Simulating Projectile Motion in C++
T. Dunn
Physics 193
Feb 13 2004

1 Introduction

I created a computer program in c++ to simulate projectile motion in two dimensions taking into consideration first gravity, then air resistance, and finally air resistance as a function of height. I used the Euler algorithm in my simulation, and later the Runge-Kutta algorithm. Air resistance was calculated based on a quadratic dependance of velocity. Using the results of the simulation, maximum distances and their related angles were estimated.

2 Theory

When solving a second order differential equation like

\[ F = ma = m \frac{d^2y}{dt^2} \]  

one method that can be implemented is to break the differential equation into two first order differential equations.

\[ \frac{dv_y}{dt} = \frac{F(t, v_y, y)}{m} \]  
\[ \frac{dy}{dt} = v_y \]

Which can be solved simultaneously by differentiating them or by another method such as the Euler method.

\[ v_{n+1} = v_n + a_n \Delta t \]  
\[ y_{n+1} = y_n + v_n \Delta t \]

where we solve by taking small incremental steps and iterating them over a desired period of time.

For these methods we calculate force from air resistance to be

\[ F = -\rho_{air} A_{obj} v^2 \]  

1
\[ F = -\rho_{\text{air}} e^{-\frac{v}{u}} A_{\text{obj}} v^2 \]  

for air resistance as a function of height. Where the force is always pointing in the opposite direction of the velocity.

3 Simulation

Using C++ I created the following program for simulating two dimensional motion:

```cpp
#include <iostream>
#include <cmath>
using namespace std;
#define SP " "
#define vel sqrt((V[0]*V[0])+(V[1]*V[1]))

void impulse(double V[], double A[], double dt, double P[]);
void accel(double V[], double A[], double P[]);

int main (){  
  // initializing values
  // V[0] = y ; V[1] = x
  double V[] = {0.0, 0.0};
  double A[] = {0.0, 0.0};
  double P[] = {0.0, 0.0};
  double t = 0.0, ang, dt;
  // getting dt value
  cout << "#enter dt: ";
  cin >> dt;
  cout << "#enter angle: ";
  cin >> ang;
  V[0] = 700.0*sin(ang);
  V[1] = 700.0*cos(ang);
  // loop to find the changes in y and x and so forth
  while (P[0] >= 0){
    impulse(V, A, dt, P);
    // update of y value
```
P[0] += V[0]*dt;
// update of x value
P[1] += V[1]*dt;
// update of t value
t += dt;
// output of t and y values
cout << t SP P[0] SP P[1] << endl;
}
return 0;
}

// use the acceleration to find the change in velocity (not exactly the impulse but close enough)
void impulse(double V[], double A[], double dt, double P[]){
    accel(V, A, P);
    V[0] += A[0]*dt;
}

// added in this to make editing for air resistance easier
void accel(double V[], double A[], double P[]){
double Bm;
Bm = (4.0e-5)*exp(-P[0]*1.0e-4);
A[0] = -9.8 -V[0]*Bm*vel;
A[1] = -V[1]*Bm*vel;
}

This program is including air resistance as a function of height; when simulating just air resistance Bm = 4.0e-5 and when simulating with no air resistance Bm = 0.

When dealing with how accurate these algorithms are I could not compare with an actual real life trial because i had no data on such simulations. I was able, however, compare my simulation without air resistance to the known answer which can be found by solving the equation:

\[ p = p_0 + v_y t + \frac{1}{2} at^2 \]  

for time and substituting it into

\[ D = v_x t_{air} \]  

(9)

to find a total distance of 50,000 meters. Where as the simulation of the same event gave values of
<table>
<thead>
<tr>
<th>Δt</th>
<th>$D_{sim}$</th>
<th>$D_{cal}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>.01</td>
<td>49.997.4</td>
<td>50.000</td>
</tr>
</tbody>
</table>

So where my simulation was not spot on accurate with a timestep of .01 seconds for the sake of experiment it is accurate enough to give a simulation of our projectile with air resistance that should be accurate within a few meters.

