Michael Cowan

Computational Physics

Problem Set 2

Bike Problem

In this problem, we calculate the terminal velocity of a cyclist taking into account the drag force acting on him (or her). We derived the equation in class, and showed that the cyclist’s ability to accelerate (disregarding the drag force) is function of their power output, their mass, and their velocity. The drag force is dependent on their velocity, their cross sectional area, their mass, the density of air, and unitless constant C.

Code:

1. m=input('Mass of the cyclist (kg): '); %Define and input all of the variables
2. P=input('Power output of the cyclist (W): ');
3. A=input('Cross sectional area (m^2): ');
4. r=input('Density of air (kg/m^3): ');
5. v(1)=input('Initial speed (m/s): ');
6. inc=input('Time increment (s): ');
7. vno(1)=v(1); %vno is the velocity without drag
8. t(1)=0;
9. n=600/inc; %calculates the number of increments in 10 minutes
10. for i=1:n %loops the calculation of v
11. v(i+1)=v(i)+((inc\*P)/(m\*v(i)))-((inc\*0.25\*r\*A\*v(i)^2)/m); %calculate the new value of v
12. vno(i+1)=vno(i)+((inc\*P)/(m\*vno(i))); %calculate the new value of vno
13. t(i+1)=t(i)+inc; %creates a matrix of time values for plotting
14. if (abs(v(i+1)-v(i))<0.0001)
15. break %break the loop if v is no longer increasing
16. end
17. end
18. fprintf('The terminal velocity of the cyclist is %g m/s \n',v(i+1));
19. plot(t,v,'-',t,vno,'--');
20. xlabel('Time (s)');
21. ylabel('Velocity (m/s)');
22. title('Graph of velocity vs. time');
23. legend('Velocity with drag','Velocity without drag');

Results:

Using the given inputs (mass = 65 kg, Power = 400 W, cross sectional area = 0.4 m2, density of air = 1.3 kg/m3, C = 1/2, an initial velocity of 1 m/s, and a time increment of 0.1 s), the program calculated a terminal velocity of 14.53 m/s. The graph below plots the velocity of the cyclist both with and without drag.

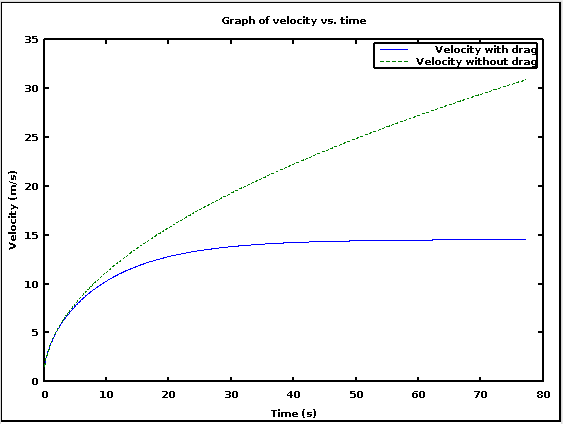


Figure 1: The velocity of the cyclist with and without drag

I also tested what would happen if the parameters were changed one at a time. Increasing the mass of the cyclist, the cross sectional area, or the density of air caused the terminal velocity to be lower, while increasing the power output of the cyclist caused the terminal velocity to be higher. All of these results are consistent with what I would have expected.

Chapter 2

2. In this problem, we calculate the absolute error of the right derivative formula, and plot the absolute error vs. the increment that we use in calculating the derivative. We do this for five different functions at a particular value of x.

a) x2 at x=1

b) x5 at x=1

c) sin(x) at x=0

d) sin(x) at x=π/4

e) sin(x) at x=π/2

Code:

1. clear all;
2. n=0:20;
3. h=10.^(-n); %set the values of h
4. for i=1:21
5. dera(i)=(((1+h(i))^2)-1^2)/h(i); %derivative for a
6. deltaa(i)=abs(2\*(1)-(dera(i))); %delta for a
7. derb(i)=(((1+h(i))^5)-1^5)/h(i); %derivative for b
8. deltab(i)=abs(5\*(1)^4-(derb(i))); %delta for b
9. derc(i)=((sin(0+h(i)))-sin(0))/h(i); %derivative for c
10. deltac(i)=abs(cos(0)-(derc(i))); %delta for c
11. derd(i)=((sin(pi/4+h(i)))-sin(pi/4))/h(i); %derivative for d
12. deltad(i)=abs(cos(pi/4)-(derd(i))); %delta for d
13. dere(i)=((sin(pi/2+h(i)))-sin(pi/2))/h(i); %derivative for e
14. deltae(i)=abs(cos(pi/2)-(dere(i))); %delta for e
15. end
16. subplot(3,3,1);
17. loglog(h,deltaa,'+');
18. xlabel('h');
19. ylabel('delta(h)');
20. title('Error delta as a function of h for y=x^2 at x=1');
21. subplot(3,3,3);
22. loglog(h,deltab,'+')
23. xlabel('h');
24. ylabel('delta(h)')
25. title('Error delta as a function of h for y=x^5 at x=1');
26. subplot(3,3,5);
27. loglog(h,deltac,'+')
28. xlabel('h');
29. ylabel('delta(h)')
30. title('Error delta as a function of h for y=sin(x) at x=0');
31. subplot(3,3,7);
32. loglog(h,deltad,'+')
33. xlabel('h');
34. ylabel('delta(h)')
35. title('Error delta as a function of h for y=sin(x) at x=pi/4');
36. subplot(3,3,9);
37. loglog(h,deltae,'+')
38. xlabel('h');
39. ylabel('delta(h)')
40. title('Error delta as a function of h for y=sin(x) at x=pi/2');

Result:

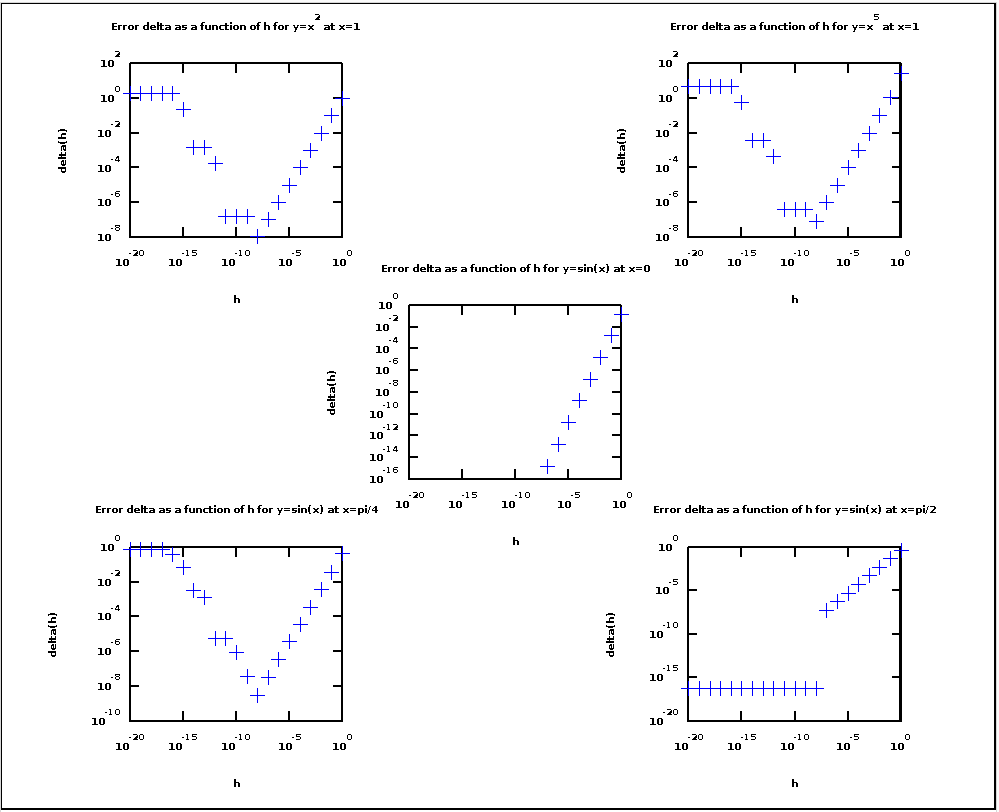


Figure 2: The error Δ as a function of h for all five functions

For a, b, and d, there is a clear minimum error at h=10-8. For c, the value of Δ appears to be 0 for values of h≤10-8. For e, the value of Δ is constant at 6.1232 X 10-17 for values of h≤10-8.

3. In this problem, we are asked to modify the **balle** program. As given to us, the program overestimates both the horizontal range and the time of flight for a projectile because it stops the program at the first point for with the y-position is <0. The task is to modify the program so that it interpolates to find the point at which the y-position=0.

