p_ix;
p_y(1) = p_iy;
v(1) = v_i;
v_x(1) = v_ix;
v_y(1) = v_iy;
check(1) = p_i + v_i;
rho(1) = rho_i;

for i = 2:length(t)
      rho(i) = rho_i*(exp(-(p_y(i-1)/p_iy)));

      p_x(i) = p_x(i-1) + v_x(i-1)*dt;
      p_y(i) = p_y(i-1) + v_y(i-1)*dt;

      v(i) = sqrt(v_x(i-1)^2 + v_y(i-1)^2);

      v_x(i) = v_x(i-1) - (rho(i)/rho_i)*((B*v(i-1)*v_x(i-1))/m)*dt;
      v_y(i) = v_y(i-1) - g*dt - (rho(i)/rho_i)*((B*v(i-1)*v_y(i-1))/m)*dt; %use
      %((G*Me)/((p_yg(i-1) + 6371000)^2))*dt for gravity wrt altitude
      check(i) = p_y(i) + v_y(i)*dt;

      if(check(i) >= obj_alt)
          above_alt = true;
          miss_1 = false;
      end
      if(above_alt && check(i) <= obj_alt || check(i) <= 0)
          break
      end


end

for k = length(O)
plot(p_x,p_y,'-r');
hold on;
end

end

%%Second Simulation with smaller theta

%Define initial variables
theta = (pi/6); %angle at which the shell is fired
O = 0:(theta):(theta);
above_alt = false;

%Create an array for the time variable (independent variable)
t = (0.0:dt:ft+dt); %0 to ft seconds

for j = 2:length(O)
%Create an array for the velocity variable (dependent variable)

p_ix = p_i*cos(O(j)); %component forms
p_iy = p_i*sin(O(j)); %component forms
v_ix = v_i*cos(O(j));%velocity in x direction
v_iy = v_i*sin(O(j));%velocity in y direction
p(1) = p_i;
p_x2(1) = p_ix;
p_y2(1) = p_iy;
v(1) = v_i;
v_x(1) = v_ix;
v_y(1) = v_iy;
check(1) = p_i + v_i;
rho(1) = rho_i;

for i = 2:length(t)
      rho(i) = rho_i*(exp(-(p_y2(i-1)/p_iy)));

      p_x2(i) = p_x2(i-1) + v_x(i-1)*dt;
      p_y2(i) = p_y2(i-1) + v_y(i-1)*dt;

      v(i) = sqrt(v_x(i-1)^2 + v_y(i-1)^2);

      v_x(i) = v_x(i-1) - (rho(i)/rho_i)*((B*v(i-1)*v_x(i-1))/m)*dt;
      v_y(i) = v_y(i-1) - g*dt - (rho(i)/rho_i)*((B*v(i-1)*v_y(i-1))/m)*dt; %use
      %((G*Me)/((p_yg(i-1) + 6371000)^2))*dt for gravity wrt altitude
      check(i) = p_y2(i) + v_y(i)*dt;

       if(check(i) >= obj_alt)
          above_alt = true;
          miss_2 = false;
      end
      if(above_alt && check(i) <= obj_alt || check(i) <= 0)
          break
      end

end

for k = length(O)
plot(p_x2,p_y2,'-b');
hold on;
end

end


clc
xlabel('Shell x-position')
ylabel('Shell y-position')
legend('With Density Correction','Without Density Correction');
title('Shell Trajectory')
max1 = max(p_x);
max2 = max(p_x2);
disp('Simulation 2.8 completed.');
disp('With density correction: ');
disp(max1);
if(miss_1)
    disp('Miss!')
else
    disp('Hit!')
end
disp('Without density correction: ');
disp(max2);
if(miss_2)
    disp('Miss!')
else
    disp('Hit!')
end
```

```
%%Problem 2.10

%% Problems 2.9

clear all;

%Define initial variables
obj_alt = 1000;
above_alt = false;
miss_1 = true;
miss_2 = true;
dt = 0.1; %time interval, s
ft = 300000; %size of the time interval desired, s
G = 6.67384e-11; %gravitational constant
Me = 5.972e24; %mass of the earth
theta = pi/4.01; %angle at which the shell is fired
rho_i = 1.225; %kg/(m^3)
p_i = 0.0000001; %initial position of the shell
p_ix = p_i*cos(theta); %component forms
p_iy = p_i*sin(theta); %component forms
v_i = 700; %initial velocity of the shell,m/s
v_ix = v_i*cos(theta);%v_i*cos(theta); %velocity in x direction
v_iy = v_i*sin(theta);%v_i*sin(theta); %velocity in y direction
m = 70; %mass of shell
B = 0.47; %drag coefficient of sphere
g = 9.8; % acceleration due to gravity, m/s^2
O = 0:(theta):(theta);
color = [’y’,’m’,’c’,’r’,’g’,’b’,’w’,’k’];
grav_diff = 0;
grav_stand_dev = 0;

