Artillery Trajectory Modeling via Simplified Aerodynamic-Kinematic Assumptions and Development of Manual Calculation Tools

Abstract

This study establishes a theoretical framework for artillery **projectile trajectory modeling** based on simplified aerodynamic and kinematic assumptions, systematically describing projectile flight behavior under constrained conditions.

For the first Question, we built a **2D planar motion-based ODE system**, integrating key trajectory-influencing physical models (atmospheric model, **dynamic drag-coefficient model**). With the **Dormand-Prince** numerical integration algorithm, we simulated the projectile's flight. Via parameter scanning and optimization, we obtained 100 elevation angle (θ)-initial velocity pairs ensuring target impact, and extracted three typical solutions. Among these, the minimum initial velocity is 122 m/s (at $\theta = 41.8°$). Overall, θ and initial velocity show a **U-shaped** relationship: the latter is smallest at 40°-50°, consistent with ballistics' minimum energy principle, verifying the model's rationality.

For the second Question,within 1000–1500 meters (varying altitudes/wind), we built a physical model incorporating gravity, air resistance, sphere damping coefficient, and air density variations. Ignoring Coriolis force and Magnus effect, we used **Runge-Kutta** (ODEs), **Levenberg-Marquardt** (least squares) with three-point finite difference **Jacobian approximation** to get optimal launch angle/velocity. Wind affects azimuth, distance impacts initial velocity, and relative altitude influences pitch angle—these interrelated factors induce complex **nonlinear** changes, requiring precise adjustments in actual combat.

For the third Question, this paper first derives a preliminary **base ballistic table** for idealized scenarios via real-world simplification, then obtains **correction tables** from the final practical projectile flight model developed in Question 2. Additionally, it develops a nomogram suitable for artillery officers' manual plotting from initial firing tables. Finally, it derives scenario-specific **simplified formulas** from the simplified model to assist officers in mental calculations, with results corrected via practical experience and the correction tables.

Keywords: ODE system; Runge-Kutta method;Levenberg-Marquardt;Base Ballistic Table;Correction Table; Nomogram

Contents

1 Introduction

In the 19th century , artillery warfare represented a critical component of military operations, yet artillery officers faced severe computational constraints. Operating without computers, calculators, or even slide rules in many cases, these officers relied solely on mathematical training, logarithm tables, and trigonometric tables to make life-or-death firing decisions under battlefield pressure. The cannons of this era fired spherical cast-iron projectiles. Combat scenarios varied dramatically—engagements occurred across mountainous terrain with significant elevation differences, in valleys, and on plains, all while contending with unpredictable wind conditions. The challenge was not merely theoretical accuracy but practical speed: an artillery officer needed to calculate firing solutions within minutes while under enemy fire, making the difference between victory and defeat.

2 Problem Restatement

This problem requires developing a practical manual calculation methodology for 19th-century artillery operations in three progressive stages. First, determine the precise muzzle velocity and firing angle needed to hit a specific target located 1,200 meters away horizontally, with both the cannon and target positioned at 500 meters above sea level. Second, extend this solution to create a systematic approach for engaging targets at horizontal distances ranging from 1,000 to 1,500 meters, accounting for various altitude differences between cannon and target positions, as well as different wind speeds and directions. Third, generalize the method to cover the cannon's full operational range under diverse combat conditions. The final deliverable must be a rapid estimation technique—such as simplified firing tables, graphical aids, or calculation shortcuts—that a mathematically-trained officer can execute in the field using only period-appropriate tools (pen, paper, and reference tables) while maintaining tactically acceptable accuracy for successful target engagement.

3 Notation

Symbol	Definition	Unit
1. Basic Physical Quantities		
ρ	Air density	$\mathrm{kg/m^3}$
μ	Dynamic viscosity of air	$\mathrm{Pa \cdot s}$
A	Cross-sectional area of the projectile ($A = \pi(d/2)^2$)	$\mathrm{m^2}$
2. Aerodynamic Parameters		
C_d	Drag coefficient	Dimensionless
$C_{d\text{-base}}$	Re-based drag coefficient	Dimensionless
$C_{d\text{-Ma-factor}}$	Ma correction factor for C_d	Dimensionless
w_{Re}	Re-Ma coupling weight	Dimensionless

Continued on next page

Symbol	Definition	Unit
$C_{d\text{-Ma-only}}$	High-Re baseline C_d	Dimensionless
F_d	Aerodynamic drag force	N
$F_{d,y}$	Vertical component of aerodynamic drag force	N
k	Drag factor ($k = \frac{1}{2}C_d\rho A v_{\text{rel}}/m$, used in ODE system)	1/m
Re	Reynolds number ($\text{Re} = \frac{\rho d v_{\text{rel}}}{\mu}$, characterizes viscous-inertial balance)	Dimensionless
Ma	Mach number	Dimensionless
v_{rel}	Relative wind speed	m/s
3. Ballistic and Trajectory Parameters		
x, y, z	Coordinates in East-North-Up (ENU) system: x (East), y (North), z (Up)	m
v_x, v_y, v_z	Velocity components in x, y, z directions	m/s
v_0	Initial muzzle velocity	m/s
θ	Launch elevation angle	degrees
φ	Launch azimuth angle	degrees
R	Horizontal range	m
t	Flight time	s
h	Altitude	m
Δh	Altitude difference between launch point and target	m
v_w	Wind speed (vector components: v_{wx} (East), v_{wy} (North), v_{wz} (Up))	m/s
$\vec{Y}$	2D state vector	-
4. Model Constants and Coefficients		
ρ_0	Standard air density at sea level (ISA model)	kg/m^3
μ_0	Standard dynamic viscosity at sea level ($T_0 = 288.15$ K)	$\text{Pa} \cdot \text{s}$
T_0	Standard sea level temperature (ISA model)	K
L	Temperature lapse rate ($L = 0.0065$ K/m)	K/m
H_p	Density scale height ($H_p \approx 8500$ m)	m
S	Sutherland constant ($S = 110.4$ K)	K
R_{gas}	Specific gas constant for dry air	$\text{J/(kg} \cdot \text{K)}$
R_{earth}	Earth's radius	m
ω_{earth}	Earth's angular velocity	rad/s
λ	Latitude	degrees
5. Numerical Integration Parameters		
h_{step}	Time step size	s
RelTol	Relative error tolerance	Dimensionless
AbsTol	Absolute error tolerance	Dimensionless
ϵ	Desired error tolerance	Dimensionless

4 Assumptions and Approximations

→ **Assumption 1:** Treat the artillery projectile as a rigid sphere.

Justification: Aligns with historical spherical projectile structure and simplifies aerodynamic analysis by avoiding irregular shape - induced complex flow calculations.

→ **Assumption 2:** Neglect Coriolis force and Earth's curvature.

Justification: Their trajectory deviation is negligible compared to air drag and wind force for short - flight, limited - range projectiles, meeting battlefield rapid calculation needs.

→ **Assumption 3:** Neglect the Magnus effect.

Justification: Aerodynamic drag dominates at high speeds, making the Magnus force's cumulative effect insignificant and simplifying 3D force analysis.

→ **Assumption 4:** Atmospheric parameters vary regularly with altitude.

Justification: Based on the standard atmospheric model, it quantifies altitude's impact on aerodynamic drag and conforms to actual atmospheric variation.

→ **Assumption 5:** Air drag follows the quadratic drag law.

Justification: Suitable for high - speed projectiles in turbulent flow, accurately describing drag - velocity squared relationship.

→ **Assumption 6:** Treat trajectory evolution as a quasi - static process.

Justification: Fast molecular thermal motion and pressure waves enable timely force and thermal balance, supporting numerical integration and simplifying unsteady analysis.

5 Ballistic Simulation Model for a Spherical Projectile under Calm Wind Conditions

Building upon classical mechanics, we propose an integrated framework for spherical projectile trajectory simulation that incorporates gravity, aerodynamic drag, and atmospheric dynamics. A Reynolds–Mach coupling model for the drag coefficient, enhanced through adaptive atmospheric parameter updates, ensures accurate nonlinear trajectory prediction. The ODE45 numerical integration method is employed to efficiently bridge model development and validation, maintaining both robustness and practical applicability.

The flowchart below summarizes the logical procedures of our approach.

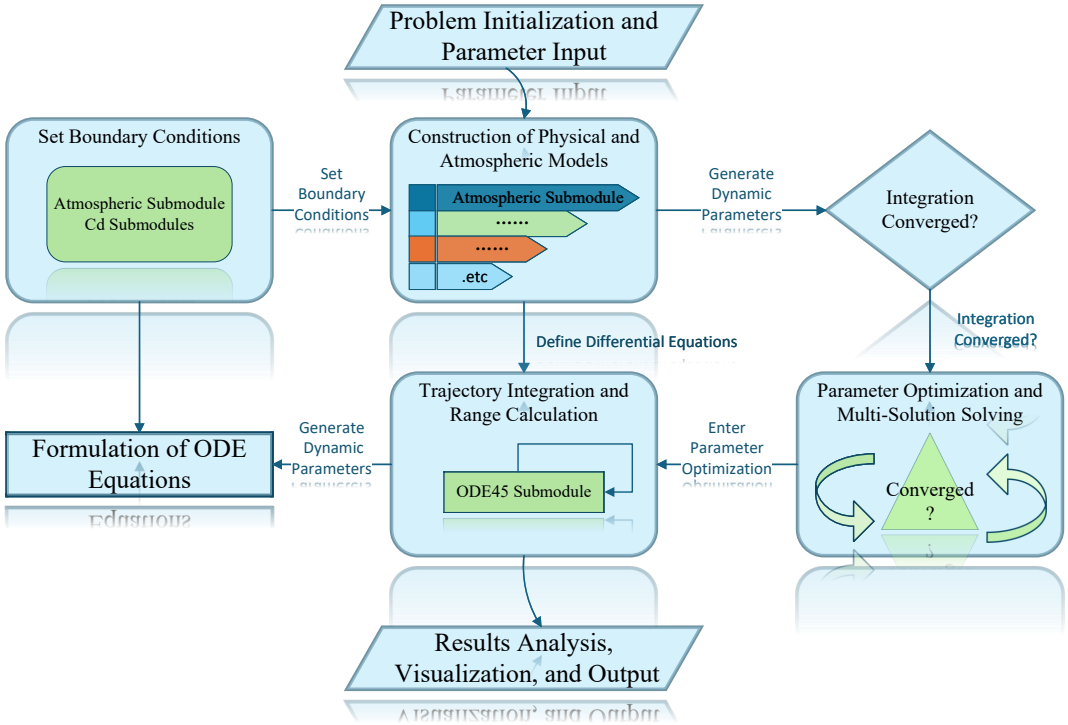


Figure 1: Logical Procedures of The Approach

5.1 The Establishment of Model

5.1.1 Problem Formulation

Firstly, in light of the specific context addressed in this study, we conduct the following analysis and formulate corresponding physical assumptions.
Given that the Earth's rotation induces a Coriolis force, which in turn can cause a deviation in the trajectory of projectile motion, it is necessary to evaluate its potential influence. Previous studies[2] indicate that, under identical conditions, the magnitude of projectile deviation is directly proportional to its velocity. Considering the historical background of this problem—namely, nineteenth-century artillery characterized by relatively short ranges, low muzzle velocities, and short flight durations—the resulting trajectory deviation is expected to be minimal. Consequently, the effects of both the Coriolis force and the Earth's curvature can be reasonably disregarded in this context. Furthermore, as wind effects are not required to be taken into account in the present scenario, the projectile motion can be idealized as occurring within a two-dimensional plane.

