MAE 4020 PROJECT

Determining the Coefficient of a Tennis Ball Using Digital Image Processing and Numerical Methods.

5/02/2018

Abstract

Numerical methods were used to determine the coefficient of restitution of a tennis ball with two different methods. The first method consisted of manipulating the law of conservation of energy to attain the velocity of the tennis ball before and after impact. The second method utilized numerical differentiation to directly determine the velocity before and after impact. Once these velocities were obtained, the coefficient of restitution could be determined. Then, the drag coefficient of the tennis ball was numerically calculated to analyze the accuracy of the coefficient of restitution found from the law of conservation of energy. The findings of the project concluded that drag affected the first method, while the second method yielded a coefficient of restitution of 0.75 for the tennis ball.

Contents

Introduction

In this project, the coefficient of restitution will be determined by using a maximum height method and a maximum velocity method. The maximum height method will utilize the law of conservation of energy to calculate the maximum velocities before and after impact. This method could be inaccurate since the law of conservation of energy neglects air resistance. Therefore, the coefficient of drag will be calculated to determine if the tennis ball is affected by air resistance. If the coefficient of drag is inversely proportional to the error between the drag approximation and the position data, this method will be deemed inaccurate. The maximum velocity method will merely differentiate the position data to find the maximum velocity of the tennis ball. The maximum velocity will then be used to determine the velocities that occurred before and after impact. It is anticipated that this method will be more accurate and will yield a coefficient of restitution of approximately 0.8.

Method

The objective of this project was to determine the coefficient of restitution using numerical methods. This was done by recording a video of a tennis ball that was dropped at 1.75 feet (approximately 0.53 meter) and made multiple collisions with the ground. The video was then imported into MATLAB where it was then analyzed at each individual frame. Each frame was treated as a picture, turned into a binary form where the tennis ball was white and everything else was black, and then recorded the centroid of the ball with respect to the frame. By recording the centroids, the position of the tennis ball was obtained with respect to time. Once the position data was recorded, the digital imaging component of this project was complete.

To determine the velocity of the tennis ball for the first method of determining the coefficient of restitution, the position data needed to be numerically differentiate. Equation (1) shows how the position data was manipulated to achieve the velocity of the tennis ball.

(1)

$$\text{Velocity} = \frac{\text{y(t + dt) - y(t - dt)}}{\text{2 dt}}$$

To determine the coefficient of restitution with the maximum velocity method, Equation (2) needed to be used. The velocity before impact would be the velocity at the element immediately before the maximum velocity. The velocity after impact would be the velocity at the element immediately after the maximum velocity. These velocities were then substituted into Equation (2) to determine the coefficient of restitution.

(2)

$$COR = \frac{v_{after}}{v_{before}}$$

Although the coefficient of restitution was a ratio of two velocities, it was determined using the first two maximum heights of the tennis ball. A tenth order polynomial curve fit was used to smooth out the position data to get a more accurate result. The position data may not have contained the exact instance when the tennis ball reached its maximum height, but the curve fit helped to take this into consideration, possibly improving the accuracy. This height was translated into a velocity by utilizing the law of conservation of energy. Equation (3) shows how the maximum height was related to the velocities before and after impact.

(3)

$$v = \sqrt{2gh}$$

Equation (3) was also used to determine the velocity after impact. This was done by finding the maximum height immediately after the collision. However, Equation (2) was utilized in the script for this project. In the script, Equation (3) was substituted for the velocities in Equation (2). Equation (4) depicts this substitution.

(4)

$$COR = \sqrt{\frac{h_{after}}{h_{before}}}$$

Once Equation (4) was defined, both methods to determine the coefficient of restitution had been obtained. However, the accuracy of the two methods needed to be determined. The accuracy of the maximum height method was questioned since it wasn't known if air resistance affected the tennis ball. To determine the accuracy of this method, the acceleration was determined, using the central finite difference in Equation (5). Non-constant acceleration made it possible for drag effects to alter the results.

(5)

$$\text{Acceleration} = \frac{\text{y(t + dt) - 2y(t) - y(t - dt)}}{\text{dt}^2}$$

To determine the impact of drag for non-constant acceleration, a fourth order Runge-Kutta approximation was utilized. In the process of using the Runge-Kutta approximation, it was plotted against Euler's Method to ensure that it was scripted correctly into MATLAB. Equation (6) outlines the equation that was used to emulate drag.

(6)

$$\text{a} = \frac{1}{2}\frac{\rho\ \text{C}_\text{d}\ \text{A}\ \text{v}^2}{\text{m}} - \text{g}$$

To determine the impact that drag had on the tennis ball, the error between Equation (6) and the position data would be analyzed. The error calculations were conducted using Equation (7). This equation is also known as the residual sum of squares.

(7)

$$Error = \sqrt{\sum_{i=1}^{n}(x(t_i) - x_i)^2}$$

The error analyzed the Runge-Kutta approximation with and without drag. If the error between the approximation and position data was less with drag, then the maximum height method could be deemed inaccurate. This could be concluded because, when the error was less with drag, the position data correlated more with the data from drag. Thus, the maximum heights method was susceptible to air resistance, altering its results for the coefficient of restitution.

