PHSX 221L Computer Simulation of a Falling Object Name:_____

Object: (a) To run a pre-written program which uses numerical methods to compute the position and speed of a freely-falling object dropped from rest at any given time. (b) To run the same simulation with a spreadsheet application, and compare the tools. (c) To modify this simulation to use average values of velocity, and to take into account air resistance. (d) To compare each of the above solutions with the exact solution obtained from mathematical methods.

Theory: Newton's 2nd law, $\Sigma \boldsymbol{F} = m\boldsymbol{a}$, is the governing equation for any accelerating object. For the one-dimensional motion considered here we drop the vector notation. We also assume the objects are dropped from rest, and we take the positive direction to be down.

For free-fall the only force is gravity ($F_g = mg$, where g is the local gravitational field strength which is 9.80 N/kg near the surface of the earth), so $mg = ma$. Canceling the m leaves $a = g$ but don't interpret this to mean that g *is* an acceleration—it is simply that the acceleration is *numerically equal* to g. (In an earlier lab we called the free-fall acceleration a_g.)

From the definitions of velocity and acceleration we can write two coupled differential equations:

$$a = \frac{\mathrm{d}v}{\mathrm{d}t} = g \tag{1}$$

$$v = \frac{\mathrm{d}y}{\mathrm{d}t} \tag{2}$$

Converting these to finite difference equations by changing the differential operator d to Δ we have

$$v_{i+1} = v_i + a_i \Delta t \tag{3}$$

$$y_{i+1} = y_i + v_i \Delta t \tag{4}$$

where a_i is always numerically equal to g, and $v_0 = y_0 = 0$. This is Euler's Method (see section 6.5 in Serway). These equations produce computations which can be tabulated. The $(i+1)$th values are computed from the ith values, *i.e.*, each row in the table is computed from the previous row.

i	t	a	v	y
0	0	g	0	0
1	Δt	g	v_1	y_1
2	$2\Delta t$	g	v_2	y_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
n	$n\Delta t$	g	v_n	y_n

Table 1: Simple Euler Method Without Air Resistance.

Notice that the $\frac{1}{2}a_i(\Delta t)^2$ term which would make equation 4 more analogous to Serway's equation 2.11 is not included in Euler's method because it is assumed to be small; but this omission will produce some error in the last column.

Of course equation 1 can be integrated (twice) analytically to obtain the exact solution

$$v = gt \tag{5}$$

$$y = \frac{1}{2}gt^2 \tag{6}$$

where both constants of integration are zero (*i.e.* at $t = 0$, $v_0 = y_0 = 0$). Then the values in the generated table can be compared to the exact solutions.

The solution by numerical methods is in general only an approximation to the exact solution. Theoretically, the numerical solution should converge to the exact solution in the limit as $\Delta t \to 0$, but there are also round-off and truncation errors. The basic Euler method doesn't obscure the physics as more sophisticated methods (such as the Runge-Kutta) might; but it is seldom used because the error is too large. However, a simple modification to the Euler method produces significantly better results; it entails using the average velocity during a time interval instead of just the velocity at the beginning of the interval to compute the next value of y.

When an object moves through a fluid, such as air, the drag force F_D is $\frac{1}{2}DA\rho v^2$ where D is the drag coefficient, A is the cross sectional area of the object normal to its path, and ρ is the density of the fluid. For situations of slow motion through a viscous fluid (low Reynolds number) D depends on the other variables in a way that leaves the drag force proportional to v. Stoke's Law, $F_D = 6\pi\eta rv$ (where η is the viscosity), then applies. For situations of faster motion through less viscous fluids (moderately high Reynolds number) the drag force is proportional to the square of the speed, $F_D \propto v^2$, because D is almost constant. We examine the second case here, as is generally done for macroscopic objects in air. (See also section 6.4 in Serway.)

For falling through air, there are two forces on the left side of Newton's 2nd law.

$$\Sigma F = mg - kv^2 \tag{7}$$

where we use k for $\frac{1}{2}DA\rho$. Newton's law then gives

$$mg - kv^2 = ma. \tag{8}$$

Acceleration is decreasing ($a = g - (k/m)v^2$) and the speed approaches a constant value called terminal speed, v_t.

