Advanced Analysis of Projectile Motion: A Comprehensive Study

Christopher Whitfeld   Arnav Nagpure

Abstract

This paper presents a comprehensive study of projectile motion, utilizing a series of tasks designed to delve deeply into the mathematics and physics of projectiles. Starting with the foundational principles, we derive key equations governing the motion of a projectile under the influence of gravity, neglecting air resistance. The study extends to the exploration of range, time of flight, and maximum height, providing analytical and numerical solutions where appropriate. We also examine the impact of varying parameters such as launch angle and initial velocity. This research offers a detailed walkthrough of the mathematics involved, emphasizing both theoretical and practical aspects of projectile motion.

tc

1 Introduction

Projectile motion, a classical topic in physics, provides a rich ground for exploring the principles of kinematics and dynamics. It involves the motion of an object thrown or projected into the air, subject only to the acceleration due to gravity. This study aims to provide a detailed mathematical exploration of projectile motion, focusing on a series of tasks designed to challenge and expand our understanding of the subject.

The tasks, ranging from the basic principles to more complex scenarios involving varying launch angles and velocities, are examined in depth. The paper seeks to derive key equations, analyse their implications, and explore the behavior of projectiles under different conditions. By solving these tasks, we aim to offer insights into the mechanics of projectiles, contributing to a deeper understanding of the topic.

We hope that our standard and extension work, demonstrates our understanding and passion for computational physics. Throughout, we used either Python or Javascript to complete standard tasks 1-9. For extensions, we also used Unity and C#, as well as HTML/CSS, React and TypeScript.

In: M. Meder, A. Rapp, T. Plumbaum, and F. Hopfgartner (eds.): Proceedings of the Data-Driven Gamification Design Workshop, Tampere, Finland, 20-September-2017, published at http://ceur-ws.org

2 Task 1: Simple Model

2.1 Introduction

The objective of this study is to create a computational model for drag-free projectile motion using a programming language or spreadsheet software. The model allows for real-time updates to the projectile's trajectory based on user-defined inputs: launch angle θ, gravitational acceleration g, launch speed u, and launch height h.

2.2 Model Development

The projectile motion is governed by the following equations of motion under the assumption of no air resistance:

2.2.1 Initial Conditions

The initial velocity components are determined as:

$$v_{x_0} = u\cos\theta$$

$$v_{y_0} = u\sin\theta$$

where θ is the launch angle converted to radians.

2.2.2 Equations of Motion

The position of the projectile at any time t is given by:

$$x(t) = v_{x_0} \cdot t$$

$$y(t) = h + v_{y_0} \cdot t - \frac{1}{2}gt^2$$

2.2.3 Time of Flight

The time of flight T is determined by solving $y(T) = 0$ using:

$$T = \frac{-v_{y_0} - \sqrt{v_{y_0}^2 + 2gh}}{-g}$$

2.3 Implementation

The model is implemented with a fixed time increment Δt, iterating through time steps to calculate and plot $x(t)$ and $y(t)$. The graph of the projectile's trajectory updates automatically when any input parameter is changed.

3 Task 2: Analytical Model

3.1 Horizontal Range

The maximum horizontal range R of the projectile is given by:

$$R = \frac{u^2 \sin(2\theta)}{g} + \frac{u \cos\theta}{g}\sqrt{u^2 \sin^2\theta + 2gh}$$

where u is the initial speed, θ is the launch angle, and h is the launch height.

3.2 Projectile Trajectory

The trajectory $y(x)$ as a function of the horizontal distance x is derived from the equations of motion:

$$y(x) = h + x\tan\theta - \frac{gx^2}{2u^2\cos^2\theta}$$

where x ranges from 0 to R.

3.3 Apogee Calculation

The apogee (maximum height $y_{\max}$) is found by evaluating $y(x)$ at $x = \frac{R}{2}$:

$$y_{\max} = h + \frac{u^2 \sin^2\theta}{2g}$$

3.4 Implementation

Define an equally spaced array of x values from 0 to R. Calculate the corresponding $y(x)$ values using the above equations and plot the trajectory. Highlight the apogee point on the graph to show the maximum height achieved.

4 Task 3: Projectile Model for a Fixed Target Position

This task involves creating a projectile model where the projectile is launched from the origin $(0, 0)$ and must pass through a fixed target position (X, Y). The goal is to calculate the minimum launch speed required to reach the target and determine the two possible trajectories: the 'low ball' and 'high ball' paths.