The two-dimensional plane motion model is illustrated in the following figure:

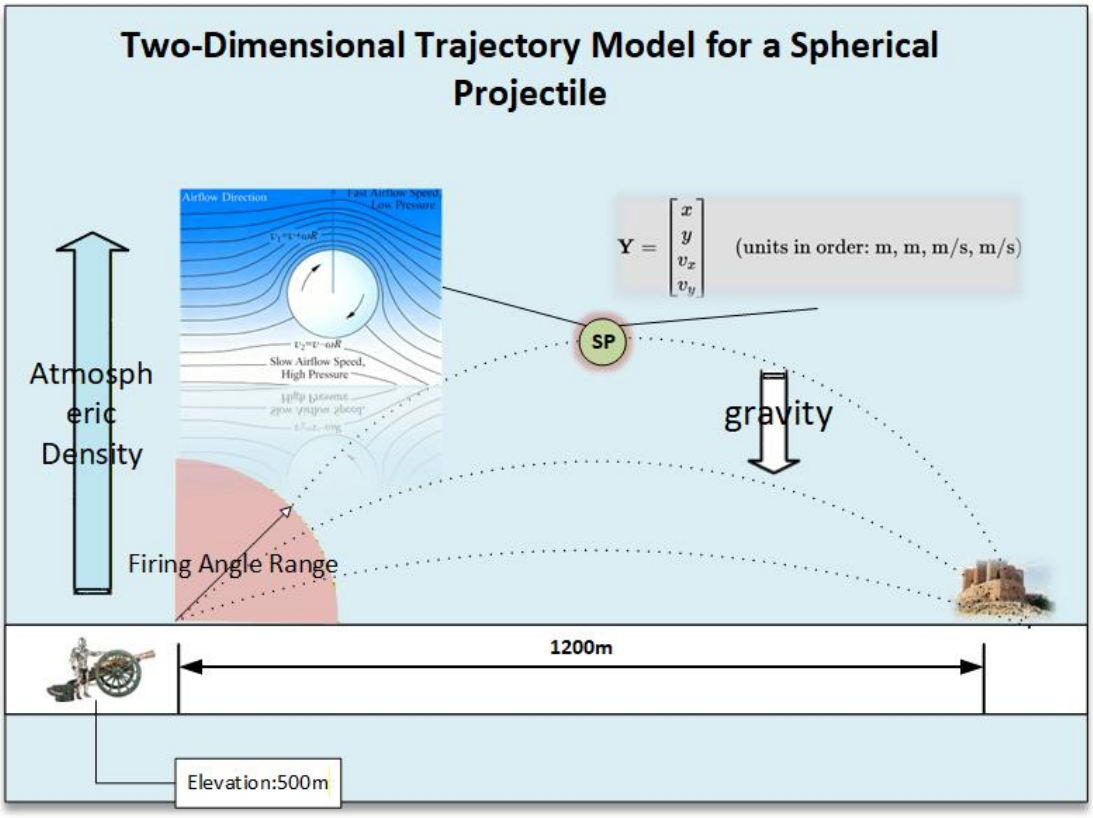


Figure 2: Two-Dimensional Trajectory Model for a Spherical Projectile

We now proceed to provide a detailed explanation of the model depicted above. In the present problem context, the spherical projectile, during its flight through the atmosphere, is subjected to the action of gravity and aerodynamic drag. The aerodynamic drag comprises pressure drag and frictional drag. Given the assumption that the projectile is perfectly smooth, frictional drag is neglected, and only pressure drag is taken into account.

Since the initial launch velocity of the projectile is relatively high—classifying it as a high-speed projectile—the quadratic drag law is adopted for modeling, rather than Stokes' law. The quadratic drag law is expressed as:

$$F_d = \frac{1}{2} C_d \rho A v^2 \tag{5.1}$$

where:

- ρ: Air density
- A: Projected area of the object
- v: Relative fluid velocity
- C_d: Drag coefficient, a dimensionless quantity that highly depends on the object's shape

Regarding the gravitational force acting on the projectile, a constant gravitational field assumption is employed, i.e., g=9.81m/s^2 is considered invariant with altitude. This

assumption is justifiable based on the scientific reality of 19th-century artillery: the limited ranges of cannons and the relatively low altitudes reached by projectiles meant negligible variation in gravitational acceleration with height.

By combining the quadratic drag law with Newton's Second Law of Motion, we can preliminarily establish the state vector and governing equations for the two-dimensional planar motion as follows:

1. **State Vector Definition**

$$\mathbf{Y} = \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix} \quad \text{(Units in order: m, m, m/s, m/s)} \tag{5.2}$$

2. **Kinematics Equations**

$$\frac{dx}{dt} = v_x, \quad \frac{dy}{dt} = v_y \tag{5.3}$$

3. **Dynamics Equations**

$$\frac{dv_x}{dt} = -\frac{1}{2m} C_d \rho A v_{\text{rel}} (v_x - v_w), \quad \frac{dv_y}{dt} = -g - \frac{1}{2m} C_d \rho A v_{\text{rel}} v_y \tag{5.4}$$

Where:

- C_d (drag coefficient, dimensionless)
- ρ (air density, kg/m^3), $A = \pi(d/2)^2$ (cross-sectional area, m^2)
- $v_{\text{rel}} = \sqrt{(v_x - v_w)^2 + v_y^2}$ (relative wind speed, m/s)
- m (projectile mass, kg), v_w (wind speed, m/s)

4. **Formulas for Constant Gravity Field Assumption**

$$g = 9.81\,\text{m/s}^2 \tag{5.5}$$

$$\frac{dv_y}{dt} = -g - \frac{F_{d,y}}{m} \tag{5.6}$$

Where: $F_{d,y} = \frac{1}{2} C_d \rho A v_{\text{rel}} v_y$ (vertical air drag force, N).

This model constitutes the foundational framework for solving the present problem.

5.1.2 Construction of the Atmospheric Model

In the preceding section, we established a two-dimensional planar motion model based on the quadratic drag law. From the governing equation, it can be observed that the magnitude of aerodynamic drag depends not only on the projectile's relative velocity in the fluid, but also on the atmospheric environment and the drag coefficient. In this section, we elaborate on the construction of the atmospheric model.

During the flight, the projectile's altitude changes continuously, and correspondingly, the atmospheric parameters at different altitudes also vary. These variations influence the magnitude of the aerodynamic drag experienced by the projectile. Here, we focus on

two parameters—air density and dynamic viscosity—and describe the methodology by which they are treated.

It is well known that as altitude increases, air becomes progressively thinner. Since F_d=ρ, a density model is required to account for the rarefaction effect at high altitudes. This study adopts the International Standard Atmosphere (ISA) framework, deriving the relevant expressions from the equations of hydrostatic balance and the ideal gas law. By simulating the exponential decay of atmospheric pressure with altitude, this method achieves a balance between computational accuracy and efficiency. The corresponding equation is given as follows:

Basic formula:

$$\rho(h) = \rho_0 \exp\left(-\frac{h}{H_p}\right) \tag{5.7}$$

where

- $\rho_0 = 1.225\,\mathrm{kg/m^3}$ (standard density at sea level)
- $H_p = 8500\,\mathrm{m}$ (density scale height)
- approximately $H_p = \frac{RT_0}{g}$, with $R = 287\,\mathrm{J/(kg \cdot K)}$
- $T_0 = 288.15\,\mathrm{K}$, and $g = 9.81\,\mathrm{m/s^2}$

Lower bound constraint:

$$\rho = \max(\rho, 0.1)\,\mathrm{kg/m^3} \tag{5.8}$$

Table 5.1: Derivation Logic of Atmospheric Pressure and Density Profile

Step	**Derivation Logic**	**Explanations**
1	Starting from the atmospheric hydrostatic equilibrium equation $\frac{dP}{dh} = -\rho g$	P is pressure; gravity induces vertical pressure gradient.
2	Combining with the ideal gas law $P = \rho RT$, leading to $\frac{dP}{P} = -\frac{g}{RT}dh$	Relate pressure to density using the ideal gas state equation.
3	Assuming isothermal condition (T constant) and integrating $P(h) = P_0 \exp\left(-\frac{h}{H}\right)$, where $H = \frac{RT}{g}$	Isothermal assumption simplifies integration to exponential pressure profile.
4	Then deriving the density profile $\rho(h) = \frac{P(h)}{RT} = \rho_0 \exp\left(-\frac{h}{H_p}\right)$ ($H_p \approx H$)	Density profile follows from pressure profile and ideal gas law; H_p approximates H.

Dynamic viscosity captures the frictional effects arising from collisions between air molecules, and exhibits significant dependence on temperature. As altitude increases,

the ambient temperature gradually decreases, leading to marked changes in air viscosity. This variation impacts the computation of the Reynolds number and, consequently, the drag coefficient C_d . To address this, a temperature-dependent viscosity model is constructed based on kinetic theory of gases, deriving the Sutherland formula from the Chapman–Enskog theory. The explicit form of the model is given as follows:

Basic formula:

$$\mu(h) = \mu_0 \left(\frac{T}{T_0}\right)^{\frac{3}{2}} \frac{T_0 + S}{T + S} \tag{5.9}$$

where

- $\mu_0 = 1.789 \times 10^{-5}\,\mathrm{Pa \cdot s}$ (sea level dynamic viscosity)
- $T_0 = 288.15\,\mathrm{K}$
- $S = 110.4\,\mathrm{K}$ (Sutherland constant, representing empirically fitted intermolecular forces)

Lower bound constraint:

$$\mu = \max\left(\mu, 1 \times 10^{-6}\right)\ \mathrm{Pa \cdot s}$$

(to avoid singularities or negative values at low pressure).

Table 5.2: Derivation Logic of Viscosity Formula

Step	**Derivation Logic**	**Formulas**
1	From the kinetic theory of gases foundation	$\mu \propto \rho\lambda\bar{v}$, $\lambda \propto \frac{T}{\rho}$ (mean free path), $\bar{v} \propto \sqrt{T}$ (average speed), leading to $\mu \propto \sqrt{T}$.
2	Considering intermolecular attraction (Lennard-Jones potential)	Corrected to $\mu \propto T^{3/2}$ (collision integral $\Omega \propto T^{-1/2}$) .
3	Introduce Sutherland term to correct low-temperature behavior	Full form from collision theory: $\mu = \mu_0 \left(\frac{T}{T_0}\right)^{3/2} \frac{T_0+S}{T+S}$.

Together, these two models incorporate the influence of altitude on projectile aerodynamics, thereby enhancing the overall accuracy of the motion model.

5.1.3 Dynamic Drag Coefficient Model

Within the framework of the quadratic drag law, the drag coefficient C_d plays a crucial role in determining aerodynamic resistance. In the present study, we establish a dynamic model for $C_d$to capture nonlinear aerodynamic effects, thereby enhancing the accuracy of the motion model across different velocities and altitudes.

The necessity of implementing a dynamic C_d arises from the multi-scale effects inherent in projectile flight: at low velocities, viscous effects dominate (Reynolds number

dependence), resulting in higher C_d values and rapid deceleration; at high velocities, compressibility effects prevail (Mach number dependence), where shock formation can cause pronounced peaks in C_d. Employing a constant C_d value—such as the classical 0.47—could lead to range errors of several tens of meters or more, particularly in the transonic regime.

Our proposed model is derived from coupled Reynolds number (Re) and Mach number (Ma) formulations, based on fundamental principles in fluid mechanics. The overall procedure is outlined in the following table.

Table 5.3: Flowchart of Drag Coefficient C_d Calculation Steps

Step	**Description**
1	Calculate Re-based $C_{d\text{-base}}$ (Viscous Effect)
2	Calculate Ma Correction Factor $C_{d\text{-Ma-factor}}$ (Compressibility Effect)
3	Introduce Re-Ma Coupling Weight w_{Re}
4	Integrate to Calculate Final C_d , Apply physical constraints to ensure validity

We now present a detailed explanation of its construction.

The Reynolds number (Re) characterizes the balance between viscous and inertial forces, obtained through the non-dimensionalization of the Navier–Stokes equations. Since the drag coefficient does not possess a fixed dimensional relationship with Re, we employ a segmented modeling approach to describe the variation of C_d with Re, as follows:

- For low Re region ($\text{Re} < 10^3$, laminar region):

$$C_{\text{d-base}} = \frac{24}{\text{Re}} + \frac{4.5}{\sqrt{\text{Re}}} + 0.42 \tag{5.10}$$

  (The base Stokes solution is 24/Re, derived from molecular diffusion; $+4.5/\sqrt{\text{Re}}$ is the Oseen correction, accounting for inertial disturbances; +0.42 is an empirical high-order term, derived from sphere drag experiments. Physically, $C_d \sim$ 10–100 at low Re, leading to rapid deceleration .)