Results

The first result of the project can be seen in Figure 1. This figure shows the position of the tennis ball with respect to time. Every method of determining the coefficient of restitution utilized this set of data. Even the methods that determine the accuracy of each coefficient will be using the data displayed in Figure 1.

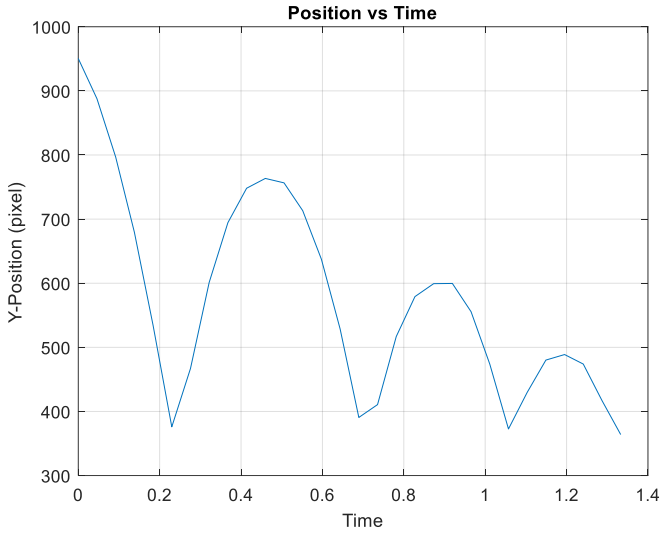


Figure 1: The Position of the Tennis Bal with Respect to Time.

The results from both methods of determining the coefficients are displayed in Table 1.

Table 1: The Coefficients of Restitutions for Both Methods.

COR using Maximum Velocities	COR using Maximum Heights
0.75	0.89

Figure 2 plotted the acceleration of the tennis ball with respect to time. Since the acceleration wasn't constant, the need for the fourth order Runge-Kutta approximation was outlined. It was believed that drag affected the tennis ball due to the results in Table 1. The coefficient of restitution using maximum heights yielding a greater coefficient of restitution than the maximum velocities method can be attributed to drag effects. Drag is directly correlated to the velocity of the tennis ball, which can be examined in Equation (6). The average velocity before impact for the tennis ball was greater than the average velocity after impact. Drag would then have a greater impact on the velocity before impact by decreasing it. This would decrease the denominator of the ratio for the coefficient of restitution. When the denominator of a fraction is decreased, it yields a larger value. Therefore, drag effects would result in a larger coefficient of restitution using the maximum heights method.

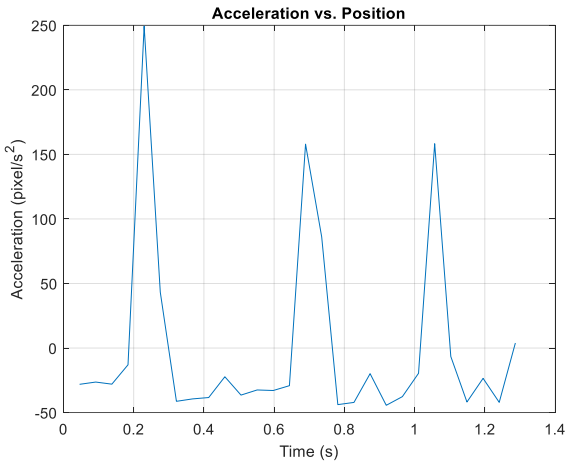


Figure 2: The Acceleration of the Tennis Ball with Respect to Time.

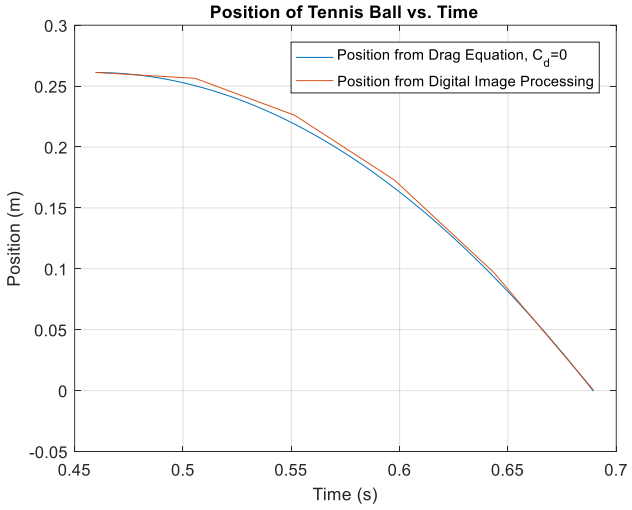


Figure 3: The Position of the Tennis Ball from the Data and from the Runge-Kutta Approximation without Drag.

Figure 3 showed the position data plotted against the fourth order Runge-Kutta approximation without drag, emulating free fall. This was plotted to see if there was a visual difference between the Runge-Kutta approximation with and without drag. A fourth order Runge-Kutta approximation with drag can be found in Figure 4.

