PRACTICE PROJECT · COMPUTATIONAL PHYSICS

# The Enhanced Projectile Motion Model

*Atmospheric Density Modeled with Observed Data*

**Abir Hossain**

March 21, 2026

Keywords: computational physics · data modeling · projectile motion · atmospheric drag

# Contents

## 1. Executive Summary

Using actual observational data, this research adds the mechanics of atmospheric drag to the traditional projectile motion model. The **U.S. Standard Atmosphere 1976 (NASA)** provided the atmospheric density data, covering elevations from sea level up to 50 km — a range that easily encompasses conventional projectile trajectories.

Physically implausible conditions are assumed by the ideal projectile model. The simulation generates trajectories more in line with reality by modeling **air density as a function of altitude** and including it in the drag force calculation. The entire process is documented in this report, from sourcing and cleaning data through creating the atmospheric density model to producing the final improved simulation.

The findings demonstrate that computational modeling and data science methods can enhance a basic physical model.

## 2. Purpose of This Project

This project was started as a self-practice exercise in **computational data modeling**, specifically integrating standard physics with real-world observational data to enhance an idealized model. This was a conscious step toward numerical mathematical modeling in Python, as prior work had been focused on data science.

The projectile motion problem was an obvious starting point because it is theoretically well-understood, making it simple to gauge how much the simulation improves with the addition of real-world physics. Drag force — directly dependent on atmospheric density — is one of the easiest modifications to the ideal model.

## 3. Project Structure

```
├── clean_data
├── cleaning_scripts
├── raw_data
├── results
└── src
```

> All source code and data files can be found here on GitHub.

The project follows three sequential stages:

→ **Establishing the ideal model** — solve the projectile ODE system numerically under vacuum conditions and verify it produces expected parabolic trajectories.
→ **Build an atmospheric density model** — fit a function $\rho(h)$ to the NASA observational data.
→ **Combine** — substitute $\rho(h)$ into the drag force term of the ideal model and re-simulate.

# 4. The Ideal Projectile Model

## 4.1. Physics Background

In classical physics, a projectile moves under gravity alone with no air resistance. The equations of motion including drag are set up below, with the ideal case achieved by setting $F_d = 0$.

## 4.2. Equations of Motion

Let the projectile have mass $m$, horizontal velocity $V_x$, vertical velocity $V_y$, and total speed:

$$V = \sqrt{V_x^2 + V_y^2}$$

The aerodynamic drag force acts opposite to the direction of motion:

$$F_d = \frac{1}{2}\rho C_d A V^2$$

where $\rho$ is the local air density, $C_d$ is the drag coefficient, and $A$ is the cross-sectional area. The coupled ODEs governing the motion are:

$$\frac{dV_x}{dt} = -\frac{\rho C_d A}{2m} \cdot V \cdot V_x$$

$$\frac{dV_y}{dt} = -g - \frac{\rho C_d A}{2m} \cdot V \cdot V_y$$

with kinematic relations:

$$\frac{dx}{dt} = V_x$$

$$\frac{dy}{dt} = V_y$$

In the ideal (vacuum) case, $F_d = 0$, so the equations reduce to:

$$\frac{dV_x}{dt} = 0$$

$\frac{dV_y}{dt} = -g$

## 4.3. Initial Conditions

The projectile is launched at initial speed $V_0$ and angle $\theta$ from the horizontal:

$$V_{x_0} = V_0 \cos\theta$$

$$V_{y_0} = V_0 \sin\theta$$

## 4.4. Numerical Solution: RK4

The ODE system is solved numerically using the **fourth-order Runge-Kutta method (RK4)**. Rather than stepping forward with a single slope estimate, RK4 computes four slope estimates per step:

$$k_1 = f(t_n, y_n)$$

$$k_2 = f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right)$$

$$k_3 = f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2\right)$$

$$k_4 = f(t_n + h, y_n + hk_3)$$

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

This gives fourth-order accuracy in the step size. The implementation is in `ideal.py` in the `src` directory.