- For transition region ($10^3 \leq \text{Re} < 2 \times 10^5$):

$$C_{\text{d-base}} = 0.5 - 0.1 \log_{10}\left(\frac{\text{Re}}{10^3}\right) \tag{5.11}$$

  (Derived from boundary layer separation theory; the logarithmic form arises from the backward shift of the separation point with increasing Re, reducing C_d; 0.5 is the initial value, based on subcritical experiments; the deceleration rate 0.1 comes from curve fitting. Physically, $C_d \sim$ 0.4–0.5 in this region, suitable for medium-speed flight.)

- For high Re region ($\text{Re} \geq 2 \times 10^5$, high Re plateau + drag crisis):

$$C_{\text{d-base}} = 0.47 \tag{5.12}$$

(Plateau region, derived from turbulent boundary layer; the constant 0.47 is an empirical value for spheres.)

If $\text{Re} > 3 \times 10^5$:

$$C_{\text{d-base}} = 0.47 \cdot \left(\frac{0.2}{0.47} + 0.8 \exp\left(-\frac{\text{Re} - 3 \times 10^5}{10^5} \right) \right) \tag{5.13}$$

This modeling strategy allows the low- to mid-velocity regime to be dominated by Re-dependent viscous effects.

The Mach number (Ma) represents compressibility effects, derived from the Euler equations. By constructing a C_d –Ma correction model, we capture the aerodynamic phenomena associated with the transonic regime, including the shock-induced increases in C_d. The model is formulated as follows:

- For $\text{Ma} \leq 0.3$ (incompressible region):

$$C_{d\text{-Ma-factor}} = 1.0 \tag{5.14}$$

(derived from Bernoulli's equation).

- For $0.3 < \text{Ma} < 0.8$ (subsonic):

$$C_{d\text{-Ma-factor}} = \frac{1}{\sqrt{1 - \text{Ma}^2}} \tag{5.15}$$

(Prandtl-Glauert rule, derived from linearized potential flow equation: Compressibility increases pressure gradient, raising C_d; $\beta = \sqrt{1 - \text{Ma}^2}$ is the compressibility factor. Physically, at $\text{Ma} = 0.7$, the factor is ~ 1.4, leading to a slight range reduction.)

- For $0.8 \leq \text{Ma} < 1.0$ (transonic rising region):

$$C_{d\text{-Ma-factor}} = 1.67 + (2.0 - 1.67) \cdot \frac{\text{Ma} - 0.8}{0.2} \tag{5.16}$$

(Linear interpolation, derived from experimental curves: At $\text{Ma} = 0.8$, ~ 1.67 (onset of local shock waves); peak at $\text{Ma} = 1.0$ is 2.0 (full shock wave). Physically, C_d rises sharply in this region, causing significant energy loss and a 15% range reduction.)

- For $1.0 \leq \text{Ma} < 1.2$ (transonic falling region):

$$C_{d\text{-Ma-factor}} = 2.0 - (2.0 - 1.5) \cdot \frac{\text{Ma} - 1.0}{0.2} \tag{5.17}$$

(Linear decrease, derived from shock wave attenuation: At $\text{Ma} = 1.2$, ~ 1.5. Physically, C_d decreases after the shock wave stabilizes.)

- For $\mathrm{Ma} \geq 1.2$ (supersonic region):

$$C_{d\text{-Ma-factor}} = 1.5 + 0.3 \cdot (\mathrm{Ma} - 1.2)^2 \tag{5.18}$$

(Quadratic growth, derived from wave drag theory: Wave drag $\propto \mathrm{Ma}^2 - 1$; the 0.3 coefficient is fitted from NASA data. Physically, C_d rises to $\sim$ 2–3 at high Ma, limiting supersonic range.)

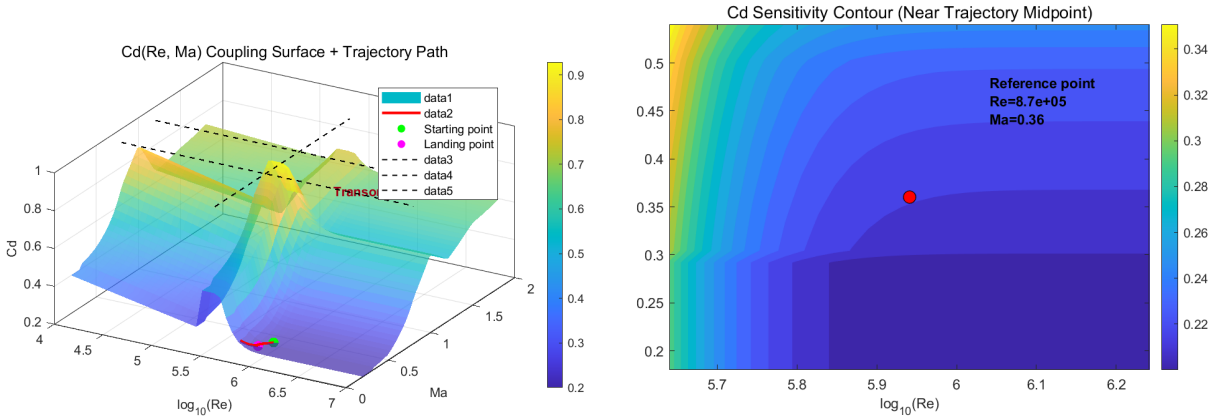
Through the aforementioned model, we are able to capture the Mach-effect-dominated high-speed regime. Next, we will couple the two models, following the logic outlined below:

- **Weight w_{Re}:**
  - If $\mathrm{Ma} < 0.5$: $w_{\mathrm{Re}} = 1.0$ (Re-dominated).
  - If $0.5 \leq \mathrm{Ma} < 1.5$: $w_{\mathrm{Re}} = 1.0 - 0.8 \cdot \frac{\mathrm{Ma}-0.5}{1.0}$.
  - If $\mathrm{Ma} \geq 1.5$: $w_{\mathrm{Re}} = 0.2$ (Ma-dominated).
- **Fused C_d:**

$$C_{\text{d-Ma-only}} = 0.47 \cdot C_{\text{d-Ma-factor}} \quad \text{(high-Re baseline, no Re correction)} \tag{5.19}$$

$$C_d = w_{\mathrm{Re}} \cdot C_{\text{d-base}} \cdot C_{\text{d-Ma-factor}} + (1 - w_{\mathrm{Re}}) \cdot C_{\text{d-Ma-only}} \tag{5.20}$$

We implemented the coupling process of Re and Ma through programming, and the coupling results are shown in the figure below:



(a) (b)

Figure 3: Visualization of the Reynolds–Mach Coupling Results

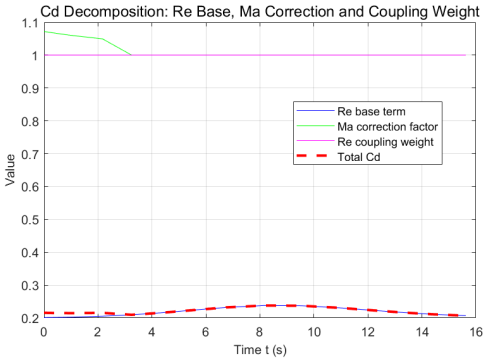


Figure 4: C_d-Ma-Re

From the results, the overall drag coefficient curve largely coincides with the Re curve. Moreover, when the C_d-Re relationship deviates, the Ma correction factor exhibits pronounced fluctuations. These observations collectively demonstrate that the Re-Ma coupling performs effectively.

5.2 Model Solving and Result Analysis

5.2.1 Formulation of the ODE System

The formulation of the ODE system constitutes the core step in our ballistic simulation modeling, as it transforms the previously established physical model into its mathematical representation, thereby facilitating numerical computation. In the preceding model, we have defined the state vector

$$\mathbf{Y} = \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix},$$

which corresponds to a nonlinear system of equations and hence requires numerical methods for its solution. Based on the physical model developed earlier, the resulting ODE system can be expressed as:

$$\frac{d\mathbf{Y}}{dt} = \begin{bmatrix} v_x \\ v_y \\ -k \cdot v_{\text{rel}_x} \\ -g - k \cdot v_{\text{rel}_y} \end{bmatrix}, \tag{5.21}$$

where the drag factor is

$$k = \frac{1}{2} C_d \rho A v_{\text{rel}} / m. \tag{5.22}$$

For the low-speed threshold, i.e.,

$$v_{\text{rel}} < 10^{-6},$$

the equations reduce to

$$\frac{dv_x}{dt} = 0, \quad \frac{dv_y}{dt} = -g. \tag{5.23}$$

Following the completion of the preceding steps, the integration of the dynamic C_d calculation can now be carried out, as detailed in the procedure below.

1. **Re-based C_d (Piecewise):** Use Stokes' formula for low Re, and the high-Re plateau value for high Re.

2. **Ma Correction:** Apply Prandtl-Glauert rule for subsonic, linear interpolation for transonic, and quadratic growth for supersonic.

3. **Coupling Weight $w_{\mathbf{Re}}$:**

$$w_{\mathrm{Re}} = 1.0 - 0.8 \cdot \frac{Ma - 0.5}{1.0} \tag{5.24}$$

(for $Ma < 1.5$), where the influence of Re weakens at high Ma.

4. **Final C_d:**

$$C_d = w_{\mathrm{Re}} \cdot C_{d,\mathrm{base}} \cdot C_{d,\mathrm{Ma\ factor}} + (1 - w_{\mathrm{Re}}) \cdot C_{d,\mathrm{Ma\ only}} \tag{5.25}$$

where $C_{d,\mathrm{Ma\ only}} = 0.47 \cdot C_{d,\mathrm{Ma\ factor}}$. Apply the physical constraint $C_d \in [0.08, 1.2]$.

5.2.2 Numerical Integration Method

The numerical integration method is employed to solve the aforementioned initial value problem for the ODEs, integrating from the initial state to the point of impact where y=0. In this study, we adopt the ODE45 solver, as the problem under consideration is non-stiff. This solver is based on an embedded Runge–Kutta (RK) algorithm, which offers a favorable balance between high accuracy and computational efficiency. A simplified flowchart illustrating the solution procedure is presented below, while many specific details will be discussed in greater depth in the subsequent section.

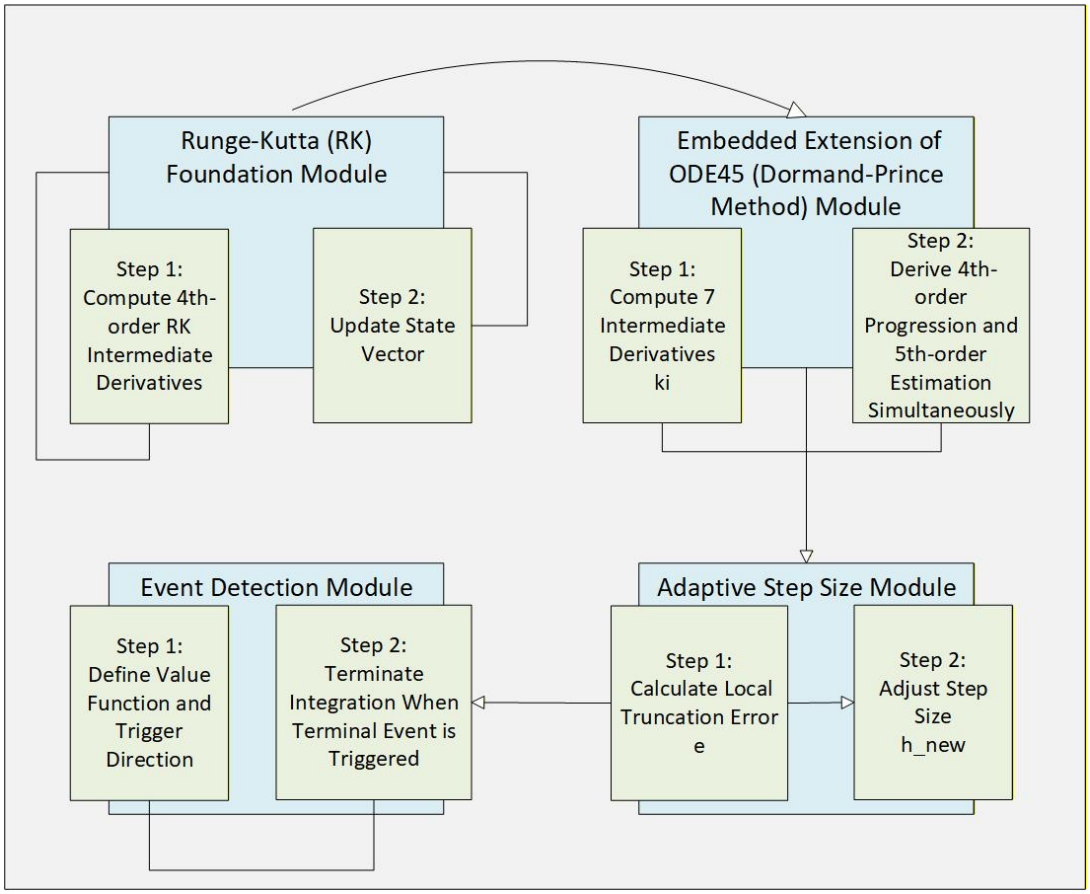


Figure 5: Solution Procedure

5.2.3 Solution Results

Based on the entirety of the model established, we implemented a computational simulation, yielding the results presented below:

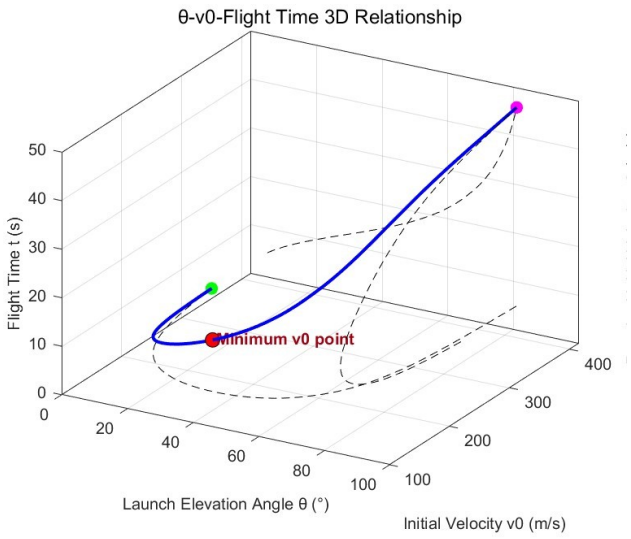


Figure 6: θ-C_d-v_0 relationship

The figure on the left shows the relationship between launch angle, initial velocity, and flight time, forming a distinct correlation curve. The blue curve highlights a spatial U-shaped trend: as the launch angle increases, the initial velocity first decreases and then rises, while the flight time steadily increases, with a marked extension at high angles. This pattern characterizes high ballistic trajectories, which feature prolonged flight durations coupled with relatively high initial velocities.

Figure 7 illustrates the dynamic evolution of the projectile drag coefficient (C_d) as a function of horizontal distance for different ballistic trajectories. It can be observed that, for the trajectory with the minimum initial velocity, the C_d curve is relatively smooth and stable, maintaining a value around 0.2. This indicates that under this trajectory the projectile velocity and ambient conditions tend toward stability, with the flow state approximating laminar or steady flow. For the low-elevation trajectory, C_d exhibits a monotonically decreasing trend, gradually declining from a higher value (approximately 0.57) to around 0.27 near the terminal stage.

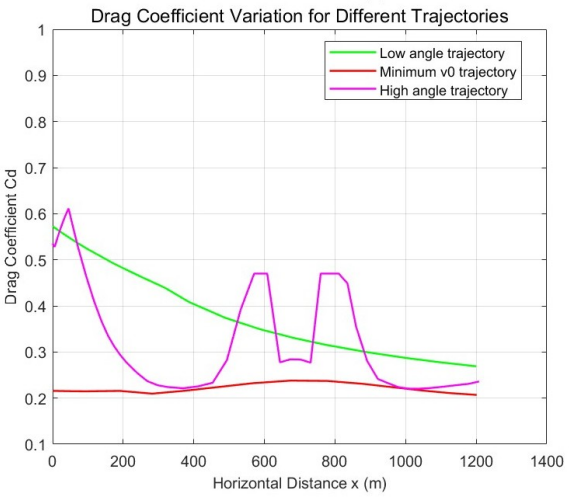


Figure 7: θ-C_d-v_0 relationship

This reflects the influence of decreasing ambient density and velocity changes with increasing flight distance, leading to a gradual reduction in C_d. In contrast, the high-elevation trajectory shows pronounced fluctuations, with multiple peaks occurring near 600m and 800m, and C_d oscillating sharply between 0.2 and 0.6. This suggests that the projectile may traverse different aerodynamic regimes, such as passing through the transonic critical zone, thereby inducing shock waves and flow separation, which are inher-

ently unsteady phenomena. These observations are consistent with the model established earlier.

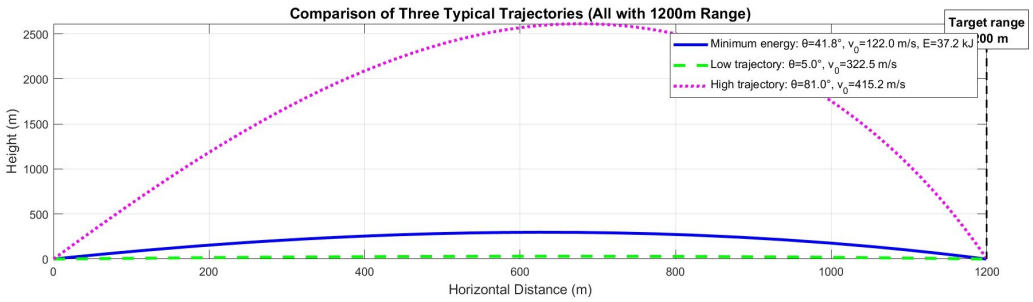


Figure 8: Comparison of Three Typical Trajectories

Figures 8 and 9 present three representative solutions: the lowest trajectory, the highest trajectory, and the trajectory with the minimum initial velocity.

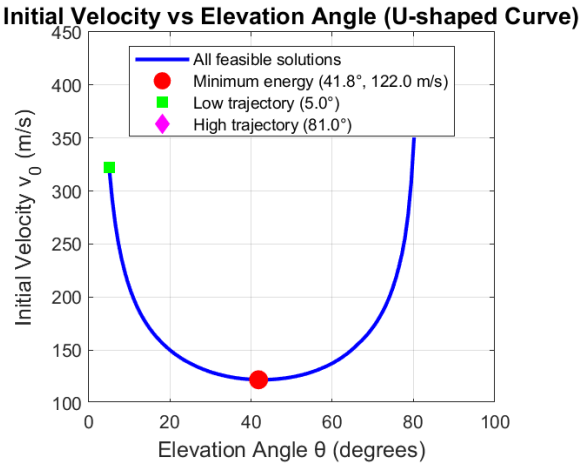


Figure 9: θ-v_0 relationship

In the context of 19th-century applications, the minimum-energy solution corresponds to the optimal launch angle that achieves the target range with a lower initial velocity, embodying the principle of energy minimization and meeting battlefield demands for conserving propellant and maximizing resource efficiency. The lowest-trajectory solution represents a rapid, low-elevation path with short flight time, suitable for swift strikes but requiring relatively high initial velocity. In contrast, the highest-trajectory solution follows a high-elevation parabolic path, characterized by longer flight time and greater altitude, enabling strikes over obstacles against well-concealed targets, albeit with higher energy requirements.

These three representative solutions effectively meet the operational requirements of 19th-century artillery, highlighting the complementary advantages of different trajectories in terms of energy efficiency, range, and tactical applicability.

6 The impact of windy conditions and altitude changes

This part investigates how to determine projectile launch parameters (initial velocity v_0, elevation angle θ, azimuth angle φ) to accurately hit a specified target point in three-dimensional space under wind field conditions.

6.1 Environmental Model

6.1.1 Gravitational Acceleration with Altitude and Latitude Variation

According to the international formula for gravity:

$$g(z,\lambda) = g_0(\lambda)\left(1 - \frac{2z}{R_{\text{earth}}}\right) \tag{6.1}$$

where:

$$g_0(\lambda) = 9.780327\left(1 + 0.0053024\sin^2\lambda - 0.0000058\sin^2 2\lambda\right)\ \text{m/s}^2$$
$$R_{\text{earth}} = 6371000\,\text{m}$$

λ is the latitude of the launch point,z represents the altitude.We can obtain the following relationship diagram:

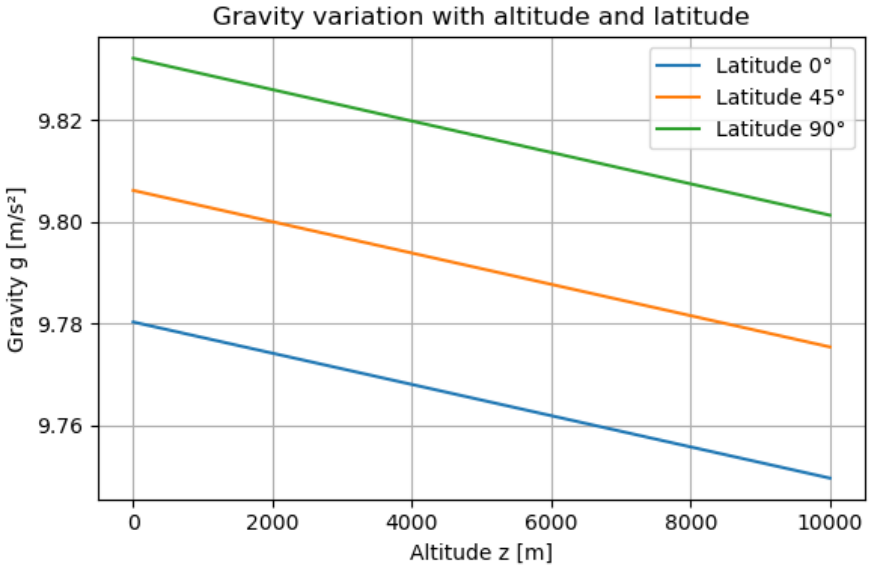


Figure 10: gravity variation

The gravitational acceleration varies at different latitudes, which in turn affects the trajectory of the shell

6.1.2 International Standard Atmosphere Model

Atmospheric density $\rho(z)$ adopts the tropospheric model:

$$T(z) = T_0 - L \cdot z \tag{6.2}$$

$$P(z) = P_0\left(1 - \frac{L \cdot z}{T_0}\right)^{\frac{g_0 M}{RL}} \tag{6.3}$$

$$\rho(z) = \frac{P(z)}{R_{\text{gas}}T(z)} \tag{6.4}$$

The sea level temperature is $T_0 = 288.15\,\text{K}$; the temperature lapse rate is $L = 0.0065\,\text{K/m}$; the sea level pressure is $P_0 = 101325\,\text{Pa}$; the molar mass of air is $M = 0.02896\,\text{kg/mol}$; the universal gas constant is $R = 8.314\,\text{J/(mol}\cdot\text{K)}$; and the specific gas constant for dry air is $R_{\text{gas}} = 287.05\,\text{J/(kg}\cdot\text{K)}$. Derived from the ideal gas law ($\rho = p/(R_a T)$) and combined with the atmospheric pressure formula, we have:

$$\rho = \rho_0 \times (1 - h/H_0)^{(g_0 M/(R^* T_0)-1)} \tag{6.5}$$

Parameter Explanations:

1. $\rho_0 = 1.205\ \text{kg/m}^3$ (standard air density at sea level);
2. Other parameters are consistent with the atmospheric pressure formula (T is taken as 288.15 K);

The change curve is shown in the following figure:

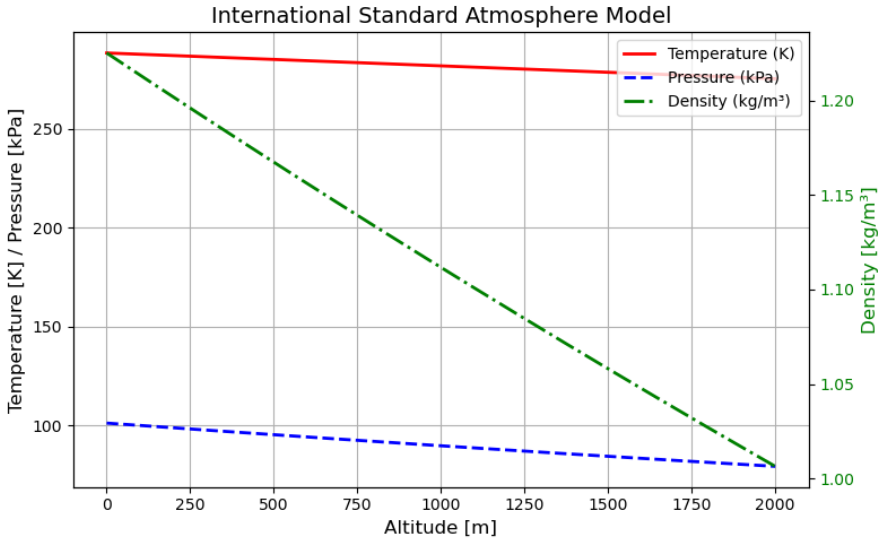


Figure 11: ISAmodel variation

It can be seen that within an altitude of 2,000 meters, the changes in temperature and air pressure are not obvious. Therefore, when calculating the air density, the variations of the two can be ignored and that is reasonable.