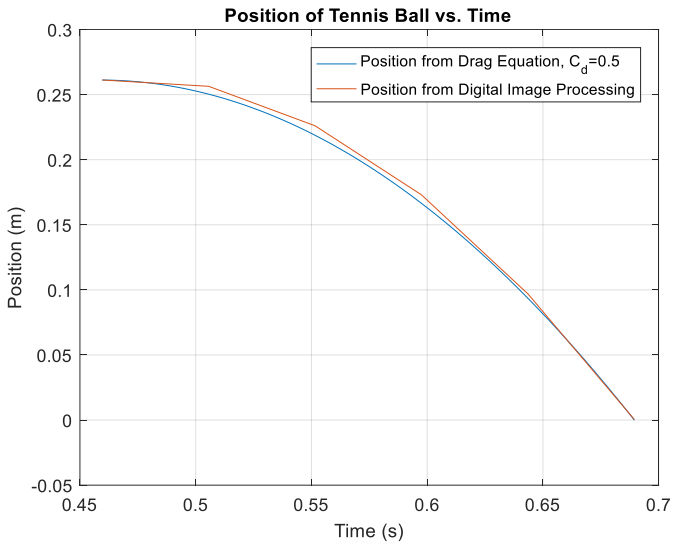


Figure 4: The Position of the Tennis Ball from the Data and Fourth Order Runge-Kutta Approximation with Drag.

Figure 4 showed there was no visual difference between the fourth order Runge-Kutta approximation with and without drag. However, Table 2 shows error between the Runge-Kutta approximations and the actual position data using Equation (7). This will numerically determine if increasing the drag coefficient decreases the error between the approximation and the position data.

Table 2: The Error for the fourth Order Runge Kutta Approximations.

Without Drag (C_d = 0)	With Drag (C_d = 0.5)
0.0121320 m	0.0120209 m

Since an increase in the drag coefficient decreases the error between the Runge-Kutta approximation and the position data, it can be concluded that drag affected the tennis ball. Therefore, the method of determining the coefficient of restitution using the maximum heights method can be deemed as inaccurate.

Conclusion

In conclusion, the coefficient of restitution of the tennis ball was 0.75. This value was obtained by plotting the position of a tennis ball and numerically differentiating with respect to time to find the velocity before and after impact. Other methods utilized maximum heights and the law of conservation of energy, but, due to drag affects (non-constant acceleration), were concluded inaccurate.

Appendices

```
% Trevor Gerken
% MAE4020_PROJECT

clear all;clc

%Inputting the video into Matlab
v = VideoReader('BouncyBouncyBouncy.mp4');

%Magnitude of color for the binary picture
Target = [160 160 40]; %Target RGB value098/

MagTar = sqrt((double(Target(1))^2)+(double(Target(2))^2)+(double(Target(3))^2));

%Organizing the frames of the video so that every frame is analyzed
A = v.Frame;
AA = floor(A);
l = 1;

%Inputting the video into pictures
Picframe = read(v);

%Looping through everyframe, turning it into a binary picture, and finding
%centroid of every binary picture
  for n = 1:AA
    I = Picframe(:,:,:,n);
Pic(:,:,l) = BinaryPic(I,MagTar);
Pic(:,1548:end,l) = 0;
Pic(883:end,600:end,l) = 0;
Pic(:,1:300,l) = 0;
y(l) = 1080-BinaryCentroid(Pic(:,:,l));
l = l+1;
  end

 %Defining the time vector, based off the length of the video, manually
 %inputted
 t = linspace(0,1.333,30);

%Position 1 is the estimated position of the max height, manually inputted
position1 = 11;

%Position 2 is the estimated position of the next max height,man. inputted
position2 = 20;

%u is the interval or confidence in your estimated position of the max
%Manually inputted
u = 3;
```