From the definitions of velocity and acceleration we can write two coupled differential equations:

$$a = \frac{\mathrm{d}v}{\mathrm{d}t} = g - \frac{k}{m}v^2 \tag{9}$$

$$v = \frac{\mathrm{d}y}{\mathrm{d}t} \tag{10}$$

Converting these to finite difference equations by changing the differential operator d to Δ we have

$$a_{i+1} = g - \frac{k}{m}v_i^2 \tag{11}$$

$$v_{i+1} = v_i + a_i \Delta t \tag{12}$$

$$y_{i+1} = y_i + v_i \Delta t \tag{13}$$

where $a_0 = g$, and $v_0 = y_0 = 0$.

These equations produce computations which can be tabulated. In the following example the simple Euler method is used with $g = 10.0$, $\Delta t = .10$, $m = 5.0$, and $k = 0.050$ in S.I. units.

i	t (s)	a (m/s^2)	v (m/s)	y (m)
0	0.00	10.000	0.000	0.000
1	0.10	10.000	1.000	0.000
2	0.20	9.990	2.000	0.100
3	0.30	9.960	2.999	0.300
4	0.40	9.910	3.995	0.600
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Table 2: Simple Euler Method With Air Resistance, numerical example.

Though analytic integration of differential equations is often not possible, equation 9 can also be integrated (twice) analytically to obtain the exact solutions

$$v = v_t \tanh(t/\tau) \tag{14}$$

$$y = v_t \, \tau \ln \cosh(t/\tau) \tag{15}$$

where $\tau = \sqrt{m/kg}$ and is called the characteristic time, and $v_t = \sqrt{mg/k}$; again both constants of integration are zero. The result can be compared with that obtained from the numerical procedure.

Procedure:

1. Type the Euler's Method free-fall program into a computer using any language you like (choose one that is common in your major, or consider BASIC if you haven't ever done any programming before). Also do it on a spreadsheet for comparison of these two tools. Run it using a total time of 10 seconds and 25 intervals (*i.e.* $\Delta t = 0.40$ s) and compute the distance fallen. Then reduce Δt to 0.10 s and find the distance fallen after 10 seconds using 100 intervals. Compare each of these with the exact solution given by equation 6. Which run shows the best agreement? Why? How close are they to the exact answers?

2. Modify the program and the spreadsheet to use the Improved Euler Method by averaging velocities over each interval. That is, compute v_{i+1} before you compute y_{i+1}, and use the average of v_{i+1} and v_i instead of just v_i in equation 4. Now run the program again with 25 time intervals and compare with the exact solution. Is the agreement better? Why? How good is it?

3. Now return to the unmodified (simple) Euler Method and add air resistance; assume the falling body has a mass of 4.0 kg and has a shape, size, and speed which cause the air resistance to be proportional to the square of the velocity. If the coefficient of resistance k is 0.035 kg/m, then $F_D = 0.035v^2$ if F is in newtons and v is in m/s. Write a new version of your program and your spreadsheet that will calculate the distance fallen in 10 seconds using equations 11–13. In addition to changing the formula for a you will need to use a technique similar to the one you used in procedure 2. (You can verify that your program is working correctly by trying the sample values on the previous page first and comparing with table 2.) Try it for both 25 and 100 intervals. How do these results agree with an exact solution obtained by integration?

4. In your report show the *changes* you made to the computer programs (without necessarily turning in the whole program). When you show results for 100 intervals show the first few lines and the last few lines, but page after page of values of intermediate results are not necessary or desirable.

5. Plot a, v, and y vs. t on sets of vertically stacked axes (with the horizontal t axes parallel) for both the free-fall case and (on another set of axes) the case with air resistance.