The primary modifications that I made to the program are in the conditional for breaking the loop (lines 47-54). My modifications call up the **intrpf** function that we used in the previous assignment to find both the horizontal position and the time at which the vertical position is 0. The key to doing this was to put the y-values from **balle** into the x-values in **intrpf**.

We were then asked to show the improvement in the computed range and time of flight when there is no air resistance.

Code (my changes are in italics):

1. clear;
2. %\* Set initial position and velocity of the baseball
3. y1 = input('Enter initial height (meters): ');
4. r1 = [0, y1]; % Initial vector position
5. speed = input('Enter initial speed (m/s): ');
6. theta = input('Enter initial angle (degrees): ');
7. v1 = [speed\*cos(theta\*pi/180), ...
8. speed\*sin(theta\*pi/180)]; % Initial velocity
9. r = r1; v = v1; % Set initial position and velocity
10. %\* Set physical parameters (mass, Cd, etc.)
11. Cd = 0.35; % Drag coefficient (dimensionless)
12. area = 4.3e-3; % Cross-sectional area of projectile (m^2)
13. grav = 9.81; % Gravitational acceleration (m/s^2)
14. mass = 0.145; % Mass of projectile (kg)
15. airFlag = input('Air resistance? (Yes:1, No:0): ');
16. if( airFlag == 0 )
17. rho = 0; % No air resistance
18. else
19. rho = 1.2; % Density of air (kg/m^3)
20. end
21. air\_const = -0.5\*Cd\*rho\*area/mass; % Air resistance constant
22. %\* Loop until ball hits ground or max steps completed
23. tau = input('Enter timestep, tau (sec): '); % (sec)
24. maxstep = 1000; % Maximum number of steps
25. for istep=1:maxstep
26. %\* Record position (computed and theoretical) for plotting
27. xplot(istep) = r(1); % Record trajectory for plot
28. yplot(istep) = r(2);
29. t = (istep-1)\*tau; % Current time
30. xNoAir(istep) = r1(1) + v1(1)\*t;
31. yNoAir(istep) = r1(2) + v1(2)\*t - 0.5\*grav\*t^2;
32. %\* Calculate the acceleration of the ball
33. accel = air\_const\*norm(v)\*v; % Air resistance
34. accel(2) = accel(2)-grav; % Gravity
35. %\* Calculate the new position and velocity using Euler method
36. r = r + tau\*v; % Euler step
37. v = v + tau\*accel;
38. %\* If ball reaches ground (y<0), break out of the loop
39. if( r(2) < 0 )
40. *y=[xplot(istep-1),xplot(istep),r(1)]; %to use intrpf as written to find the x coordinate where y=0*
41. *x=[yplot(istep-1),yplot(istep),r(2)]; %need to switch the x and y data, use the current and two previous points*
42. *xi = 0;*
43. *yi = intrpf(xi,x,y); % Use intrpf function to interpolate to find the final x position*
44. *xplot(istep+1) = yi; % Record final value provided by intrpf*
45. *yplot(istep+1) = 0;*
46. *y=[(istep-2)\*tau,(istep-1)\*tau,istep\*tau]; %using the final three time values*
47. *t=intrpf(xi,x,y); %use intrpf function to interpolate to find the final time*
48. break; % Break out of the for loop
49. end
50. end
51. %\* Print maximum range and time of flight
52. *fprintf('Maximum range is %g meters\n',xplot(istep+1));*
53. *fprintf('Time of flight is %g seconds\n',t);*
54. %\* Graph the trajectory of the baseball
55. clf; figure(gcf); % Clear figure window and bring it forward
56. % Mark the location of the ground by a straight line
57. xground = [0 max(xNoAir)]; yground = [0 0];
58. % Plot the computed trajectory and parabolic, no-air curve
59. plot(xplot,yplot,'+',xNoAir,yNoAir,'-',xground,yground,'-');
60. legend('Euler method','Theory (No air) ');
61. xlabel('Range (m)'); ylabel('Height (m)');
62. title('Projectile motion');

Results:

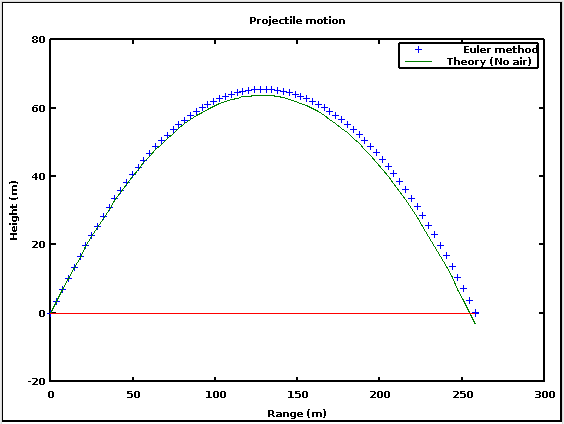
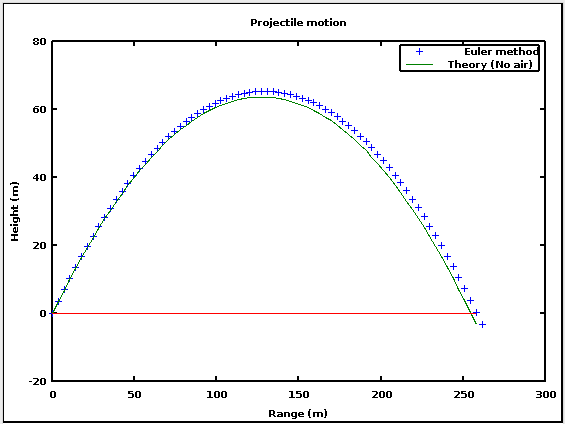


Figure 3: A comparison of the unmodified and modified **balle** program; initial height=0m, initial speed=50m/s, initial angle=45°, time step=0.1s, no air resistance

As you can see from the above graphs, the modified **balle** program (right) has the final point at y=0, whereas the unmodified **balle** program (left) has the final point at y<0. Below is a table that compares the calculated horizontal ranges and times of flight for four different time steps. In all four cases, the modified program gives a smaller range and time of flight, which is commensurate with the final point being at y=0 instead of y<0.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Unmodified | | Modified | |
| Time Step(s) | Range (m) | Time of flight(s) | Range (m) | Time of flight(s) |
| 0.01 | 255.266 | 7.22 | 255.196 | 7.21802 |
| 0.05 | 258.094 | 7.3 | 256.61 | 7.25802 |
| 0.1 | 261.63 | 7.4 | 258.378 | 7.30802 |
| 0.5 | 282.843 | 8 | 272.578 | 7.70967 |

Figure 4: Table of values for the horizontal range and time of flight calculated by the unmodified and modified **balle** programs for four different time steps

6. In this problem, we modify the **balle** program to compare the result of dropping two spherical masses at the same time.

In part a, we use this program to assess the validity of Galileo’s claim that if a 1 pound sphere and a 100 pound sphere are dropped from a height of 50 meters, the 1 pound sphere will be about two inches off the ground when the 100 pound sphere lands.

In part b, we verify that , where , then verify that this formula gives the same answer as we got it part a.

In part c, we find the value of *Cd* for which Galileo’s claim of a two inch height difference is correct.

Part a

Code (unmodified lines of **balle** in italics):

