Projectile Motion

1 Objective

Study graphically the path of a projectile with and without air drag. Find the horizontal range and maximum height in either case

2 Introduction

A projectile is any object thrown into space upon which the only acting force is gravity. This does not necessarily mean that other forces do not act on it, it is just that their effect is minimal compared to gravity. When a particle is thrown obliquely near the earth's surface, it moves along a curved path under constant acceleration directed towards the center of the earth (we assume that the particle remains close to the earth's surface). The path of such a particle is called a projectile, and the motion is called projectile motion.

3 Parameters Required for the C++ Code

To simulate the projectile motion using C++, we need to list the parameters involved in its dynamics:

1. A body with **mass** and **radius (r)**(for simplicity, only spherical body is considered)
2. To observe the body's motion in the air, we need to release it with some **initial velocity (v_0)** and at a certain **release angle (theta)**.
3. Another important parameter is the **drag coefficient (drag_coeff)** of the body. The drag coefficient mainly depends on the shape of the body, but the coefficient also contains the complex dependencies of object inclination, the effects of air viscosity, and compressibility. In this simulation, we have considered a simpler case where the body only has a spherical shape and air is the only medium under consideration,
   – drag_coeff=0, when air-drag is ignored
   – drag_coeff=0.47, when air-drag is being applied on a spherical body

Parameters Input by User	**Symbol**
Mass	mass
Radius	r
Initial Velocity	v_0
Release Angle	theta
Drag Coefficient	drag_coeff

Table 1

Pre-defined Parameters	**Symbol and Value**
Acceleration due to Gravity	g = - 9.8 m/s.s
Density of Air	density = 1.29 kg/m^3
Step size	h=0.001
Initial x-position	x_0 = 0.0
Initial y-position	y_0 = 0.0

Table 2

4 Projectile Motion in the Absence of Air Drag

1. Along the x-axis, uniform velocity is responsible for the particle's horizontal (forward) motion.
2. Along the y-axis: uniform acceleration, responsible for the vertical (downwards) motion of the particle.

In the case of ideal projectile motion, the only force acting on the body during its time in the air is the weight which arises because of the acceleration due to gravity. This leads to an acceleration acting in a vertically downward direction. There is no acceleration in the horizontal direction, which means that the velocity of the particle in the horizontal direction remains constant.

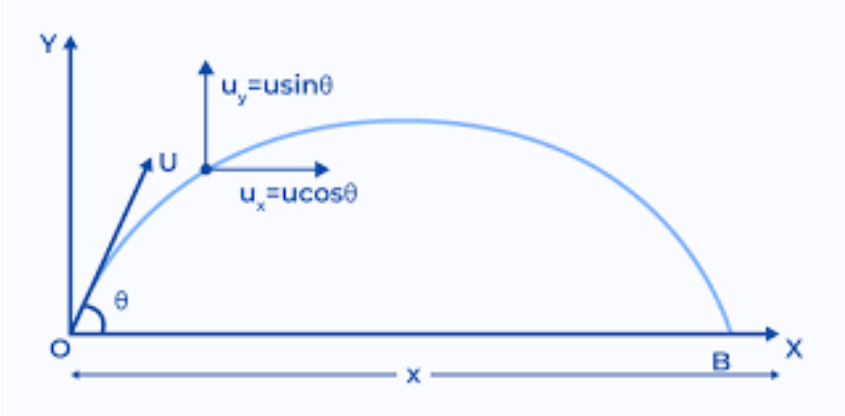

Figure 1: Horizontal and vertical components of the initial velocity of projectile

5 Equations Involved

For the body under consideration, let u be the initial velocity of the body, t is the total time, v is the final velocity of the body, and g is the acceleration due to gravity. To find the different parameters mentioned, we use the following set of equations:

$$v = u - gt \tag{1}$$

$$S = ut - \frac{1}{2}gt^2 \tag{2}$$

$$v^2 = u^2 - 2gs \tag{3}$$

Other important parameters are:

1. **Range** - It is the total distance covered in the horizontal direction.

$$R = \frac{u^2 sin(2\theta)}{g} \tag{4}$$

2. **Maximum Height** - It is the greatest height achieved by the projectile.