Conclusions:

1. How did you tailor this lab to fit your major?

2. Compare the appropriateness of the two tools: a program in a programming language and a spreadsheet application.

3. How does one know when a particular numeric approximation is "good enough"?

4. What did you learn from this lab?

```
10 rem this BASIC program performs numerical integration to calculate the speed
20 rem and distance of fall as a function of time for a freely-falling body
30 print "Enter the total time of fall and the number of intervals"
40 input "Total time = ";tt
50 input "Number of intervals = ";n
60 dt = tt/n : rem dt is delta t
70 i = 0 : t = 0 : a = 9.8 : v = 0 : y = 0 : rem initialize other vars
80 rem i is counter, t is time, a is accel, v is velocity, y is distance.
90 print "   i      t          a              v              y"
100 print "------------------------------------------------------------"
110 for i = 0 to n
120 print using "####";i;" ";
130 print using "####.###";t,a,v,y
140 t = t+dt
150 y = y+v*dt
160 v = v+a*dt
170 next i
180 end
```

This BASIC program was written with Chipmunk BASIC (which is free) on a Macintosh; but Chipmunk BASIC doesn't print directly—the code or results must be copied and pasted into a text editor (such as Alpha) and printed from there. There are also some versions of BASIC for PCs, but I don't know which are available to you. If you use Chipmunk BASIC you will want to use the LIST, RUN, RENUM, SAVE, and LOAD commands (the last two can be done from the File menu).

```
/* FreeFall.java  Example */
/* This Java program performs numerical integration to calculate the speed*/
/* and distance of fall as a function of time for a free-falling body */
/* Input the total time and number of intervals on the command line. */
class FreeFall
{
   public static void main(String[] args)
   {
      int N = 25, i = 1;
      double t = 0.0, a = 9.8, v = 0.0, y = 0.0, T = 10.0, dt;

      if (args.length > 0)
      {
         T = Integer.parseInt( args[0] );
         N = Integer.parseInt( args[1] );
      }
      dt = T/N;

      System.out.println("   i      t       a       v       y");
      System.out.println("_______________________________________");
      for (i = 0; i <= N; ++i)
      {
         System.out.println( "   " + i + "    " + t + "    "
                             + a + "    " + v + "    " + y);
         t = t + dt;
         y = y + v*dt;
         v = v + a*dt;
      }
   }
}
```

```
/*
Purpose:    Contains a C++ program that performs numerical integration to
            calculate the speed and distance of fall as a function of
            time for a free-falling body
File:       FreeFall.cpp
Author:     Garth Sorenson
Date:       28 Sep 2000
*/


#include <iostream>
using std::cin;
using std::cout;
using std::endl;
using std::ios;

#include <iomanip>
using std::setw;
using std::setprecision;
using std::setiosflags;


/*
Purpose:    Controls free-fall program
Input:      total time as floatin point, number of intervals as an integer
Output:     interval, time, acceleration, velocity, distance for each inteval
*/
int main()
{
    // local data
    int N,      // number of intervals
        i;      // loop control variable
    double T,   // total time
           t,   // current time
           a,   // acceleration
           v,   // velocity
           y,   // distance
           dt;  // delta time

    // greeting
    cout << "This program will calculate the speed and distance\n"
         << "of fall as a function of time for a freely-falling body.\n\n"
         << "Enter the total time of fall, and the number of intervals.\n"
         << endl;
```

```
    // get total time and number of intervals from user
    cout << "Total Time = ";
    cin >> T;
    cout << "Number of Intervals = ";
    cin >> N;
    cin.ignore(80, '\n');

    // initialize delta time, current time,
    // distance, velocity, and acceleration
    dt = T / static_cast<double>(N);
    i = t = y = v = 0;
    a = 9.80;

    // display table header
    cout << setw(3) << "i" << setw(8) << "t" << setw(10) << "a"
         << setw(10) << "v" << setw(10) << "y" << endl
         << "-----------------------------------------" << endl;

    // set formatting
    cout << setiosflags(ios::fixed | ios::showpoint)
         << setprecision(2);

    // display interval values
    for (i = 0; i <= N; i++)
    {
        cout << setw(3) << i << setw(10) << t << setw(10) << a
             << setw(10) << v << setw(10) << y << endl;
        t += dt;
        y += v * dt;
        v += a * dt;
        cin.get();
    }

    return 0;
}
```