%Create an array for the time variable (independent variable)
t = (0.0:dt:ft+dt); %0 to ft seconds

clf

for j = 2:length(O)
%Create an array for the velocity variable (dependent variable)

p_ix = p_i*cos(O(j)); %component forms
p_iy = p_i*sin(O(j)); %component forms
v_ix = v_i*cos(O(j));%velocity in x direction
v_iy = v_i*sin(O(j));%velocity in y direction
p(1) = p_i;
p_x(1) = p_ix;
p_y(1) = p_iy;
v(1) = v_i;
v_x(1) = v_ix;
v_y(1) = v_iy;
check(1) = p_i + v_i;
```

```
rho(1) = rho_i;

for i = 2:length(t)
      rho(i) = rho_i*(exp(-(p_y(i-1)/p_iy)));

      p_x(i) = p_x(i-1) + v_x(i-1)*dt;
      p_y(i) = p_y(i-1) + v_y(i-1)*dt;

      v(i) = sqrt(v_x(i-1)^2 + v_y(i-1)^2);

      v_x(i) = v_x(i-1) - (rho(i)/rho_i)*((B*v(i-1)*v_x(i-1))/m)*dt;
      v_y(i) = v_y(i-1) - g*dt - (rho(i)/rho_i)*((B*v(i-1)*v_y(i-1))/m)*dt; %use
      %((G*Me)/((p_yg(i-1) + 6371000)^2))*dt for gravity wrt altitude
      check(i) = p_y(i) + v_y(i)*dt;

      if(check(i) >= obj_alt)
          above_alt = true;
          miss_1 = false;
      end
      if(above_alt && check(i) <= obj_alt || check(i) <= 0)
          break
      end


end

for k = length(O)
plot(p_x,p_y,'-r');
hold on;
end

end

%%Second Simulation with smaller theta

%Define initial variables
theta = (pi/6); %angle at which the shell is fired
O = 0:(theta):(theta);
above_alt = false;

%Create an array for the time variable (independent variable)
t = (0.0:dt:ft+dt); %0 to ft seconds

for j = 2:length(O)
%Create an array for the velocity variable (dependent variable)

p_ix = p_i*cos(O(j)); %component forms
p_iy = p_i*sin(O(j)); %component forms
v_ix = v_i*cos(O(j));%velocity in x direction
v_iy = v_i*sin(O(j));%velocity in y direction
```

```
p(1) = p_i;
p_x2(1) = p_ix;
p_y2(1) = p_iy;
v(1) = v_i;
v_x(1) = v_ix;
v_y(1) = v_iy;
check(1) = p_i + v_i;
rho(1) = rho_i;

for i = 2:length(t)
      rho(i) = rho_i %*(exp(-(p_y2(i-1)/p_iy)));

      p_x2(i) = p_x2(i-1) + v_x(i-1)*dt;
      p_y2(i) = p_y2(i-1) + v_y(i-1)*dt;

      v(i) = sqrt(v_x(i-1)^2 + v_y(i-1)^2);

      v_x(i) = v_x(i-1) - (rho(i)/rho_i)*((B*v(i-1)*v_x(i-1))/m)*dt;
      v_y(i) = v_y(i-1) - g*dt - (rho(i)/rho_i)*((B*v(i-1)*v_y(i-1))/m)*dt; %use
      %((G*Me)/((p_yg(i-1) + 6371000)^2))*dt for gravity wrt altitude
      check(i) = p_y2(i) + v_y(i)*dt;

       if(check(i) >= obj_alt)
          above_alt = true;
          miss_2 = false;
      end
      if(above_alt && check(i) <= obj_alt || check(i) <= 0)
          break
      end

end

for k = length(O)
plot(p_x2,p_y2,'-b');
hold on;
end

end



clc
xlabel('Shell x-position')
ylabel('Shell y-position')
legend('Theta = pi/4.01','Theta = pi/6');
title('Shell Trajectory')
max1 = max(p_x);
max2 = max(p_x2);
disp('Simulation 2.8 completed.');
disp('With density correction: ');
disp(max1);
```

```
if(miss_1)
    disp('Miss!')
else
    disp('Hit!')
end
disp('Without density correction: ');
disp(max2);
if(miss_2)
    disp('Miss!')
else
    disp('Hit!')
end
```