4.1 Model Development

4.2 Minimum Launch Speed

The minimum launch speed $u_{\min}$ required to reach the target (X, Y) is given by:

$$u_{\min} \geq \sqrt{g\left(\frac{Y}{X} + \sqrt{1 + \left(\frac{Y}{X}\right)^2}\right)}$$

where g is the gravitational acceleration.

4.3 Trajectory Calculation

The trajectory equation $y(x)$ that passes through the point (X, Y) is:

$$Y = h + X\tan\theta - \frac{gX^2}{2u^2\cos^2\theta}$$

Given $h = 0$, this simplifies to:

$$Y = X\tan\theta - \frac{gX^2}{2u^2\cos^2\theta}$$

4.4 Quadratic Form and Solutions for θ

The above equation can be rearranged into a quadratic form:

$$a\tan^2\theta + b\tan\theta + c = 0$$

where:

$$a = \frac{gX^2}{2u^2}, \quad b = -X, \quad c = Y$$

The two possible solutions for θ, corresponding to the 'low ball' and 'high ball' trajectories, are given by:

$$\theta_{\pm} = \tan^{-1}\left(\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}\right)$$

4.5 Implementation

Using the derived equations, calculate the launch speed and angles for both trajectories. Define an array of x values, and use the angle and speed to compute the trajectory $y(x)$ for both the 'low ball' and 'high ball' paths. Plot these trajectories on the same graph, indicating the fixed target position (X, Y).

5 Task 4: Comparing Trajectories with Maximum Horizontal Range

In this task, we develop a projectile model to compare a given trajectory with the trajectory that maximizes the horizontal range for the same initial conditions. The inputs to the model include the initial speed u, launch height h, gravitational acceleration g, and launch angle θ. We calculate the optimal launch angle $\theta_{\max}$ for maximizing the range, which differs from 45° when the launch height is non-zero.

5.1 Model Development

5.1.1 Maximum Range Calculation

The maximum horizontal range $R_{\max}$ for a projectile launched from a height h is given by:

$$R_{\max} = \frac{u^2}{g}\sqrt{1 + \frac{2gh}{u^2}}$$

This equation accounts for the fact that the launch height h influences the maximum range.

5.1.2 Optimal Launch Angle

The optimal launch angle $\theta_{\max}$ for achieving maximum range is determined using:

$$\theta_{\max} = \sin^{-1}\left(\frac{1}{\sqrt{2 + \frac{2gh}{u^2}}}\right)$$

This formula accounts for the effect of the initial height on the optimal launch angle, which deviates from 45° when $h > 0$.

5.1.3 Trajectory Equations

For any given launch angle θ, the trajectory of the projectile is given by:

$$y(x) = h + x\tan\theta - \frac{gx^2}{2u^2\cos^2\theta}$$

For the trajectory corresponding to the optimal launch angle $\theta_{\max}$, the same equation applies with $\theta = \theta_{\max}$.

5.2 Implementation

5.2.1 Step 1: Input Parameters

First, accept the user-defined inputs: initial speed u, launch height h, gravitational acceleration g, and launch angle θ. Compute $R_{\max}$ and $\theta_{\max}$ using the equations provided.

5.2.2 Step 2: Trajectory Calculation

Using the trajectory equation, calculate the $y(x)$ values for both the user-defined launch angle θ and the optimal launch angle $\theta_{\max}$. Define an array of x values from 0 to the greater of the two ranges.

5.2.3 Step 3: Plotting the Results

Plot both trajectories on the same graph for comparison. Highlight the point of maximum range $R_{\max}$ and indicate the corresponding optimal angle $\theta_{\max}$. This comparison allows visualization of how different launch angles affect the range and trajectory shape.

6 Task 5: Enhanced Projectile Model with Bounding Parabola

In this task, the projectile model is enhanced to include a bounding parabola, which defines the region of possible (X, Y) coordinates that can be reached by the projectile for given inputs u (initial speed), h (launch height), and g (gravitational acceleration). Additionally, the model will include the minimum speed, maximum range, and high and low ball trajectories.

6.1 Model Development

6.1.1 Bounding Parabola Equation

The bounding parabola is the locus of all points (X, Y) that can be reached by the projectile given the inputs u, h, and g. The equation of the bounding parabola is:

$$y = \frac{u^2}{2g} - \frac{g}{2u^2}x^2$$

This parabola represents the theoretical maximum height y that can be achieved for any horizontal distance x.