## 4.5. Ideal Trajectory

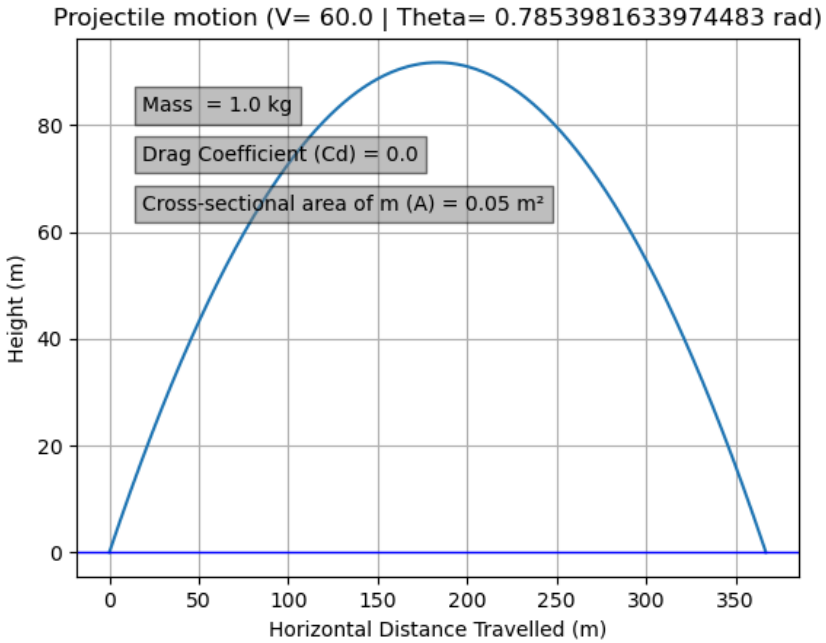


Figure 1: Ideal projectile trajectory under vacuum conditions — parabolic, as expected.

## 4.6. Parametric Cases

Several scenarios were tested by varying $V_0$, $\theta$, $m$, $C_d$, $A$, etc., to verify the reliability of the code.