6.1.3 Air Dynamic Viscosity (μ) Variation with Altitude

μ is mainly affected by temperature. As altitude increases, the temperature decreases slightly, but the overall influence is moderate. The formula is as follows:

$$\mu = \mu_0 \times (T_0 + C)/(T(h) + C) \times (T(h)/T_0)^{(3/2)} \tag{6.6}$$

$\mu_0 = 1.81 \times 10^{-5}$ Pa · s denotes the dynamic viscosity at sea level at 20°C; $T_0 = 288.15$ K is the standard temperature at sea level; $T(h) = 288.15 - 0.0065h$ represents the approximate temperature formula at altitude h, applicable for the range of 0 10000m with a temperature drop of 6.5K per 1000m increase in altitude; and $C = 110.4$ K is the Sutherland constant, a fixed value.The change curve is shown in the following figure

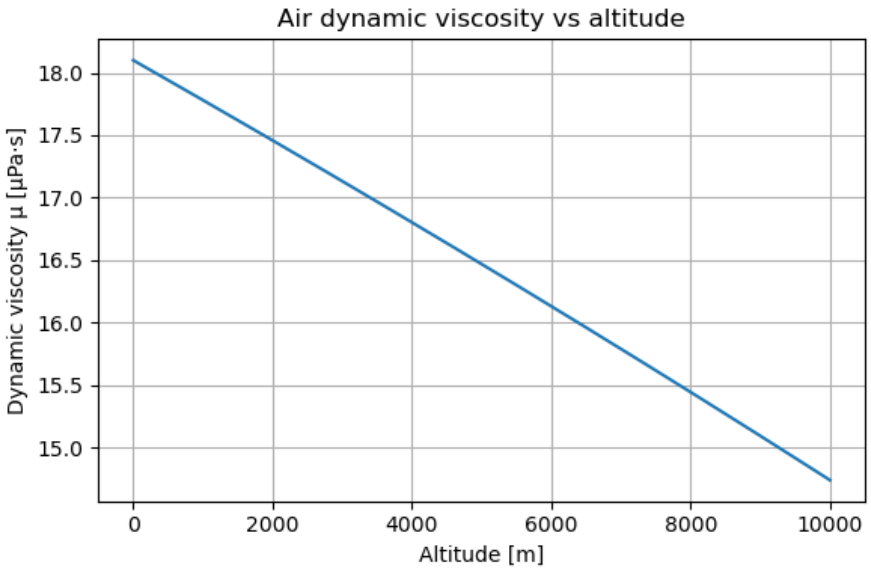


Figure 12: viscosity variation.

6.2 Physical Model

During the flight of a spherical projectile, in addition to gravity, it also interacts with the air. The magnitude and direction of gravity are constant, denoted as mg, and directed vertically downward. What needs to be discussed is the interaction between the sphere and the air. This involves aerodynamic problems and is relatively complex, requiring modeling simplification.

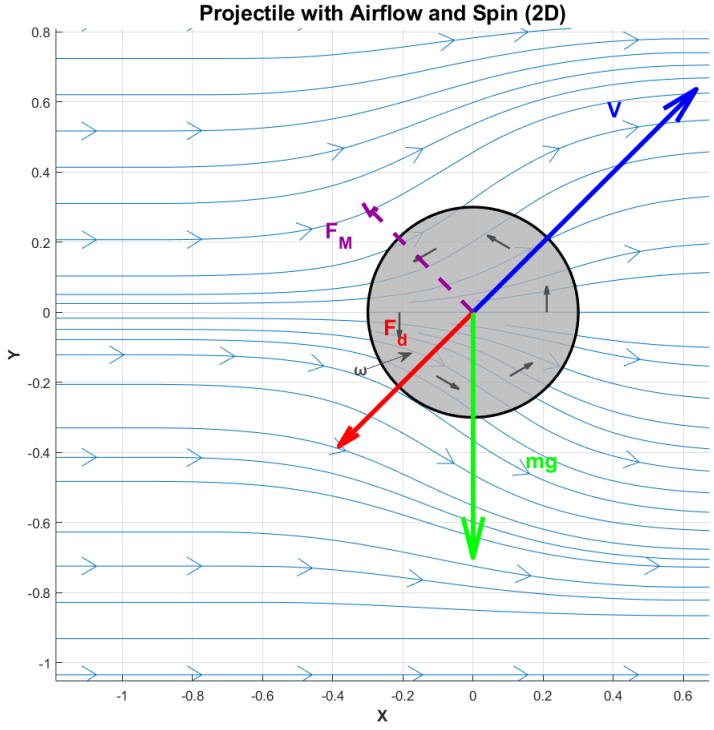


Figure 13: Projectile with Airflow and Spin

The interaction between the sphere and the air can be simplified into three forces: the buoyancy force (The buoyancy force is very small compared to the gravitational force of the sphere, so it is not considered.), the air drag force $\mathbf{F}_{\mathrm{D}}$, and the Magnus force $\mathbf{F}_{\mathrm{M}}$ generated by rotation. The force situation is shown above. (1)

6.2.1 Air Resistance

The direction of air drag $\mathbf{F}_{\mathrm{D}}$ is always opposite to the velocity direction, and the magnitude of $\mathbf{F}_{\mathrm{D}}$ is related to velocity, as shown below

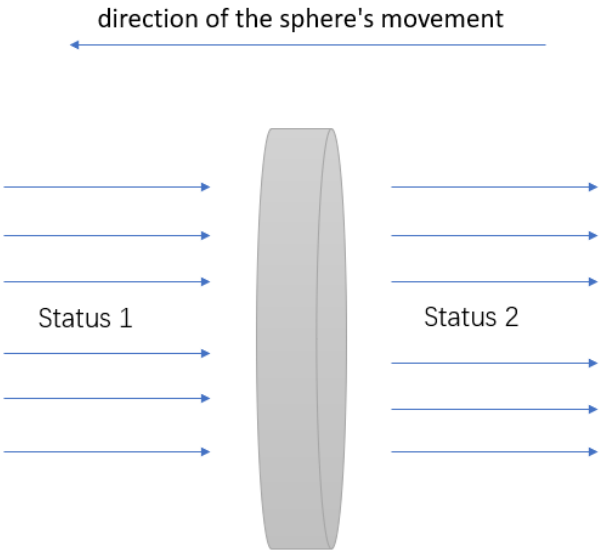


Figure 14: Diagram of the airflow on both sides of the disc

Assume the sphere moves in air with velocity v. Take a differential area element in front of the sphere, denoted as 1, with pressure p_1 and velocity $v_1 = v$. Take a differential area element behind the sphere, denoted as 2, with pressure p_2 and velocity $v_2 = 0$. According to Bernoulli's equation:

$$p_1 + \frac{1}{2}\rho v_1^2 = p_2 + \frac{1}{2}\rho v_2^2 \tag{6.7}$$

From equation (6.7), we obtain:

$$p_1 - p_2 = \frac{1}{2}\rho v^2 \tag{6.8}$$

Let S be the cross-sectional area of the sphere, then the air drag force is:

$$F_{\mathrm{D}} = (p_1 - p_2)S = \frac{1}{2}\rho S v^2 \tag{6.9}$$

Equation (6.9) is only a rough estimate. In reality, the airflow distribution around the sphere is complex. The airflow separates at the sphere surface and forms a wake, and these factors all affect the magnitude of drag. To describe air resistance more accurately, a drag coefficient C_{d} is introduced. (1)

In this problem, the projectile is modeled as a rigid sphere, and due to its relatively high velocity, the entire motion process can be regarded as a turbulent flow process. The drag coefficient satisfies the empirical formula Thus, the final calculation formula can be derived as follows:

$$C_d = \frac{24\mu}{\rho dv} + \frac{6}{1+\sqrt{\frac{\rho dv}{\mu}}} + 0.4 \tag{6.10}$$

Therefore, the air drag force can be expressed as:

$$\mathbf{F}_{\mathrm{D}} = -\frac{1}{2} C_{\mathrm{d}} \rho S v^2 \hat{\mathbf{v}} \tag{6.11}$$

where S is the cross-sectional area of the sphere; ρ is the air density; $\hat{\mathbf{v}}$ is the unit vector in the velocity direction. It should be noted that the drag coefficient is related to the surface roughness of the sphere; the rougher the surface, the smaller the drag coefficient.

6.2.2 Magnus Force

When the sphere rotates, it generates Magnus force. The mechanism of its generation is shown in Figure 3.

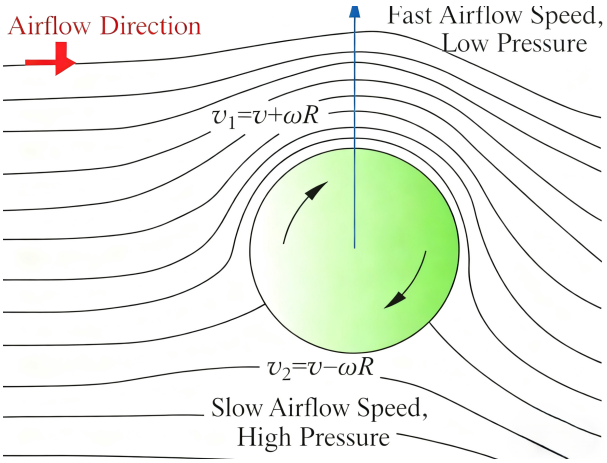


Figure 15: Mechanism of Magnus force generation

As shown in Figure 3, the sphere moves to the right with velocity v while rotating counterclockwise about an axis perpendicular to the paper with angular velocity ω. According to Bernoulli's equation:

$$p_1 + \frac{1}{2}\rho v_1^2 = p_2 + \frac{1}{2}\rho v_2^2 = c \tag{6.12}$$

where:

$$v_1 = v + \omega R, \quad v_2 = v - \omega R \tag{6.13}$$

ω is the angular velocity, and R is the radius of the sphere.
Therefore:

$$F_{\mathrm{M}} = S(p_2 - p_1) = \frac{1}{2}\pi R^2 \rho (v_1^2 - v_2^2) = 2\pi \rho R^3 v \omega \tag{6.14}$$

Similar to the drag coefficient C_d in equation (6.11), introducing a lift coefficient C_l, and noting that $\mathbf{F}_\mathrm{M} \sim \boldsymbol{\omega} \times \hat{\mathbf{v}}$, the Magnus force can be expressed as:

$$\mathbf{F}_\mathrm{M} = \frac{1}{2} C_\mathrm{l} \rho S v^2 \frac{\boldsymbol{\omega} \times \hat{\mathbf{v}}}{|\boldsymbol{\omega} \times \hat{\mathbf{v}}|} \tag{6.15}$$

In equation (6.15), C_l is related to the angular velocity, surface roughness, velocity, and Reynolds number.

In this model, after estimation, due to the high velocity of the projectile (up to 450 m/s at maximum), the cumulative effect of the Magnus force is not significant. The aerodynamic drag dominates, while the Magnus force relatively small. Therefore, the Magnus force is neglected for the sake of simplifying calculations.