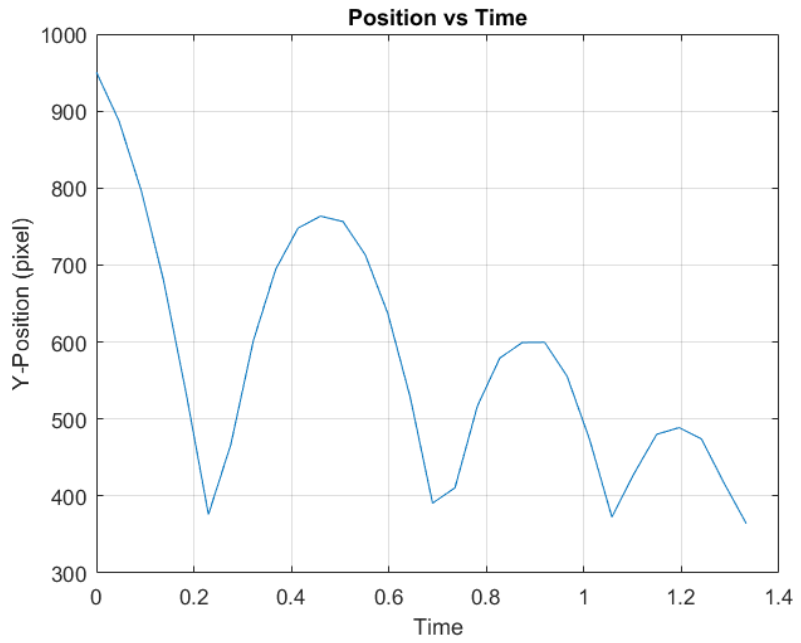
```
%first curvefit
curvedata1 = y((position1-u):(position1+u));
timedata1 = t((position1-u):(position1+u));
curve1 = tenthorderpoly(curvedata1,timedata1);

%Finding the maximum height in the first curvefit
[max1] = Maxval(curve1,timedata1);

%second curvefit
curvedata2 = y((position2-u):(position2+u));
timedata2 = t((position2-u):(position2+u));
curve2 = tenthorderpoly(curvedata2,timedata2);

%Finding the maximum height in the second curvefit
[max2] = Maxval(curve2,timedata2);

%Dermining height coefficient using the maximum heights
heightcoefficient = sqrt(max2/max1);

%Finding the velocity data using differentiaiton, special central finite
velocity = centraldif(y);

%Begin the process for determining resistution coefficient based on the
%maximum velocities
n = length(velocity);

%Finding maximum velocity
maxv = 0;
element = 1;
for nn = 23:n

    if velocity(nn)>maxv
        maxv = velocity(nn);
        element = nn;
    else
    end
end

%Find the velocity for the element before the max velocity
vb4collision = velocity(element-1);

%Find the velocity for the element after the max velocity
vaftercollision = velocity(element+1);

%Using the velocity before and after the max velocity, find the coefficient
%of restitution
velocitycoefficient = abs((vb4collision)/(vaftercollision));

%Determining the acceleration using numerical differentation
a = acceleration(y);
```

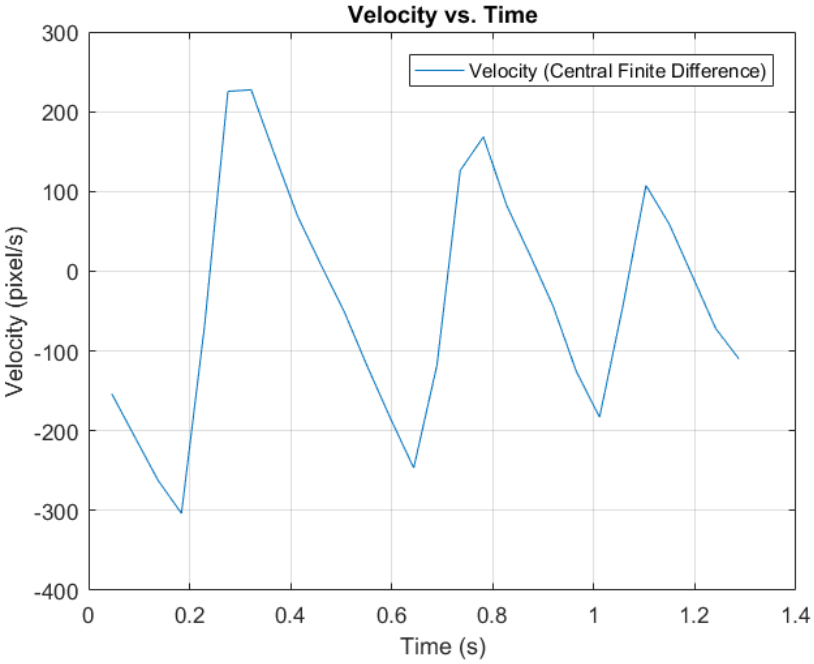
```
 %plotting position
 figure(1);clf;hold on;box on;grid on
 plot(t,y)
 title('Position vs Time')
 xlabel('Time')
 ylabel('Y-Position (pixel)')

 %Plotting the central finite difference velocity
figure(2);clf;hold on;box on;grid on
plot(t(2:end-1),velocity)
title('Velocity vs. Time')
xlabel('Time (s)')
ylabel('Velocity (pixel/s)')
legend('Velocity (Central Finite Difference)')

%Acceleration needs to be plotted to determine if it is constant

%With non-constant acceleration, the effects of air resistance might be
%magnfied and create inaccurate results.

%If acceleration is constant, the maximum heights method will be accurate.

%if acceleration is not constant, then the drag needs to be analyzed

%Plotting the accceleration
figure(3);clf;hold on;box on;grid on
plot(t(2:end-1),a)
title('Acceleration vs. Position')
xlabel('Time (s)')
ylabel('Acceleration (pixel/s^2)')

%Plotting the first curvefit
figure(4);hold on;box on;grid on
plot(timedata1,curve1)
title('The First Max Curve-Fit for the Position of the Ball')
xlabel('Time (s)')
ylabel('Height (pixel)')
legend('Tenth Order Polynomial')

%plotting the second curvefit
figure(5);hold on;box on;grid on
plot(timedata2,curve2)
title('The Second Max Curve-Fit for the Position of the Ball')
xlabel('Time (s)')
ylabel('Height (pixel')
ylim([400,700])
legend('Tenth Order Polynomial')


%Modelling the system with drag
```