Next in order to get a more accurate simulation I incorporated the Runge-Kutta Algorithm where instead of taking values at the beginning of each iteration, half steps are taken in order to get midpoint values for each variable. Then using those midpoint values new values are calculated which are then used in order to find the next full step in the iteration.

The accuracy of this method can be compared to my original Euler method as shown in Table .1

<table>
<thead>
<tr>
<th>timestep</th>
<th>error (Euler)</th>
<th>error (Runge-Kutta)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.137346</td>
<td>.000035</td>
</tr>
<tr>
<td>.5</td>
<td>.068661</td>
<td>.000009</td>
</tr>
<tr>
<td>.1</td>
<td>.013736</td>
<td>3.76e-7</td>
</tr>
<tr>
<td>.05</td>
<td>.006865</td>
<td>1.11e-7</td>
</tr>
</tbody>
</table>

Table 1: This is a comparison of the errors caused by the Euler and Runge-Kutta methods used in a program to find the quantity of a radio active substance after a set time. In this case we started with 1000.0 units of radio active substance and let it sit for 1000 minutes, it had a half life of 15 hours.

It should be noted that the Runge Kutta method does not change the outcome of of the simulations without air resistance because the force on the projectile is constant. The Runge-Kutta method is not used for the rest of this report as I was not so much concerned with getting a super accurate reading as I was concerned with seeing a general result in how air resistance affects the path of our projectile.

4 Data

Once it had been determined that my program could give accurate values for distance without air resistance my next goal was to find the maximum
distance the projectile could go including air resistance and then air resistance as a function of time. This was done empirically through trial and error by changing the angle as shown in Table .2.

<table>
<thead>
<tr>
<th>trial</th>
<th>angle (air)</th>
<th>distance</th>
<th>angle (height)</th>
<th>distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.7854</td>
<td>21693.1</td>
<td>1</td>
<td>24807.3</td>
</tr>
<tr>
<td>2</td>
<td>.7</td>
<td>22045.3</td>
<td>.5</td>
<td>23144.6</td>
</tr>
<tr>
<td>3</td>
<td>.6</td>
<td>21869</td>
<td>.6</td>
<td>25002.4</td>
</tr>
<tr>
<td>4</td>
<td>.65</td>
<td>22039.9</td>
<td>.7</td>
<td>26184.2</td>
</tr>
<tr>
<td>5</td>
<td>.67</td>
<td>22061.2</td>
<td>.9</td>
<td>26182.6</td>
</tr>
<tr>
<td>6</td>
<td>.68</td>
<td>22062.3</td>
<td>.81</td>
<td>26608.7</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td>.79</td>
<td>26605</td>
</tr>
<tr>
<td>best</td>
<td>.675</td>
<td>22063.4</td>
<td>.8</td>
<td>26612.6</td>
</tr>
</tbody>
</table>

Table 2: Trials used to find the maximum distance as a function of angle using air friction and air friction as a function of height. Angles are measured in radians and distances in meters. All were run with a timestep of .01 and all distances were taken to be the first value when \( y < 0 \)

Next I compared the maximums of the different simulations in order to see how air resistance effected the path of the projectile. I then plotted their paths in Gnuplot. The trajectories of the various simulations at their maximum distances are shown in figure 1.

From this it is easy to see that air resistance plays a large role in determining how a projectile will behave when fired through air.
Figure 1: Compares the maximum distance the projectile traveled with air resistance (twodair.dat), air resistance as a function of height (twodh.dat), and no air resistance (twod.dat). The angles and maximum distance were found empirically for air resistance and air resistance as a function of height to be: .675 radians and 22,063.4 meters, and .8 radian and 26,612.6 meters respectively.
5 Analysis

It was quite surprising to see the dramatic effect that air resistance had on the path of a projectile. Where my calculated maximum distances are not as accurate as they could have been if I had taken a timestep of $\Delta t = .001$ seconds. The goal of this experiment was simply to view how air resistance would affect the path of our projectile. It showed that air resistance plays a major role in effecting the path of a projectile. Our projectile at best traveled just more than half the distance of our projectile did without air resistance. The next step in this simulation would be to compare it to know values of actual projectile motion in air.