6.2.3 The Coriolis Force

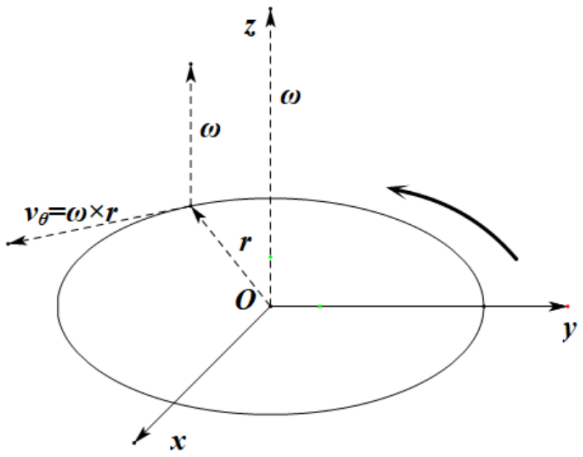


Figure 16: Coriolis Force

The Coriolis force is an apparent inertial force experienced in a rotating reference frame, such as the Earth. It arises due to the rotation of the Earth and acts on moving objects relative to the Earth's surface. In vector form, it is expressed as:

$$\mathbf{F}_\mathrm{c} = -2m\,(\boldsymbol{\omega} \times \mathbf{v}) \tag{6.16}$$

where m is the mass of the object, $\mathbf{v}$ is its velocity relative to the rotating frame, and $\boldsymbol{\omega}$ is the Earth's angular velocity vector ($\|\boldsymbol{\omega}\| \approx 7.292115 \times 10^{-5}$ rad/s).

Physically, the Coriolis force deflects the trajectory of the moving body to the right in the Northern Hemisphere and to the left in the Southern Hemisphere. Its magnitude depends on both the velocity of the object and the sine of the latitude. In the present study, the Coriolis force is neglected because the projectile's flight time is only a few seconds and the range is relatively short, making the resulting deflection due to Earth's rotation negligible compared with dominant aerodynamic effects such as drag variation with altitude.

6.2.4 Equations of Motion

We adopted the East-North-Up (ENU) coordinate system, where the x-axis points East, the y-axis points North, and the z-axis points Up (vertically upward).

The launch point coordinates are (x_0, y_0, z_0), and the target point coordinates are (x_t, y_t, z_t).

The projectile motion in wind field $\vec{v}_w = (v_{wx}, v_{wy}, v_{wz})$ is described by the following system of ordinary differential equations:

$$\frac{dx}{dt} = v_x, \quad \frac{dy}{dt} = v_y, \quad \frac{dz}{dt} = v_z \tag{6.17}$$

$$\frac{dv_x}{dt} = -k(z, v) \cdot v_{\text{rel}} \cdot (v_x - v_{wx}) \tag{6.18}$$

$$\frac{dv_y}{dt} = -k(z, v) \cdot v_{\text{rel}} \cdot (v_y - v_{wy}) \tag{6.19}$$

$$\frac{dv_z}{dt} = -g(z, \lambda) - k(z, v) \cdot v_{\text{rel}} \cdot (v_z - v_{wz}) \tag{6.20}$$

where:

- Relative wind speed: $v_{\text{rel}} = \sqrt{(v_x - v_{wx})^2 + (v_y - v_{wy})^2 + (v_z - v_{wz})^2}$
- Drag coefficient: $k(z, v) = \frac{\rho(z) C_d(M) A}{2m}$

Given launch parameters (v_0, θ, φ), the initial velocity components are:

$$v_{x0} = v_0 \cos\theta \cos\varphi \tag{6.21}$$

$$v_{y0} = v_0 \cos\theta \sin\varphi \tag{6.22}$$

$$v_{z0} = v_0 \sin\theta \tag{6.23}$$

Initial state vector:

$$\vec{s}_0 = (x_0, y_0, z_0, v_{x0}, v_{y0}, v_{z0}) \tag{6.24}$$

6.3 Numerical Solution Method

6.3.1 Numerical Integration Method: Runge-Kutta Method

The Runge-Kutta (RK) family of methods are explicit numerical schemes for solving ordinary differential equations (ODEs) of the form:

$$\frac{d\vec{y}}{dt} = \vec{f}(t, \vec{y}), \quad \vec{y}(t_0) = \vec{y}_0 \tag{6.25}$$

The classical fourth-order Runge-Kutta method (RK4) approximates the solution by evaluating the derivative at multiple intermediate points within each time step:

$$\vec{k}_1 = \vec{f}(t_n, \vec{y}_n) \tag{6.26}$$

$$\vec{k}_2 = \vec{f}(t_n + \frac{h}{2}, \vec{y}_n + \frac{h}{2}\vec{k}_1) \tag{6.27}$$

$$\vec{k}_3 = \vec{f}(t_n + \frac{h}{2}, \vec{y}_n + \frac{h}{2}\vec{k}_2) \tag{6.28}$$

$$\vec{k}_4 = \vec{f}(t_n + h, \vec{y}_n + h\vec{k}_3) \tag{6.29}$$

$$\vec{y}_{n+1} = \vec{y}_n + \frac{h}{6}(\vec{k}_1 + 2\vec{k}_2 + 2\vec{k}_3 + \vec{k}_4) \tag{6.30}$$

where h is the time step size. The local truncation error is $\mathcal{O}(h^5)$, making RK4 highly accurate for smooth problems.The method is as shown in the following figure:

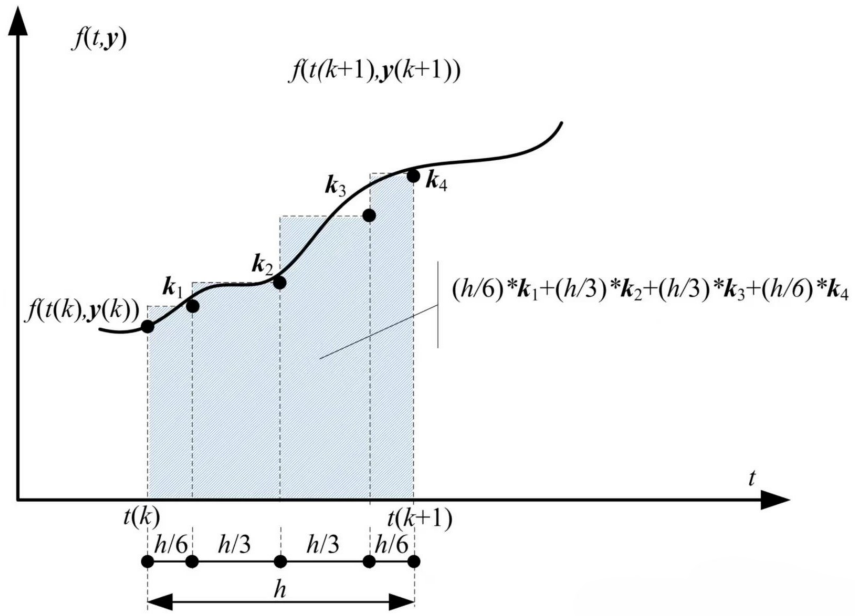


Figure 17: fourth-order Runge-Kutta method

The Runge-Kutta-Fehlberg method (RK45) extends RK4 by computing two estimates of different orders:

- A fifth-order accurate solution: $\vec{y}_{n+1}^{(5)}$ with error $\mathcal{O}(h^6)$
- A fourth-order accurate solution: $\vec{y}_{n+1}^{(4)}$ with error $\mathcal{O}(h^5)$

The local error estimate is obtained by comparing these two solutions:

$$\vec{e}_n = \vec{y}_{n+1}^{(5)} - \vec{y}_{n+1}^{(4)} \tag{6.31}$$

The step size h is dynamically adjusted to maintain the error below user-specified tolerances:

$$h_{\text{new}} = h_{\text{old}} \cdot \min\left(f_{\max}, \max\left(f_{\min}, s\left(\frac{\epsilon}{||\vec{e}_n||}\right)^{1/5}\right)\right) \tag{6.32}$$

where ϵ is the desired tolerance, $s \approx 0.9$ is a safety factor, and $f_{\min}, f_{\max}$ are bounds on step size changes.(3)

we chosed RK45 method because it offers adaptive step-size control, automatically adjusting the time step to match the projectile's changing dynamics avoiding wasted computation from overly small fixed steps and accuracy loss from overly large ones. Its fifth-order local accuracy keeps cumulative errors negligible over long simulations (up to 500s), ensuring precise impact point prediction. RK45 also supports event detection for accurately terminating integration at the exact moment of ground impact.

6.3.2 Trajectory Simulation Implementation

The six-dimensional state vector is defined as:

$$\vec{s}(t) = \begin{bmatrix} x(t) & y(t) & z(t) & v_x(t) & v_y(t) & v_z(t) \end{bmatrix}^T \tag{6.33}$$

The governing ODE system, derived from Newton's second law with aerodynamic drag and gravitational acceleration, is:

$$\frac{d\vec{s}}{dt} = \begin{bmatrix} v_x \\ v_y \\ v_z \\ -k(z,v) \cdot v_{\text{rel}} \cdot (v_x - w_x) \\ -k(z,v) \cdot v_{\text{rel}} \cdot (v_y - w_y) \\ -g(z, \phi_{\text{lat}}) - k(z,v) \cdot v_{\text{rel}} \cdot (v_z - w_z) \end{bmatrix} \tag{6.34}$$

The RK45 integrator is configured with the following parameters to ensure both accuracy and computational efficiency:

Parameter	**Value**
Numerical integrator	`scipy.integrate.solve_ivp` (RK45 method)
Integration interval	$t \in [0, 500]$ s
Relative error tolerance	$\epsilon_{\text{rel}} = 10^{-8}$
Absolute error tolerance	$\epsilon_{\text{abs}} = 10^{-8}$
Maximum step size	$\Delta t_{\max} = 0.05$ s
Initial step size	$\Delta t_0 = 10^{-4}$ s (auto-adjusted)

Table 6.1: RK45 integration parameters

The strict 10^{-8} error tolerances keep position errors under 1mm over 500s, velocity errors under 10^{-5}m/s, and energy conservation errors below 0.01%.

To precisely determine the moment when the projectile reaches the target altitude z_t, an event function is defined:

$$f_{\text{hit}}(t, \vec{s}) = z(t) - z_t \tag{6.35}$$

The event is triggered when:

$$f_{\text{hit}}(t^*, \vec{s}(t^*)) = 0 \quad \text{and} \quad \left.\frac{dz}{dt}\right|_{t=t^*} = 0 \tag{6.36}$$

Upon event detection, the integrator refines t^* to machine precision via root-finding (bisection method), terminates integration immediately, and records the impact state $(x_{\text{imp}}, y_{\text{imp}}, z_{\text{imp}}, v_{x,\text{imp}}, v_{y,\text{imp}}, v_{z,\text{imp}})$. For each set of launch parameters (v_0, θ, φ), the RK45 simulation outputs the adaptive-step trajectory data, impact point and velocity, flight time, and range. These results are then used to iteratively refine the launch parameters in the optimization loop.

6.3.3 Launch Parameter Optimization

Given a target location $\vec{r}_t = (x_t, y_t, z_t)$ and launch point $\vec{r}_0 = (x_0, y_0, z_0)$, the goal is to find the launch parameters (v_0, θ, φ) such that the projectile hits as close as possible to the target. This is formulated as a nonlinear least squares problem:

$$\min_{v_0,\theta,\varphi} \|\vec{r}(v_0, \theta, \varphi)\|_2^2 \tag{6.37}$$

where the residual vector includes horizontal impact coordinate errors:

$$\vec{r}(v_0, \theta, \varphi) = \begin{bmatrix} x_{\text{imp}}(v_0, \theta, \varphi) - x_t \\ y_{\text{imp}}(v_0, \theta, \varphi) - y_t \end{bmatrix} \tag{6.38}$$

Note the vertical error $z_{\text{imp}} - z_t$ is implicitly enforced by the event detection of projectile impact at the target elevation, so it is not explicitly minimized.

Optimization algorithm: Levenberg–Marquardt (L–M) method implemented via `scipy.optimize.least_squares`, with 3-point finite difference Jacobian approximation offers stable and fast convergence.