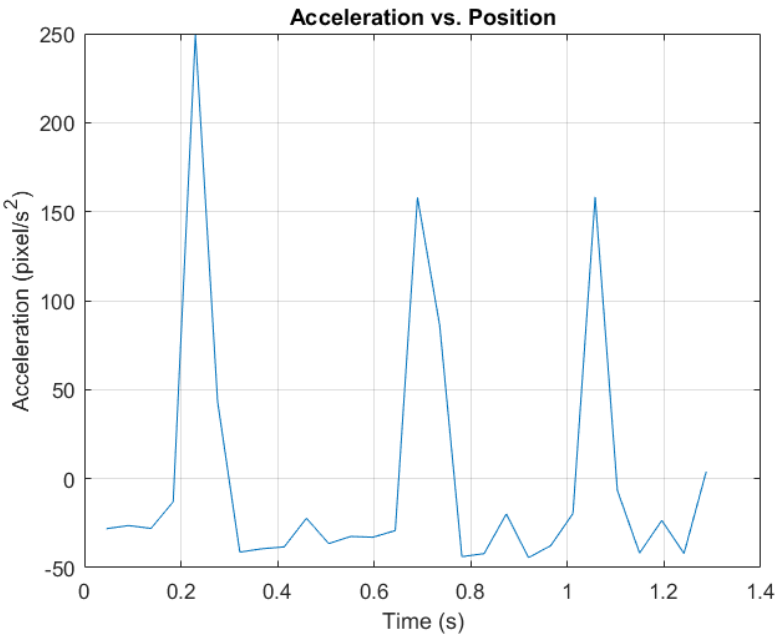
```
ywithunits = [y-390]*(1.75/2502);
ydraginterval = ywithunits(11:16);
tdraginterval = t(11:16);
tdrag = linspace(t(11),t(16),101);
ya = [y-390]*(1.75/2502);
y0 = ya(11);
yd0 = 0;
CD = 0.5;
yRK = RungeKutta4project(tdrag,y0,yd0,CD);
yEulers = EulersProject(tdrag,y0,yd0,CD);

%Plotting Runge Kutta and Eulers to ensure that the algorithm is correct
figure(6);clf;hold on;box on;grid on
plot(tdrag,yRK)
plot(tdrag,yEulers)
title('Position vs. Time From Drag')
xlabel('Time (s)')
ylabel('Position')
legend('4th Order Runge Kutta','Eulers')

%Find Position Data from drag with Runge Kutta, with a CD = 0.5
yRKCD0 = RungeKutta4project(tdrag,y0,yd0,0.5);

%Find Position Data from drag with Runge Kutta, with a CD = 0
yRKCDZero = RungeKutta4project(tdrag,y0,yd0,0);

%The first step of determining the residual sum squares between the
%position data and the approximations with and without drag.
aux1 = 1;
sumrunge = 0;
aux2 = 1;
sumRunge0 = 0;
for n = 1:6
    aux3 = ((yRKCD0(aux2)-ydraginterval(aux1))^2);
    aux4 = ((yRKCDZero(aux2)-ydraginterval(aux1))^2);
    aux1 = aux1 + 1;
    aux2 = aux2 +20;
    sumRunge0 = sumRunge0 +aux3;
    sumrunge = sumrunge +aux4;
end

%The second step of calculating the residual sum of squares between the
%position data and the approximation with and without drag
errorwithoutdrag = sqrt(sumrunge);
errorwithdrag = sqrt(sumRunge0);

%Plotting the Runge Kutta Approximation with CD = 0 vs position data
%to determine if drag affects the tennis ball.
figure(7);clf;hold on;box on;grid on
plot(tdrag,yRKCDZero)
plot(tdraginterval,ydraginterval)
```

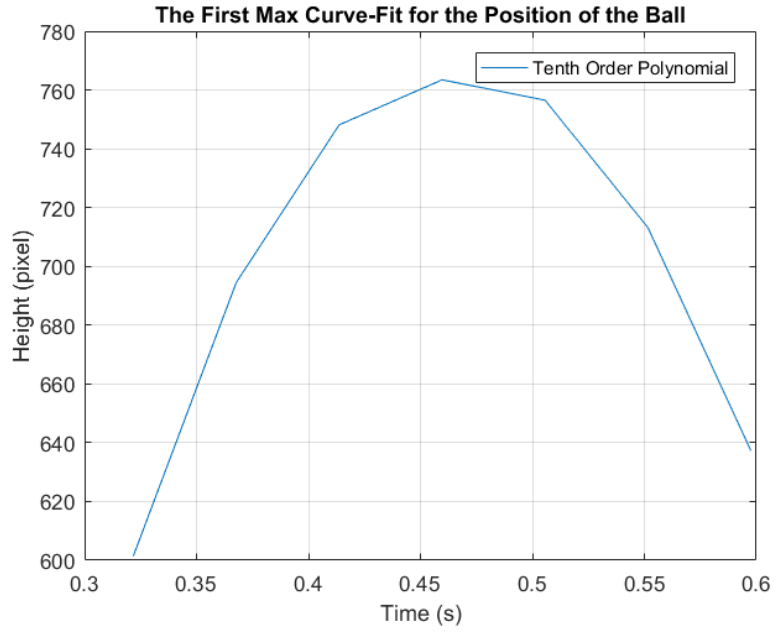
```
title('Position of Tennis Ball vs. Time')
xlabel('Time (s)')
ylabel('Position (m)')
legend('Position from Drag Equation, C_d=0','Position from Digital Image Processing')

%Plotting the Rung eKutta Approximation with CD = 0 vs position data to
%determine if drag affects the tennis ball.
figure(8);clf;hold on;box on;grid on
plot(tdrag,yRKCD0)
plot(tdraginterval,ydraginterval)
title('Position of Tennis Ball vs. Time')
xlabel('Time (s)')
ylabel('Position (m)')
legend('Position from Drag Equation, C_d=0.5','Position from Digital Image Processing')

%The two coefficients of restitution are displayed, along with the two
%errors betweent he position data and the Runge-Kutta approximation with
%and without drag.
fprintf('The coefficient of restitution using maximum heights was:
%1.2f\n',heightcoefficient)
fprintf('The error between the position data and Runge-Kutta withDrag:
%1.7f\n',errorwithoutdrag)
fprintf('The error between the position data and Runge-Kutta with Drag:
%1.7f\n',errorwithdrag)
fprintf('The coefficient of restitution using maximum velocities was:
%1.2f\n',velocitycoefficient)
```