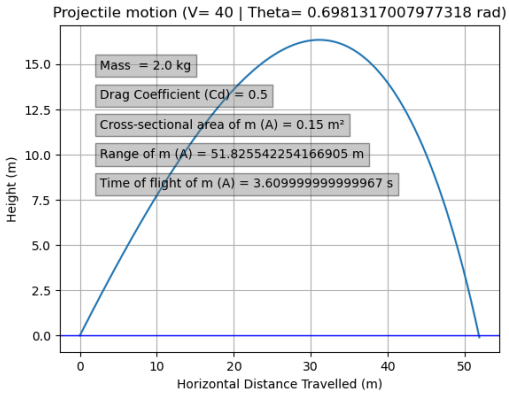


Figure 2: Case 1

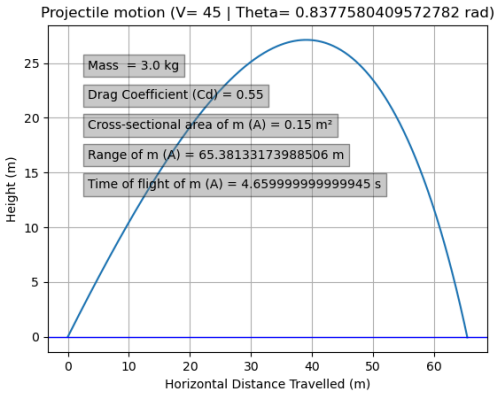


Figure 3: Case 2

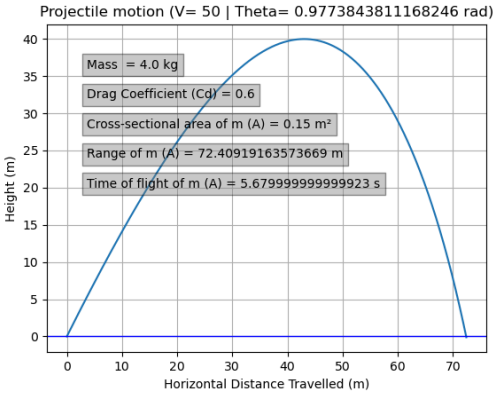


Figure 4: Case 3

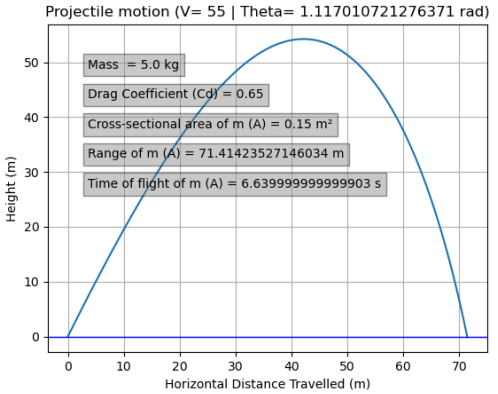


Figure 5: Case 4

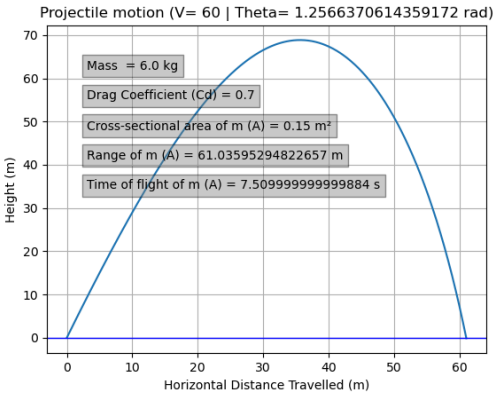


Figure 6: Case 5

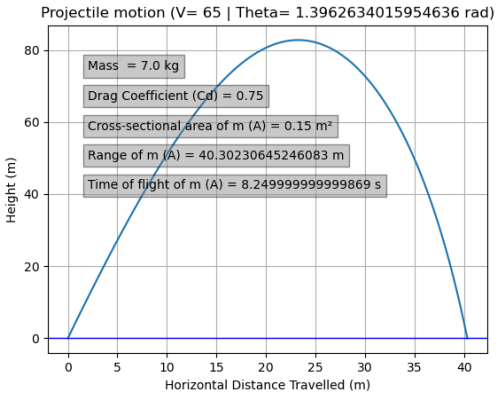


Figure 7: Case 6

All varying conditions are reported in the respective plots.

# 5. Data Sourcing and Preparation

## 5.1. Source

The official NASA U.S. Standard Atmosphere 1976 document exists as a PDF, but the scanned quality made direct extraction impractical. The data was instead sourced manually from a third-party website that reproduced the NASA tabular values in a clean,

structured format. Parameters extracted: altitude (per 200 metres), air density (kg/m³), and dynamic viscosity (Pa · s), from sea level up to 50 km. The raw data was saved as us_atmosphere_NASA.csv in the raw_data directory.

### 5.2. Cleaning

The cleaning script clean_us_NASA.py in the cleaning_scripts directory standardized column names and reformatted the data, outputting atmosphere_nasa_clean.csv to the clean_data directory.

## 6. The Atmospheric Density Model

### 6.1. Observed Shape of the Data

The cleaned data was first plotted using plot_test.py to examine its structure:

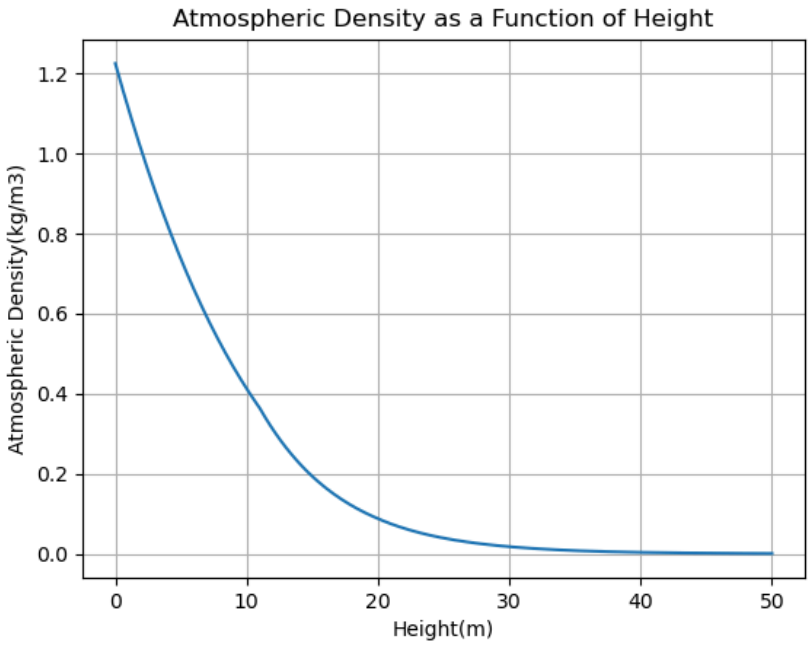


Figure 8: Air density as a function of altitude from the NASA dataset.