1. *clear;*
2. *%\* Set initial position and velocity of the baseball*
3. m1=input('Enter the mass of the larger sphere (lb): ');
4. m2=input('Enter the mass of the smaller sphere (lb): ');
5. h = input('Enter initial height (meters): '); % Initial height
6. r1 = h; %Set initial position for both masses
7. r2 = h;
8. v1 = 0; % Set initial velocity for both masses
9. v2 = 0;
10. m1=m1/2.20462; %convert lb to kg
11. m2=m2/2.20462;
12. *%\* Set physical parameters (mass, Cd, etc.)*
13. *Cd = 0.5; % Drag coefficient (dimensionless)*
14. area1 = pi\*((3/(4\*pi))\*(m1/2.2)\*(1000/7.8)\*10^-6)^(2/3); % Cross-sectional area of projectile (m^2)...
15. %computed using the mass and density, plus several conversion factors
16. area2 = pi\*((3/(4\*pi))\*(m2/2.2)\*(1000/7.8)\*10^-6)^(2/3);
17. *grav = 9.81; % Gravitational acceleration (m/s^2)*
18. *airFlag = input('Air resistance? (Yes:1, No:0): ');*
19. *if( airFlag == 0 )*
20. *rho = 0; % No air resistance*
21. *else*
22. *rho = 1.2; % Density of air (kg/m^3)*
23. *end*
24. air\_const1 = -0.5\*Cd\*rho\*area1/m1; % Air resistance constant
25. air\_const2 = -0.5\*Cd\*rho\*area2/m2; % for each mass
26. *%\* Loop until ball hits ground or max steps completed*
27. *tau = input('Enter timestep, tau (sec): '); % (sec)*
28. *maxstep = 1000; % Maximum number of steps*
29. *for istep=1:maxstep*
30. *%\* Record position (computed and theoretical) for plotting*
31. yplot1(istep) = r1; % Record trajectory for plot
32. yplot2(istep) = r2;
33. t(istep) = (istep-1)\*tau; % Current time
34. %\* Calculate the acceleration of the ball
35. accel1 = air\_const1\*norm(v1)\*v1-grav; % for mass 1
36. accel2 = air\_const2\*norm(v2)\*v2-grav; % for mass 2
37. %\* Calculate the new position and velocity using Euler method
38. r1 = r1 + tau\*v1; % Euler step
39. v1 = v1 + tau\*accel1;
40. r2 = r2 + tau\*v2; % Euler step
41. v2 = v2 + tau\*accel2;
42. *%\* If ball reaches ground (y<0), break out of the loop*
43. if(r1 < 0 )
44. y=[(istep-2)\*tau,(istep-1)\*tau,istep\*tau]; %to use intrpf as written to find the time where y=0 for mass 1
45. x=[yplot1(istep-1),yplot1(istep),r1]; %need to switch the x and y data, use the current and two previous points
46. xi = 0;
47. t(istep+1) = intrpf(xi,x,y); % Use intrpf function to interpolate to find the final time
48. yplot1(istep+1) = 0;
49. y=[yplot2(istep-1),yplot2(istep),r2]; %using the final three time values
50. yplot2(istep+1)=intrpf(xi,x,y); %use intrpf function to interpolate to find height of the smaller mass...
51. %when the larger mass hits the ground
52. *break; % Break out of the for loop*
53. *end*
54. *end*
55. %\* Print time of flight and height of lighter mass
56. fprintf('It takes %g seconds for the larger mass to hit the ground\n',t(istep+1));
57. fprintf('The height of the smaller mass at that time is %g meters\n',yplot2(istep+1));
58. *%\* Graph the trajectory of the baseball*
59. *clf; figure(gcf); % Clear figure window and bring it forward*
60. % Plot the heights of the two masses
61. plot(t,yplot1,'+',t,yplot2,'o');
62. legend('mass 1','mass 2');
63. xlabel('Time (s)'); ylabel('Height (m)');
64. title('Falling with air resistance');

Results:

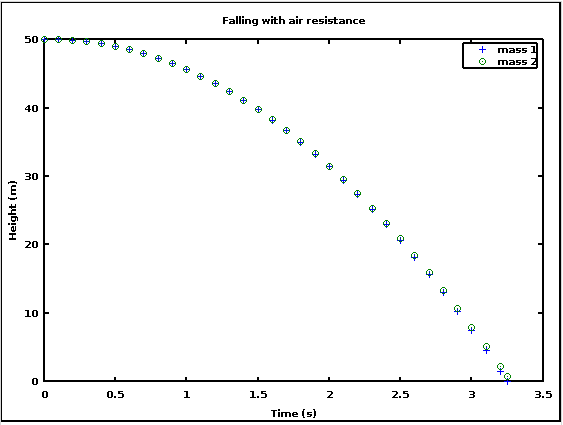


Figure 5: Plot of the heights of the two masses as a function of time; the lighter mass is slightly above the ground when the heavier mass impacts.

Enter the mass of the larger sphere (lb): 100

Enter the mass of the smaller sphere (lb): 1

Enter initial height (meters): 50

Air resistance? (Yes:1, No:0): 1

Enter timestep, tau (sec): .1

It takes 3.24963 seconds for the larger mass to hit the ground

The height of the smaller mass at that time is 0.717131 meters

The result of 0.717 meters is just over 28 inches, so Galileo’s claim is not correct.

Part b

See attached page.

Inserting the value for t from part a gives negative values for y for both objects. At first I thought that I had made a mistake, but by looking at the matrix of values for the y-positions of both objects, I realized that the fault lies primarily with the Euler method. y(1) is the starting height of 50 meters; then y(2) is calculated from y(1) using v(1). But v(1)=0, so y(2)=y(1)=50. This means that the programed projectile will always be slightly higher than a real projectile under the same circumstances. So when I used the final time value that the program gave me for when y=0 in the equation that we derived, it makes sense that the real projectile is at a negative value when the programed projectile is at 0.