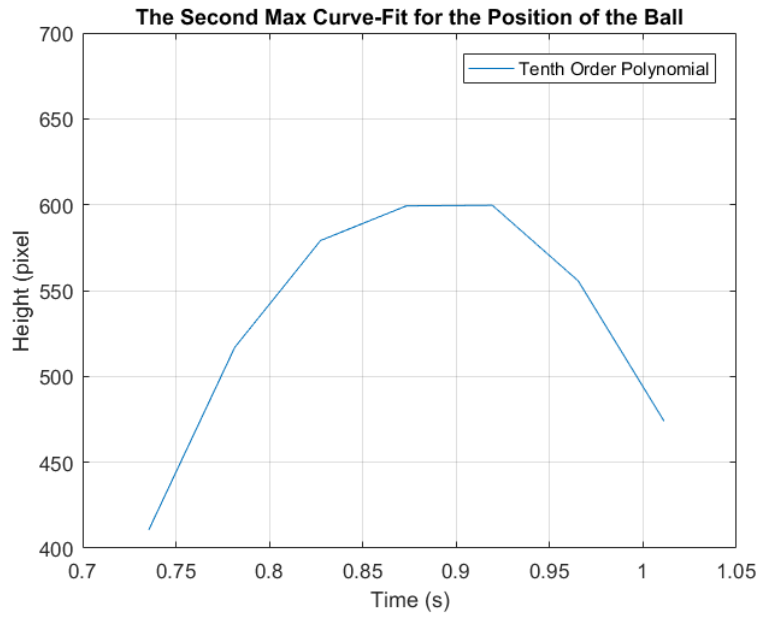
```
The coefficient of restitution using maximum heights was: 0.89

The error between the position data and Runge-Kutta withDrag:  0.0121320

The error between the position data and Runge-Kutta with Drag: 0.0120209

The coefficient of restitution using maximum velocities was: 0.75
```

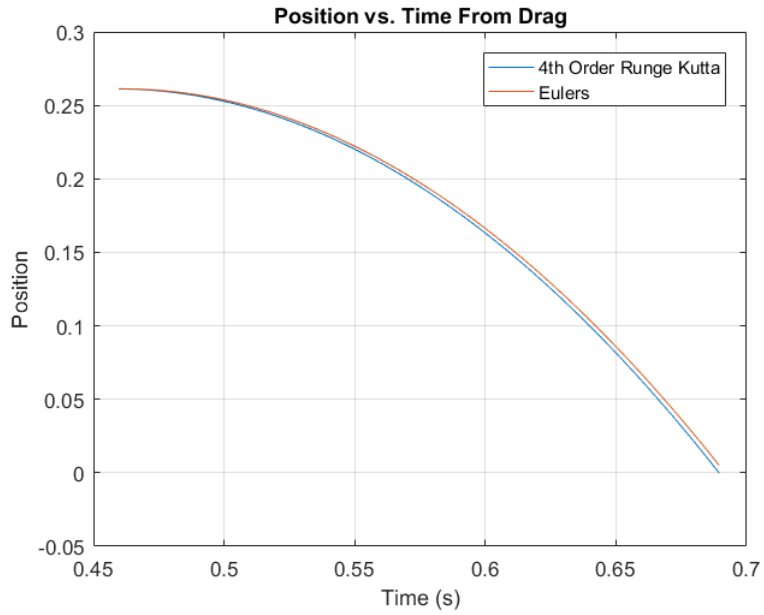

Position vs Time
Y-Position (pixel)
Time
1000
900
800
700
600
500
400
300
0
0.2
0.4
0.6
0.8
1
1.2
1.4


Velocity vs. Time
Velocity (Central Finite Difference)
Velocity (pixel/s)
Time (s)
300
200
100
0
-100
-200
-300
-400
0
0.2
0.4
0.6
0.8
1
1.2
1.4