The density decreases rapidly and smoothly with altitude — consistent with a **decaying exponential**, following from the **barometric formula** derived from hydrostatic equilibrium and the ideal gas law.

### 6.2. Mathematical Ground

#### 6.2.1. Exponential Base Model

The standard barometric approximation for density as a function of altitude $h$ is:

$$\rho_{\text{base}(h)} = \rho_0 e^{-h/H}$$

where $\rho_0 = 1.225\text{kg/m}^3$ is the sea-level density and $H = 8500$ m is the atmospheric scale height.

> Notice $H = 8500$ m. The collected data and the target range far exceed that. Although the effects are almost negligible.

Real atmospheric density deviates from a single-scale-height exponential due to temperature stratification across different layers — the troposphere, stratosphere, and so on.

#### 6.2.2. More Rigorous Alternatives

For reference, more physically complete models exist:

→ **The International Standard Atmosphere (ISA)** — a piecewise model with distinct temperature lapse rates per atmospheric layer.

→ **NRLMSISE-00** — an empirical model accounting for solar activity, geographic location, and time of day. Commonly used in orbital mechanics and atmospheric re-entry calculations.

> Those more rigorous models could also be used as per necessity.

#### 6.2.3. Hybrid Correction: Residual Fitting

Rather than replacing the base exponential model, a **residual correction** was applied. The residual at each altitude is:

$$\varepsilon(h) = \rho_{\text{observed}(h)} - \rho_{\text{base}(h)}$$

A **cubic spline** $\hat{\varepsilon}(h)$ is then fitted to these residuals using piecewise cubic polynomials with the "not-a-knot" boundary condition. The final hybrid density model is:

$$\rho_{\text{hybrid}(h)} = \rho_{\text{base}(h)} + \hat{\varepsilon}(h)$$

By construction, $\rho_{\text{hybrid}}$ passes exactly through every data point in the cleaned NASA dataset.

### 6.3. Application

The hybrid density was coded and plotted using `atmosphere_model.py`:

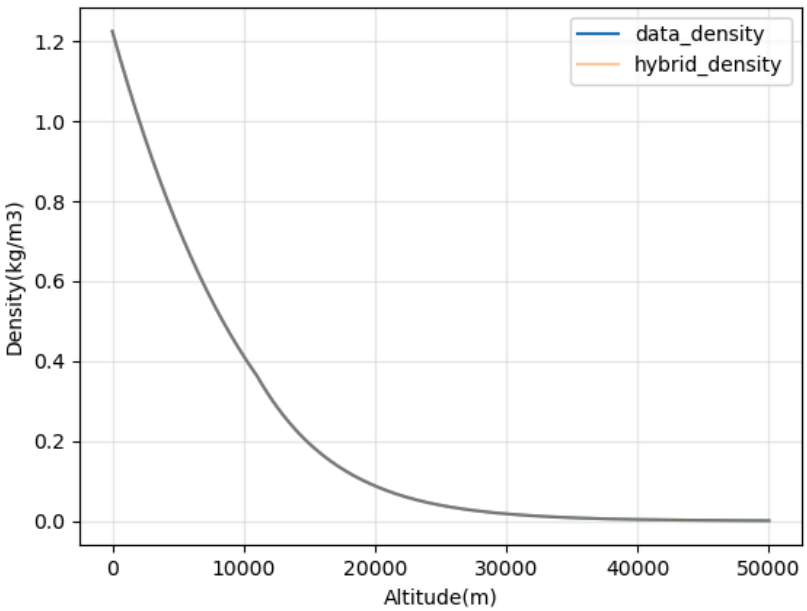


Figure 9: Hybrid density model overlaid on observed NASA data. The nearly perfect overlap between the two curves results from the model tracking the data exactly.

> Notice how the 2 differently colored lines merged almost perfectly to create a greyish locus!

The hybrid atmospheric density function was then passed directly into `rho_hybrid_projectile.py` to create the improved projectile simulation.