6.1.2 Incorporating the Bounding Parabola

To visualize the region of possible trajectories, plot the bounding parabola on the same graph as the minimum speed, maximum range, and high and low ball trajectories. The bounding parabola serves as a boundary beyond which the projectile cannot pass for the given initial conditions.

7 Task 8: Bouncing Ball Simulation

This section outlines the key features of the simulation and visualization of a bouncing ball, incorporating the effects of gravity and restitution.

7.1 Constants and Parameters

Coefficient of restitution: $e = 0.3$ (determines the energy loss upon impact) Time step: $\Delta t = 0.01\,\mathrm{s}$

7.2 Equations of Motion

The simulation updates the position and velocity of the ball as follows:

- Height update:

$$h = h + v\Delta t \tag{1}$$

- Vertical velocity update:

$$v = v - g\Delta t \tag{2}$$

- Horizontal displacement update:

$$x = x + v_x\Delta t \tag{3}$$

7.3 Collision Handling

When the ball hits the ground ($h \leq 0$), the height is reset to zero, and the vertical velocity is updated using the coefficient of restitution:

$$v = -e \cdot v \tag{4}$$

The simulation continues until the ball has bounced three times.

7.4 Visualization

The trajectory is visualized using 'matplotlib' with animation:

- Plot Setup: The plot includes horizontal and vertical displacements, with appropriate axis labels and titles.
- Animation: A 'FuncAnimation' object animates the ball's motion and trajectory, showing real-time updates of the ball's position and trajectory path.

The resulting plot illustrates the path of the bouncing ball and visualizes the effects of multiple collisions due to the coefficient of restitution.

8 Task 9: A drag model of air resistance

Compared to previous tasks, this was relatively simple. Incorporating air resistance is a case of adjusting acceleration equations with air resistance coefficients.

8.1 Constants and Parameters

Gravitational acceleration: $g = 9.81\,\mathrm{m/s^2}$ Time step: $\Delta t = 0.01\,\mathrm{s}$ Air resistance coefficient: $k = 0.005\,\mathrm{kg^{-1}m^{-1}}$ (chosen to match the competition organizer's parameters)

8.2 Drag Model

In the presence of air resistance, the acceleration components are given by:

$$a_x = -\frac{kv_x}{m}v \tag{5}$$

$$a_y = -g - \frac{kv_y}{m}v \tag{6}$$

where $v = \sqrt{v_x^2 + v_y^2}$ is the speed of the projectile.

8.3 Numerical Integration

With Drag: Positions and velocities are updated iteratively, incorporating air resistance effects on both components.

9 Graphing and GUI Optimisations

9.1 WebAssembly Integration

To maximise the efficiency, both in terms of resources and speed, we opted to use WebAssembly which compiled Rust code into a highly performant, browser-compatible format, enabling the complex physics simulations to be executed efficiently on the client side. To bind the two environments together we compiled the rust using Cargo with a compile target set to

```
wasm32-unknown-unknown
```

stating a compilation target for a 32-bit WebAssembly platform (wasm32), without any specific operating system (unknown) or application binary interface (unknown).

9.2 Progressive Web Application

The project was developed as a Progressive Web App (PWA) to provide a seamless, app-like experience across all devices. This has allowed us to build an iOS and Android application offering offline capabilities through service workers that cache key resources for quick loading and functionality. The WebAssembly-based physics simulations were integrated directly into the PWA, ensuring high performance while providing a native-like experience.

9.3 Further Optimisations

The **Ramer-Douglas-Peuker** algorithm was employed to optimize the graphing of the projectile trajectories by hugely pruning the number of points in the path without significantly altering its shape. We opted to prune around about 70% of points to strike a balance between computational speed and resource efficiency while also preserving the main shape of the graph itself.

This algorithm works by selectively removing points from the original dataset essentially decimating the curve by removing line segments focusing on preserving the critical points that define the overall structure of the curve. This reduction in points allowed for smoother rendering and faster updates in the browser, enhancing the user experience when interacting with the simulation.

10 Acknowledgements

Use a third level heading for the acknowledgements. All acknowledgements go at the end of the paper.

References

[Com79] D. Comer. The ubiquitous b-tree. *Computing Surveys*, 11(2):121–137, June 1979.

[Knu73] D. E. Knuth. *The Art of Computer Programming – Volume 3 / Sorting and Searching.* Addison-Wesley, 1973.