Acceleration vs. Position
Acceleration (pixel/s^2)
Time (s)
250
200
150
100
50
0
-50
0
0.2
0.4
0.6
0.8
1
1.2
1.4


The First Max Curve-Fit for the Position of the Ball
Tenth Order Polynomial
Height (pixel)
Time (s)
780
760
740
720
700
680
660
640
620
600
0.3
0.35
0.4
0.45
0.5
0.55
0.6


The Second Max Curve-Fit for the Position of the Ball
Tenth Order Polynomial
Height (pixel
Time (s)
700
650
600
550
500
450
400
0.7
0.75
0.8
0.85
0.9
0.95
1
1.05


Position vs. Time From Drag
4th Order Runge Kutta
Eulers
Position
Time (s)
0.3
0.25
0.2
0.15
0.1
0.05
0
-0.05
0.45
0.5
0.55
0.6
0.65
0.7

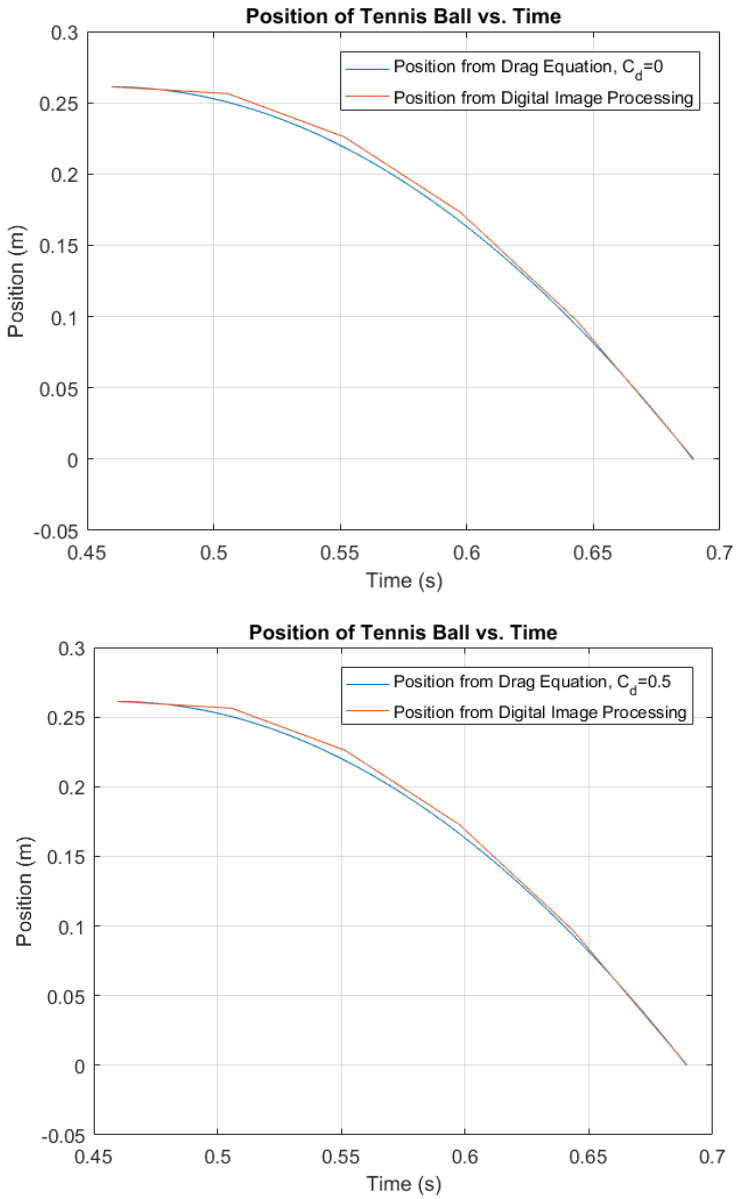


Figure 5: The Main Script for the Project that Contains the Position, Velocity, and Acceleration vs. Time, Tenth Order Polynomial Curve Fit, the Euler and Fourth Order Runge-Kutta, and the Drag Approximation and Position Data Plotsl. The Numerical Results for the Coefficients of Restitution and Error are Found in this Script as Well. The Next Figures are all utilized in the Main Script. They will be Presented in the Order that they Appear in Figure 5.

```
function [IBinary] = BinaryPic(I,MagTar)
%BinaryPic turns any picture into a binary picture
%   I is the frame or picture that needs to be processed
%   mag is the rgb magnitude for the white parts of the object

[m,n,l] = size(I);
for i = 1:m
    for j = 1:n
        A = sqrt((double(I(i,j,1))^2)+(double(I(i,j,2))^2)+(double(I(i,j,3))^2));
        AA = double(A);
        Difference = abs(AA-MagTar);
        if Difference<55
        II(i,j) = 1;
        else
            II(i,j) = 0;
        end
    end
end

IBinary = II;

end
```

Figure 7: The Algorithm that Turned Each Frame of the Video into a Binary Picture.

```
function [y] = BinaryCentroid(II)
%Binary Centroid finds the centroid of a white area in a binary image.
%   II is a binary picture

xx = 0;
yy = 0;
xc = 0;
yc = 0;
[m,n] = size(II);
for ii = 1:m
    for jj = 1:n
        if II(ii,jj) == 1
            yy = yy+1;
            yc = yc+jj;
            xx = xx+1;
            xc = xc+ii;
        else
        end
    end
end
y = xc/xx;
```

```
end
```