The iteration step is:

$$\vec{p}_{k+1} = \vec{p}_k - (\mathbf{J}_k^\top \mathbf{J}_k + \lambda_k \mathbf{I})^{-1} \mathbf{J}_k^\top \vec{r}(\vec{p}_k) \tag{6.39}$$

where $\vec{p} = [v_0, \theta, \varphi]^\top$ and λ_k is an adaptive damping parameter.

Parameter bounds:

$$\begin{cases} 50 \le v_0 \le v_{0,\max} \quad (\text{e.g., } 450\, m/s) \\ 5^\circ \le \theta \le 85^\circ \\ \varphi_{\text{init}} - 45^\circ \le \varphi \le \varphi_{\text{init}} + 45^\circ \end{cases}$$

to ensure physical feasibility and limit search space.

Initial guess strategy:

- Compute horizontal displacement and azimuth:

$$dx = x_t - x_0, \quad dy = y_t - y_0, \quad dz = z_t - z_0$$
$$d_{\text{horiz}} = \sqrt{dx^2 + dy^2}, \quad \varphi_{\text{init}} = \arctan 2(dy, dx)$$

- Evaluate local gravity g_{local} at launch altitude.
- For large vertical offset $dz > 50\, m$, set:

$$v_0^{\text{guess}} = 0.95\, v_{0,\max}, \quad \theta^{\text{guess}} = 45^\circ$$

- Otherwise estimate flight time and elevation angle:

$$v_0^{\text{guess}} = 0.85\, v_{0,\max}$$
$$t_{\text{guess}} = \frac{d_{\text{horiz}}}{v_0^{\text{guess}} \cos 30^\circ \times 0.6}$$
$$\theta^{\text{guess}} = \arctan \frac{dz + \frac{1}{2} g_{\text{local}} t_{\text{guess}}^2}{d_{\text{horiz}}}$$

clipped within $[10^\circ, 85^\circ]$.

6.4 model result

Among the three factors of horizontal distance, altitude and wind force, two are fixed and only one is changed to observe the influence on the trajectory of the shell, the launch elevation Angle, the azimuth Angle and the initial velocity:
Only change the wind speed:

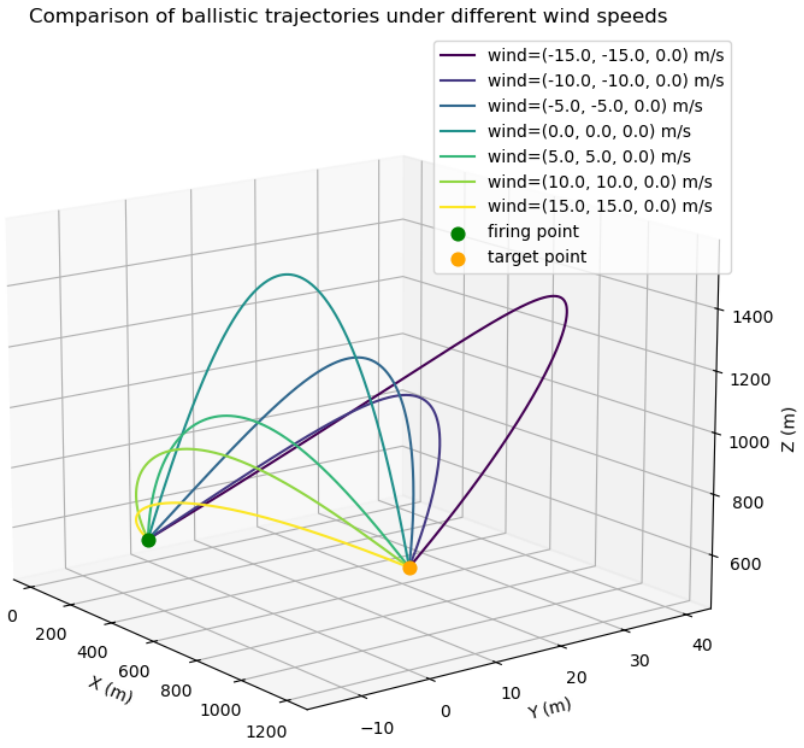


Figure 18: impact of different wind

The launch point we selected is [0, 0, 500], and the target point is [1200, 0, 750]. The wind speed adopts the wind combination in the X and Y directions (-15m/s,15m/s). From the trajectory, when the wind speed changes from (-15, -15,0) to (15,15,0), the trajectory of the shell tilts with the change of wind speed, which is very reasonable.When only the influence of wind force was considered, it was found after simulating the trajectory that selecting launch points and target points at different altitudes and distances had little impact on the results

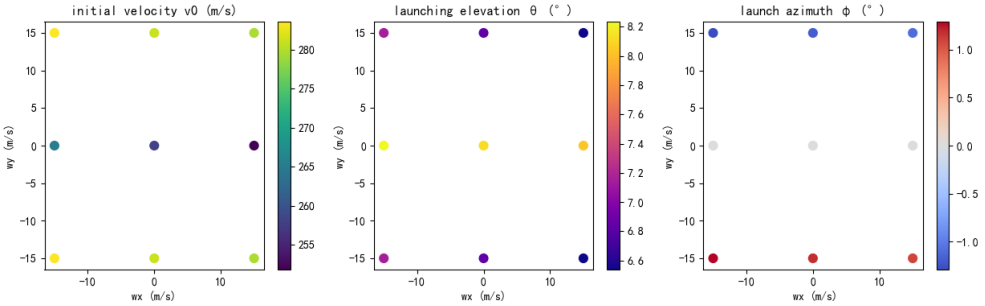


Figure 19: angle and initial velocity

From the perspective of launch Angle and initial velocity, the initial velocity is less than that with wind when there is no wind, and the azimuth is 0 when there is no wind. The pitch Angle is greater than that with wind. That is, when there is no wind, the launch can only be aimed at the target, while when there is wind, the launch Angle needs to be adjusted considering the influence of wind force. The greater the wind force, the greater the influence, which is also very reasonable

Only change the altitude:

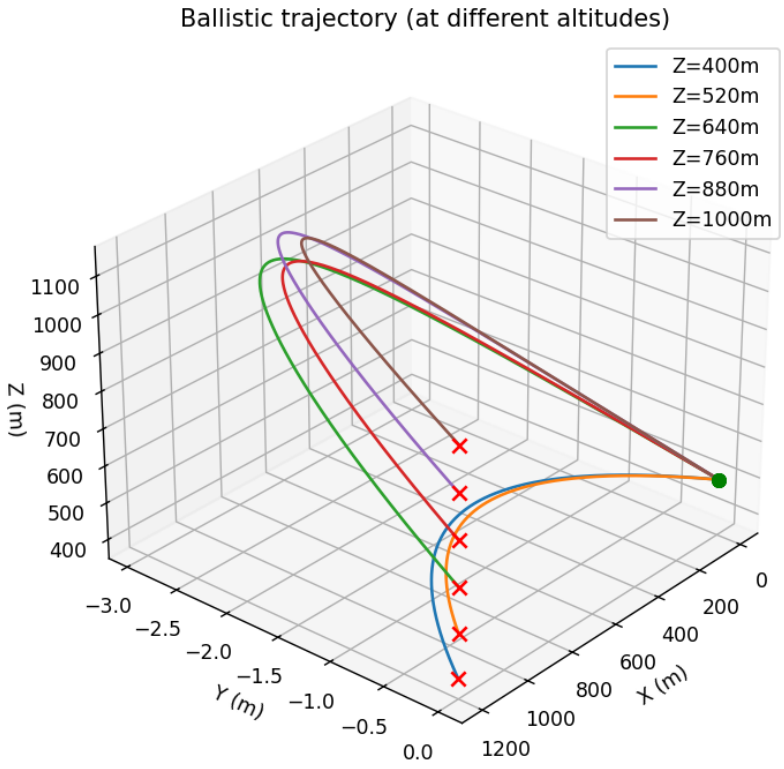


Figure 20: trajectory when the altitude is changed

The launch points we have chosen are [0,0,500], the X and Y coordinates of the target points are [1200,0], and the altitudes are selected from six points ranging from 400 meters to 1000 meters. The wind speed is [0,5,0]. From the perspective of the trajectory, all the trajectories shift along the direction of the wind, which is reasonable. The trajectory of

the target point at a higher altitude is significantly higher than that at a lower altitude. Through simulation, it is known that the given wind force at this time has little impact on the result

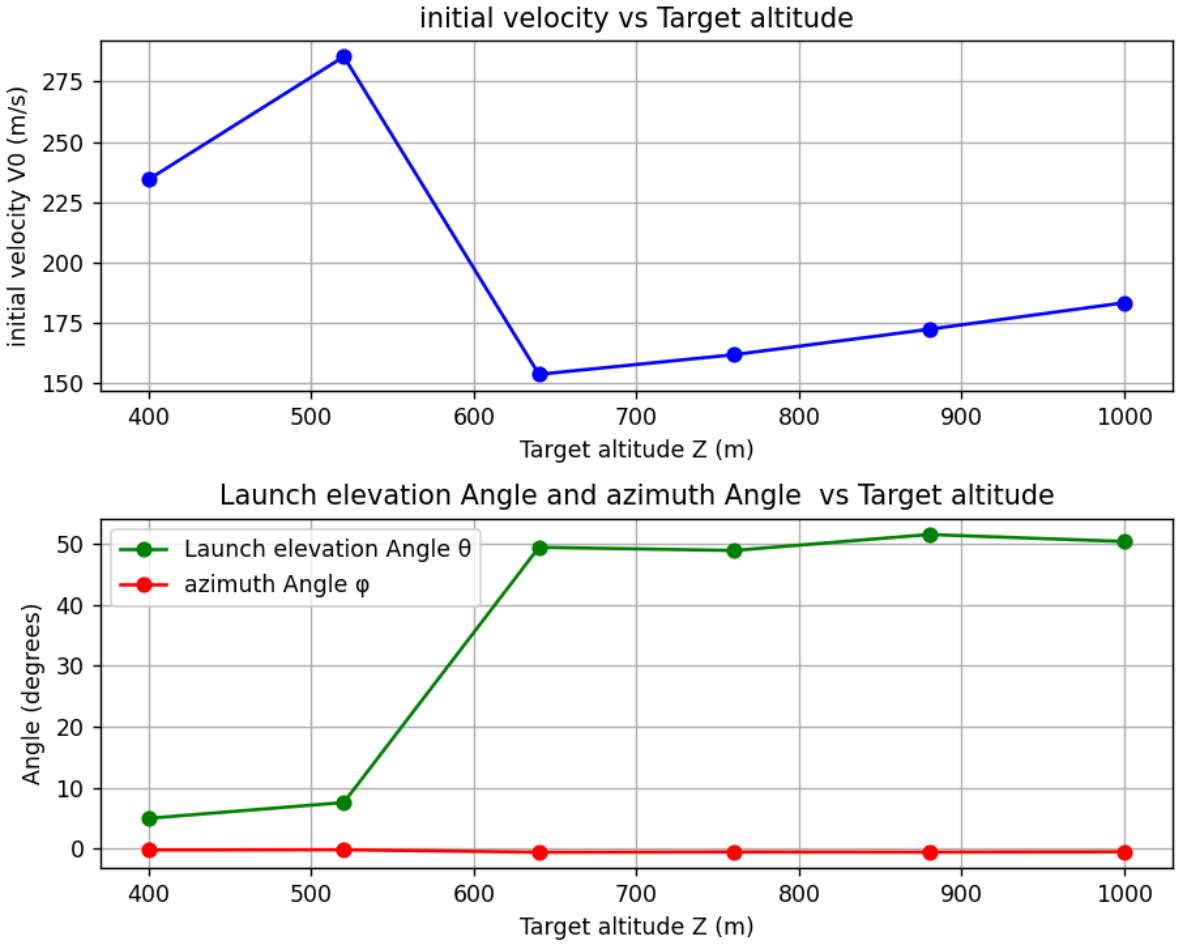


Figure 21: Altitude - Angle and initial velocity

From a perspective, due to the influence of wind force, the azimuth angles of each trajectory do not differ much. However, it can be seen that when the altitude is below 500 meters, the pitch angle changes very little. When the altitude exceeds 500 meters to a certain extent (that is, the altitude of the launch point), the launch pitch angle suddenly increases and then remains stable. This indicates that when the target point is higher than the launch point by a certain degree, the pitch angle should increase rapidly. In terms of initial velocity, when the height of the target point is the same as that of the launch point, its initial velocity is the greatest, which is the simplest case. When the target is higher than the launch point, the initial velocity drops sharply and then rises, which corresponds to the change in pitch angle and is also very reasonable. This indicates that when the height of the target increases, the initial velocity should be appropriately reduced before rising again.

only the horizontal distance is changed:
When only the horizontal distance is changed, the height of the fixed target point does have an impact on the result.The altitude of the launch point we chose is 500 meters.