Part c

Here I modified my program from Part a to run a loop to test different values of Cd.

Code (unmodified lines of **balle** in italics):

1. *clear;*
2. *%\* Set initial position and velocity of the baseball*
3. m1=input('Enter the mass of the larger sphere (lb): ');
4. m2=input('Enter the mass of the smaller sphere (lb): ');
5. h = input('Enter initial height (meters): '); % Initial height
6. m1=m1/2.20462; %convert lb to kg
7. m2=m2/2.20462;
8. area1 = pi\*((3/(4\*pi))\*m1\*(1000/7.8)\*10^-6)^(2/3); % Cross-sectional area of projectile (m^2)...
9. %computed using the mass and density, plus several conversion factors
10. area2 = pi\*((3/(4\*pi))\*m2\*(1000/7.8)\*10^-6)^(2/3);
11. *grav = 9.81; % Gravitational acceleration (m/s^2)*
12. *airFlag = input('Air resistance? (Yes:1, No:0): ');*
13. *if( airFlag == 0 )*
14. *rho = 0; % No air resistance*
15. *else*
16. *rho = 1.2; % Density of air (kg/m^3)*
17. *end*
18. *tau = input('Enter timestep, tau (sec): '); % (sec)*
19. *maxstep = 1000; % Maximum number of steps*
20. for n=1:10000 %loop to try different values of Cd
21. Cd=n./10000; %set the values of Cd for each iteration
22. r1 = h; %Set initial position for both masses
23. r2 = h;
24. v1 = 0; % Set initial velocity for both masses
25. v2 = 0;
26. air\_const1 = -0.5\*Cd\*rho\*area1/m1; % Air resistance constant
27. air\_const2 = -0.5\*Cd\*rho\*area2/m2; % for each mass
28. for istep=1:maxstep %run the loop to calculate new positions and velocities
29. %\* Record position (computed and theoretical) for plotting
30. yplot1(istep) = r1; % Record trajectory for plot
31. yplot2(istep) = r2;
32. t(istep) = (istep-1)\*tau; % Current time
33. %\* Calculate the acceleration of the ball
34. accel1 = air\_const1\*norm(v1)\*v1-grav; % for mass 1
35. accel2 = air\_const2\*norm(v2)\*v2-grav; % for mass 2
36. %\* Calculate the new position and velocity using Euler method
37. r1 = r1 + tau\*v1; % Euler step
38. v1 = v1 + tau\*accel1;
39. r2 = r2 + tau\*v2; % Euler step
40. v2 = v2 + tau\*accel2;
41. *%\* If ball reaches ground (y<0), break out of the loop*
42. if(r1 < 0 )
43. y=[(istep-2)\*tau,(istep-1)\*tau,istep\*tau]; %to use intrpf as written to find the time where y=0 for mass 1
44. x=[yplot1(istep-1),yplot1(istep),r1]; %need to switch the x and y data, use the current and two previous points
45. xi = 0;
46. t(istep+1) = intrpf(xi,x,y); % Use intrpf function to interpolate to find the final time
47. yplot1(istep+1) = 0;
48. y=[yplot2(istep-1),yplot2(istep),r2]; %using the final three time values
49. yplot2(istep+1)=intrpf(xi,x,y); %use intrpf function to interpolate to find height of the smaller mass...
50. %when the larger mass hits the ground
51. *break; % Break out of the for loop*
52. *end*
53. *end*
54. if (yplot2(istep+1)>0.0508)
55. break;
56. end
57. end
58. fprintf('It takes %g seconds for the larger mass to hit the ground\n',t(istep+1));
59. fprintf('The height of the smaller mass at that time is %g meters\n',yplot2(istep+1));
60. fprintf('The drag coeffecient Cd is %g\n',Cd);

Results:

Enter the mass of the larger sphere (lb): 100

Enter the mass of the smaller sphere (lb): 1

Enter initial height (meters): 50

Air resistance? (Yes:1, No:0): 1

Enter timestep, tau (sec): .1

It takes 3.24361 seconds for the larger mass to hit the ground

The height of the smaller mass at that time is 0.0508583 meters

The drag coeffecient Cd is 0.0346

So a drag coefficient of Cd≈0.0346 gives a height difference between the two balls of about two inches.