$$H = \frac{u^2 (sin(\theta))^2}{2g} \tag{5}$$

The equation of trajectory for projectile motion proves that the motion is always parabolic,

$$y(x) = xtan(\theta) - \frac{gx^2}{2u^2 cos((2\theta))^2} \tag{6}$$

6 Projectile Motion in the Presence of Air Drag

Air resistance (often called air drag) has a major effect on the projectile motion. Drag force depends on velocity and it exists only when there is relative motion between an object and a fluid. The equation for drag force is given as:

$$F_d = \frac{1}{2} * \rho v^2 C_d A, \tag{7}$$

where ρ = Density of fluid,
v = speed of the object relative to the fluid,
C_d = Drag coefficient, and
A = Cross-sectional area

where

$$v^2 = v_x^2 + v_y^2, \tag{8}$$

$$v_x = v_0 * cos(theta) \tag{9}$$

$$v_y = v_0 * sin(theta) \tag{10}$$

Let us take

$$D = \frac{1}{2} * \rho C_d A \tag{11}$$

Hence, the drag force equation becomes

$$F_d = Dv^2 \tag{12}$$

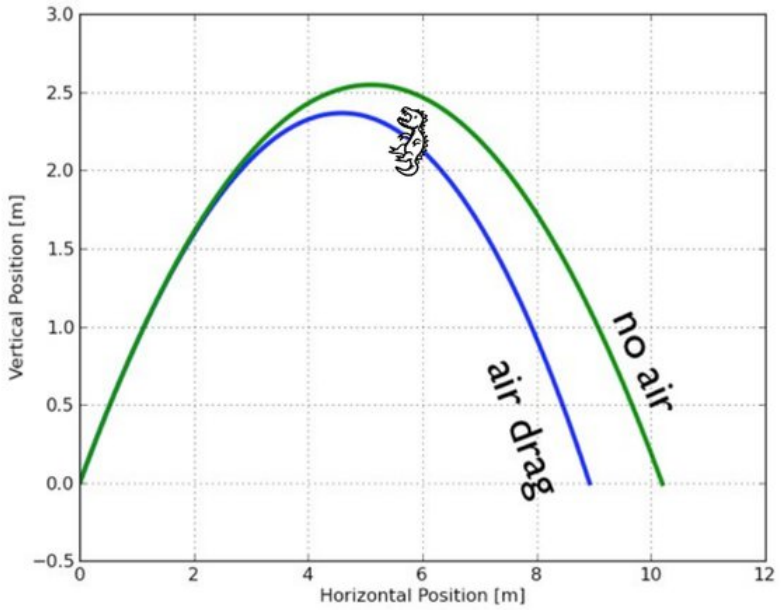

Figure 2: Motion in the presence of air drag vs in the absence of it

7 Equations Involved

We'll assume that the air is still, so is the velocity of the projectile relative to the ground as well as to the air. The direction of is opposite the direction of D, so we can write,

$$f = -Dvv \tag{13}$$

and the components of f are:

$$D_x = -Dvv_x \tag{14}$$

and,

$$D_y = -Dvv_y \tag{15}$$

Now, using Newton's second law of motion, we get

$$\sum f_x = -Dvv_x \implies ma_x \tag{16}$$

$$\sum f_y = -Dvv_y - mg \implies ma_y \tag{17}$$

Finally, the components of acceleration are:

$$\sum a_x = \frac{-Dvv_x}{m} \tag{18}$$

$$\sum a_y = -g - \frac{Dvv_y}{m} \tag{19}$$

The constant D depends on the density of air ρ, the silhouette area of the body (its area as seen from the front), and a dimensionless constant called the drag coefficient that depends on the body's shape. Typical values of C for baseballs, tennis balls, and the like are in the range of 0.2 to 1.0.

8 Algorithm

1. **Include all the required header files.**
2. **Define a class 'projectile'** which has all the above-listed parameters in Table 1 and Table 2 declared as private members.
3. **Declare public member functions** in the class that can be accessed from outside the class using an object or instance of the class defined in the main function (the compiler starts the execution of any C++ program from the main function)

   (a) **Definition of Member function 1**: void input(void){//}

   The function inputs or initializes the initial parameters, like initial velocity, launch angle, drag coefficient of the body, etc. Apart from taking input from the user, the function implements three other important tasks:

   - **Conversion of angle** input by the user in degrees, to radians as the formulas involved require the angle in radians only.

```
// Conversion of angle from degrees to radians
{ // some lines of code
    theta = (M_PI/180.0)*theta;
    /* M_PI is a way to use the value of PI in C++. Another and a better way is:*/
    theta = 4*atan(1)*theta/180
    /* Here the method of representation of PI is different. This formula comes
        from atan(1)= PI/4 and hence PI=4*(atan(1)) where atan(1) represents the
        tan^-1(PI/4)*/
    // some other lines of code
}
```

   - Calculation of the **maximum height** that a projectile can achieve in the ideal conditions. This block of code uses the standard formula mentioned in the equation (5) above.
   - A **for loop** to ensure that drag_coeff becomes zero, for the case when projectile motion is observed in the ideal conditions, i.e., in the absence of air drag.