## 7. Enhanced Projectile Simulation

With $\rho_{\text{hybrid}(h)}$ defined, it was substituted directly into the drag force term:

$$F_{d(h)} = \frac{1}{2}\rho_{\text{hybrid}(h)} C_d A V^2$$

Since $\rho$ is now a function of the projectile's current altitude $y(t)$, the drag force varies continuously throughout the flight. The RK4 integrator handles this naturally, re-evaluating the force at each sub-step.

### 7.1. Result

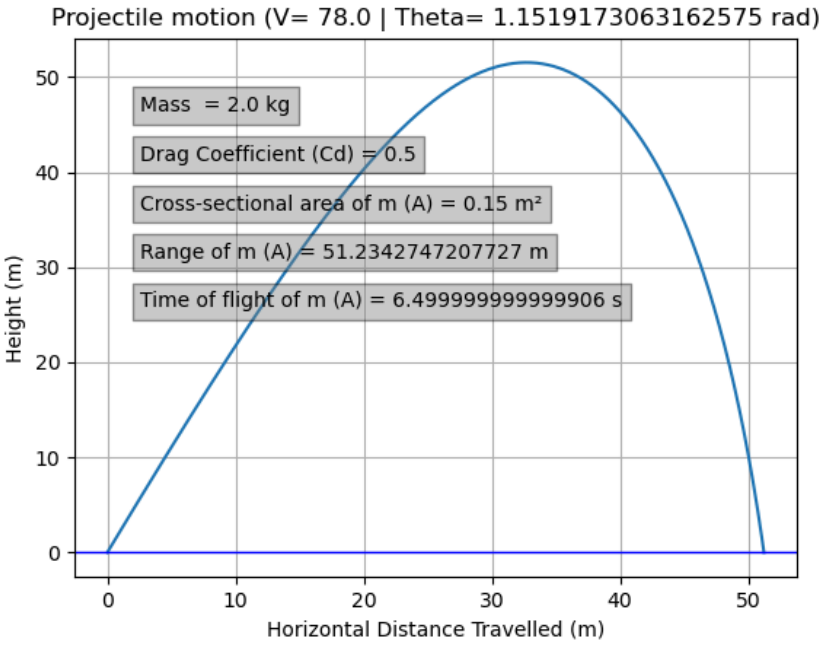


Figure 10: Enhanced projectile trajectory with atmospheric drag. The trajectory is noticeably shorter and lower than the ideal parabola.

The drag effect is most pronounced at high speeds — consistent with the $V^2$ dependence of $F_d$ — and diminishes as the projectile slows.

### 7.2. Possible Extensions

→ **Drag coefficient $C_d$** — assumed constant throughout. In reality, $C_d$ depends on Mach and Reynolds numbers, both of which change during flight.

→ **Cross-sectional area $A$** — assumed fixed. For a tumbling or spinning projectile, the effective presented area changes over time.

→ **Dynamic viscosity** — included in the NASA dataset but unused. Becomes relevant in low-speed, low-altitude regimes.

→ **Wind** — a horizontal wind profile $u(h)$ could be incorporated by adjusting the effective horizontal velocity at each altitude.
→ **Extended altitude range** — above 50 km, a layered ISA or NRLMSISE-00 model would be required.
→ **3D motion** — the current model is two-dimensional. Real trajectories include lateral drift from the Coriolis effect and crosswinds.

## 8. Author's Note

Computational modeling has become a standard tool across science and engineering — not as a replacement for theory or experiment, but as a bridge between the two:

→ **Analytical limits** — many realistic physical systems have no closed-form solution. Numerical methods can solve them to any required precision.
→ **Experimental cost** — physical experiments are often slow and expensive. A validated computational model can explore a large parameter space at a fraction of the cost.
→ **Theory-experiment gaps** — simulations can quantify the discrepancy between theoretical predictions and observed measurements.
→ **Engineering design** — simulations allow iterative refinement before anything is physically built.

This project demonstrates the same pattern at a small scale: take an idealized model, identify where it diverges from reality, source the relevant data, build a correction, and re-simulate. The same approach applies broadly to fluid dynamics, biochemical kinetics, ecological modeling, socioeconomic modeling, financial modeling, and more.

> This project is asked not to be regarded as scientific-research work and this report is not to be judged as a scientific paper. The project was undertaken as a practice exercise in computational physics. I hoped to use conventional tools and techniques from data science workflows in a physics model, driven by the curiosity established from my undergrad in the subject. And I really enjoyed doing it.