Figure 8: The Algorithm that Found the Centroid of the Tennis Ball at Each Frame in the Video.

```
function [F] = tenthorderpoly(b,t)
%10thorderpoly finds the equation to curvefit a set of data
%   b is a single column of data
%   t is a single column of data

[m]=length(b);
b = b';
for i=1:m
    p=0;
    for j=1:m
        A(i,j) = (t(i))^p;
        p=p+1;
    end
end

a = inv(A)*(b);
[s,u] = size(a);
F = 0;
mnp(1) = @(x) 0;
mnp = 1;

for z=1:s;
    Faux = (t.^(mnp-1))*a(mnp);
    F = F + Faux;
    mnp = mnp+1;
end

end
```

Figure 9: The Tenth Order Polynomial Curve Fit.

```
function [max] = Maxval(f,t)
%Maxval finds the maximum value for a value in a given range.
%   f is the function, f @(t)
%   t is the independent variable

n = length(t);
max = 0;
l = 1;
for h = l:n
    if f(l)>max
        max = f(l);
    else
    end
    l = l+1;
end

end
```

Figure 10: Determining the Maximum Value in a Given Set of Data. Determines Maximum Height and Maximum Velocity in the Main Script.

```
function [cfp] = centraldif(f)
%centraldif is the central differentiation of a function
%   f is the dependent variable

n = length(f);
l = 2;
hh = 1;
for h = 2:(n-1)
    cfp(hh) = (f(l+1)-f(l-1));
    l = l +1;
    hh = hh + 1;
end

end
```

Figure 11: Determining the Velocity of the Data Using the Central Finite Difference Method.

```
function [a] = acceleration(y)
%acceleration finds the acceleration with position data
%   y is the position data
%   this is analyzed in dt = 1 element. I.E. dt = 1

n = length(y);
l = 2;
for i = 2:1:(n-1)
    a(i-1) = ((y(l+1)-(2*y(l))+(y(l-1))));
    l = l+1;
end

end
```

Figure 12: The Algorithm that Determined the Acceleration of the Tennis Ball.

```
function [y0] = RungeKutta4project(t,y0,yd0,cd)
%RungeKutta4 solves a differential equation
%   t is a vector for time that needs to be analyzed.
%   y0 is the initial condition for position
%   yd0 is the initial condition for the velocity
%   cd is the drag coefficient

%   F(yd) is the function for acceleration, already implemented below
yd(1) = yd0;
y0(1) = y0;
D = 0.0673; %m %Diameter of the tennis ball
A = (pi/4)*(D^2); %m^2 %Area of the tennis ball
rho = 1.225; %kg/m3 %Density of the tennis ball
m = 0.058; %kg %mass of the tennis ball
g = 9.81; %m/s^2 %acceleration due to gravity

l = 1;
F= @(cd,yd) ((((0.5)*(rho)*(cd)*(A)*((yd)^2))/(m))-g);
h = t(2)-t(1);
n = 100;
for i = 1:n
    yd(l+1) = yd(l) + h*F(cd,yd(l));
    k1 = yd(l) + h*F(cd,yd(l));
    x2 = yd(l)+((h/3)*k1);
    k2 = yd(l) + h*(F(cd,x2));
    x3 = yd(l)-((h/3)*k1)+(h*k2);
    k3 = yd(l) + h*(F(cd,x3));
    x4 = yd(l) + (h*k1)-(h*k2)+(h*k3);
    k4 = yd(l) + h*(F(cd,x4));
    y0(l+1) = y0(l) + (h/8)*((k1)+(3*k2)+(3*k3)+(k4));
    l = l+1;
    clear k1 k2 k3 k4 x2 x3 x4
end
end
```

Figure 13: The Function for the Fourth Order Runge-Kutta Drag Approximation.

```
function [y00] = EulersProject(t,y0,yd0,cd)
%Eulers finds the position data of an object via the drag equation
%   t is a vector for time that needs to be analyzed.
%   y0 is the initial condition for position
%   yd0 is the initial condition for the velocity
%   cd is the drag coefficient

%   F(yd) is the function for acceleration, already implemented below

yd(1) = yd0;
y00(1) = y0;
D = 0.0673; %m %Diameter of the tennis ball
A = (pi/4)*(D^2); %m^2 %Area of the tennis ball
rho = 1.225; %kg/m3 %Density of the tennis ball
m = 0.058; %kg %mass of the tennis ball
g = 9.81; %m/s^2 %acceleration due to gravity

l = 1;
F= @(cd,yd) ((((0.5)*(rho)*(cd)*(A)*((yd)^2))/(m))-g);
h = t(2)-t(1);

r = 1;
n = 100;
for i = 1:n
    yd(r+1) = yd(r) + F(cd,yd(r))*h;
    y00(r+1) = y00(r) + h*yd(r);
    r = r+1;
    end
end
```

Figure 14: The Function for the Euler Drag Approximation.