   (b) **Definition of Member function 2:** double euler(double,double,double){//}
   The function implements the Euler method to solve the problem and approximate the solution.

   - Return data-type: double
     This function returns the computed value of v(t+h) i.e., the value of the projectile's velocity at a particular time step, or y(t+h) i.e., the value of the projectile's displacement at a particular time step, to the function calc() from where the function call is made.

- The function accepts 3 parameters of double data type:
  The first parameter is the velocity/displacement computed in the previous time step, the second parameter is the acceleration/velocity evaluated, respectively, at the previous time step, and the third parameter is the time step itself. Don't get confused by the slashed parameters. This is a generic function that can compute either the velocity or the displacement of the projectile, depending on the parameters passed to it.) Using these parameters the Euler Method can be easily executed.
  Check out the documentation of the Euler Method for more information.

(c) **Definition of member function 3**: double drag(double);

The function calculates the projectile's acceleration in x- and y-directions.

- Return data-type: double
  This function evaluates the values of acceleration using formulas represented in equations (18) & (19) and returns the values to euler() function from where the call is made.
- The function accepts 1 parameter of double data type:
  The function accepts velocity as a parameter which is computed via the Euler Method by euler() function; other values in equations (18) & (19) are just constants.

(d) **Definition of member function 4:** void calc(void)

The function enters the values of displacements along x-and y-directions into a data file.

- return type: void
  This function doesn't return any value but implements an integral part of the code, i.e. entering the data computed to a .txt file.
  This action is implemented using file input/output commands.
- The function doesn't accept any parameter.
  The function makes function calls to the euler() function to compute either the velocities or the displacements of the particle at different time steps.
- The function also evaluates the range of the projectile.
  Just by making a function call to the euler() function, we can easily find the range of the projectile:

```
// Logic for computing the range of projectile
for(double t=0;t<100;t+=h)
{
    // some lines of code, computing v_x
    euler(x_0,v_x,h);
    /* This function call passes the initial value of displacement, velocity, and
        step size to euler(), and as the loop executes, each iteration gives the
        value of x(t+h). At the final iteration, we obtain the maximum value of x
        i.e., the range.*/
}
```

- The function evaluates the maximum height of the projectile under the effects of drag, too.

```
// Logic for computing the maximum height
for(double t=0;t<100;t+=h)
{
    // some lines of code, computing v_x and y
    if (y_0>max_h)
        {
            max_h=y_0;
        }
    /* max_h is initialized to zero first and inside the loop, this if-condition
        is checked at each iteration. First, when the loop runs, the computed
        value of y_0 (via euler()), is greater than zero, hence max_h will be
        assigned that value. Now, when the loop runs next time, y_0 has a new
        value and this is again compared to the max_h (=previous value of y_0),
        if y_0 is still greater then that value of y_0 is assigned to max_h and
```

```
            this process goes on. Finally, a case will come when the subsequent value
            of y_0 is less than max_h, this will stop the execution of the loop, and
            whatever the value of max_h, it will appear as the maximum height
            attained by the projectile. */
    }
```

(e) **Definition of member function 5:** void output(void);

The function displays the range and maximum height computed by the calc() function on the screen.

4. Define a **main() function**; this is the most important function in any C++ program because the compiler starts execution from this function:

```
{
    int main()
    {
        // lines of code
    }
}
```

- Return data-type: int
  The main() function has 'int' as its return data type, which means that the function will return an integer value. The decision about which value to return is made by the following statement:

```
{
    return 0;
    /* The main() function will return 0 if the program successfully completes its
        execution; otherwise, some other non-zero integer will be returned, showing a
        failure.*/
}
```

- The function doesn't accept any parameters.
- An instance or object of the class is created here, and all the necessary function calls are made using this object.

5. The final step in the algorithm involves a crucial step i.e., plotting the data saved in the file by the calc() function using **Gnuplot software**.
   Gnuplot is a graphing utility for Linux, OS/2, MS Windows, OSX, VMS, and many other platforms. Its source code is freely distributed and is extensively used for data visualization.
   Follow the following steps to plot data on Gnuplot:

   (a) Download Gnuplot.

   (b) Go to File → Change directory.
   Select the folder where the 'proj_ectory.txt' data file is stored.

   (c) Write the following command in Gnuplot:

```
plot "proj_trajectory.txt" u 1:2 w l
\* This command asks the Gnuplot to plot Column 2 with respect to Column 1
    with a line-type graph. There are many other variations to this command,
    you can check them out at the official website of Gnuplot or watch some
    free tutorials on YouTube.*/
```