We selected three target points with altitudes of 300 meters, 500 meters, and 700 meters. The trajectory is shown in the following diagram.

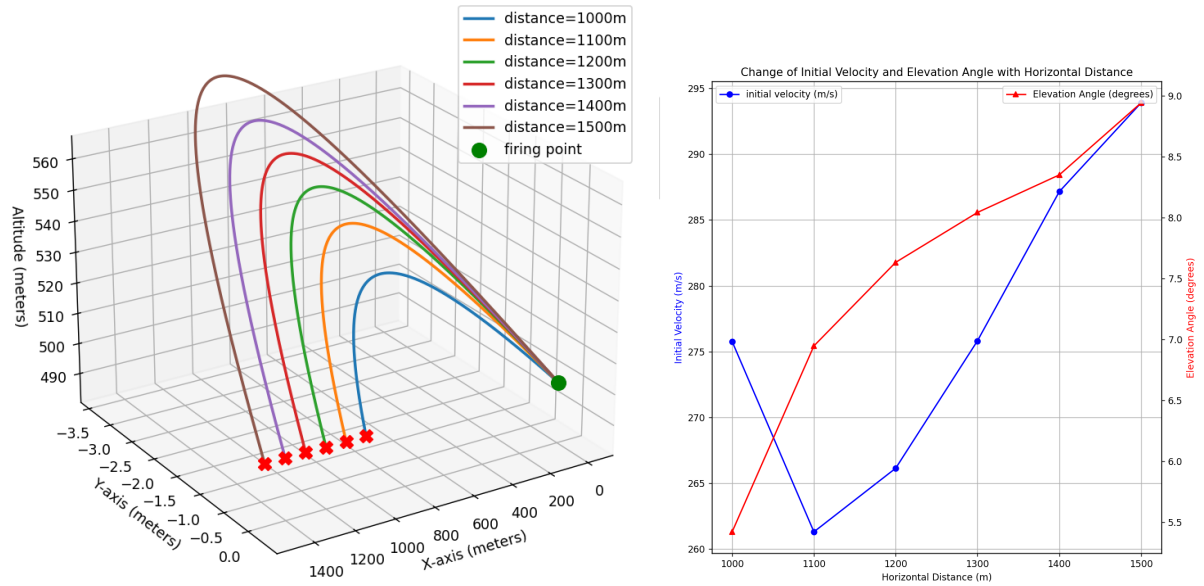


Figure 22: The target point is at an altitude of 500 meters.

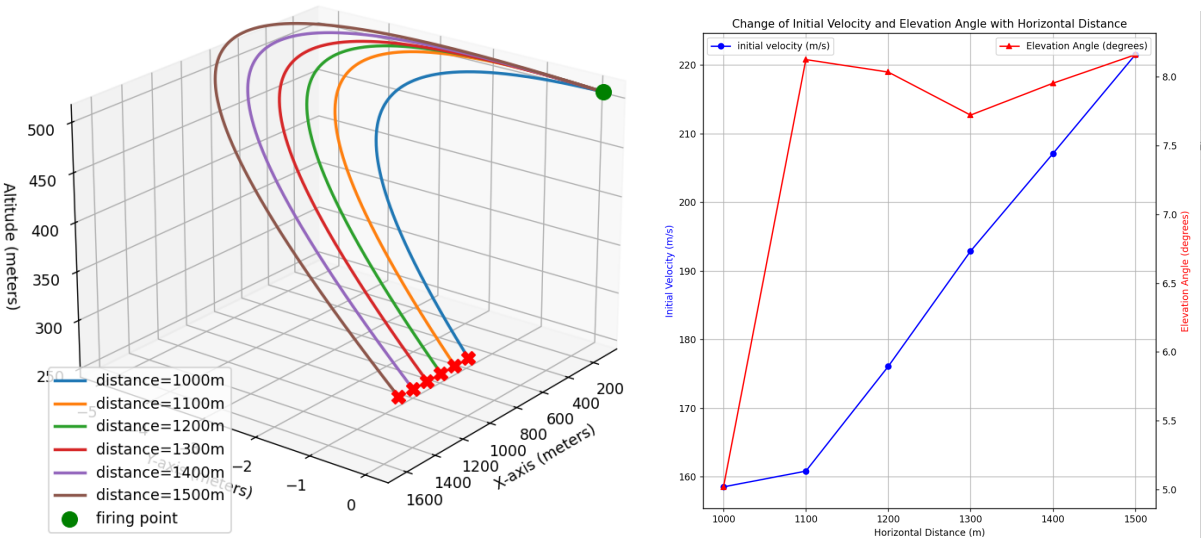


Figure 23: The target point is at an altitude of 300 meters.

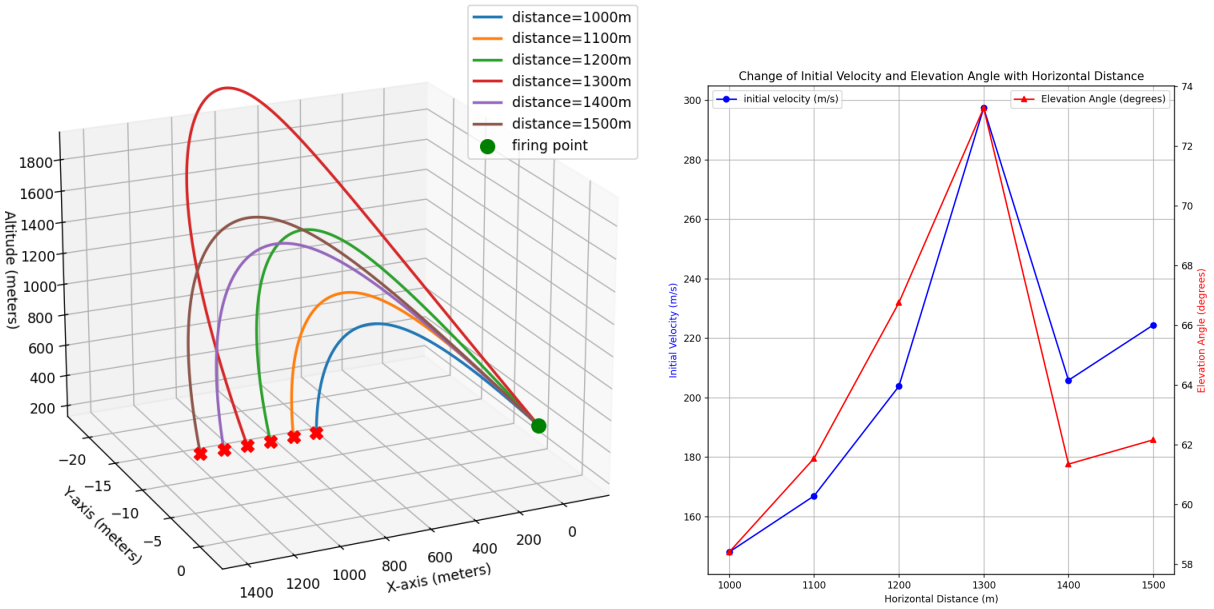


Figure 24: The target point is at an altitude of 700 meters.

we can tell from these graph that:

- **Horizontal Distance as a Primary Factor for Initial Velocity:** A monotonic increase in required initial velocity ($\boldsymbol{v_0}$) is consistently observed as the target's horizontal distance increases. Longer distances invariably necessitate higher muzzle velocities to successfully overcome gravity and air resistance, accurately delivering the projectile to the target point.
- **Target Altitude's Profound Influence on Velocity and Angle Trends:** The target's altitude, particularly its relationship to the launch point's altitude, dictates distinct trends in both initial velocity and elevation angle.
  - **Target Below Launch Point (Figure 23: Target altitude 300m vs. Launch point 500m):**
    - *Initial Velocity:* Relatively lower v_0 is required compared to other scenarios (approximately 220 m/s for a 1500m distance). The v_0 growth curve is generally smoother and more monotonic with increasing distance.
    - *Elevation Angle:* The elevation angle changes within a relatively smaller range, often showing a trend of slight initial decrease followed by a gradual increase with distance. Trajectories are typically flatter and more direct, effectively leveraging the inherent altitude advantage.
  - **Target at Same Altitude as Launch Point (Figure 22: Target altitude 500m vs. Launch point 500m):**
    - *Initial Velocity:* Moderate v_0 is required (approximately 290 m/s for a 1500m distance). The v_0 and elevation angle curves may exhibit minor fluctuations at shorter ranges (1000m to 1100m), potentially indicating the optimizer exploring different local optimal ballistic strategies (switching between high and low flight paths for efficiency).

  - ∗ *Elevation Angle:* Overall, the elevation angle tends to increase with distance, but its trend can be non-monotonic due to optimization nuances.
- **Target Above Launch Point (Figure 24: Target altitude 700m vs. Launch point 500m):**
  - ∗ *Initial Velocity:* This scenario demands the highest v_0 (e.g., peaking at approximately 290 m/s for a 1300m distance).
  - ∗ *Elevation Angle:* Both v_0 and elevation angle curves show the most significant and dramatic fluctuations, notably peaking at intermediate distances (e.g., 1300m) before potentially decreasing. To ascend to higher target altitudes, the **elevation angle must increase significantly** (e.g., approaching 70 degrees at 1300m), resulting in distinctly higher-arching and more "lofted" trajectories.

7 Rapid Estimation Method for Initial Velocity and firing Angle

Building upon the above context, we have derived the relationships between altitude, firing initial velocity, firing pitch angle, firing horizontal deflection angle, horizontal distance, and wind force. By substituting these relationships, we respectively obtain three methods for quickly estimating the launch initial velocity and firing angle from these relationships, as detailed below:

7.1 Rapid Lookup Table Method

Artillery officers can make preliminary decisions on the firing status of artillery pieces in actual combat by quickly collecting core parameters, locating basic entries in the ballistic table, and finally making environmental deviation corrections.

7.1.1 Basic Ballistic Table

The Basic Ballistic Table is a standardized technical document adapted to specific combinations of artillery models and projectile types. Its core function is to provide benchmark firing data, including the artillery elevation angle, horizontal distance, and initial artillery velocity, at fixed intervals within the conventional combat distance range, with the target's horizontal distance as the core index. It facilitates artillery officers to quickly retrieve basic entries in actual combat and provides accurate and efficient basic ballistic data support for the preliminary decision-making on the artillery firing status. Based on the factor relationship model in the aforementioned context, we have derived the Basic Ballistic Table as follows:

Base Firing Table (500-3000m, 51 points)

Range(m)	Elevation(°)	Muzzle Vel...
500	5.0	210
550	5.0	200
600	5.0	204
650	5.3	208
700	5.0	224
750	6.8	200
800	7.4	200
850	8.0	200
900	6.8	224
950	8.9	204
1000	10.0	200
1050	10.8	200
1100	11.6	200
1150	12.4	200
1200	13.2	200
1250	14.0	200
1300	15.0	200
1350	16.0	200
1400	16.8	200
1450	18.0	200
1500	19.3	200
1550	20.6	200
1600	21.8	200
1650	23.3	200
1700	25.1	200
1750	27.2	200

Range(m)	Elevation(°)	Muzzle Vel...
1800	29.8	200
1850	33.1	200
1900	33.9	204
1950	34.9	208
2000	32.0	216
2050	32.9	220
2100	34.1	224
2150	31.7	232
2200	32.8	236
2250	33.8	240
2300	34.2	245
2350	32.3	253
2400	33.3	257
2450	36.9	260
2500	33.9	268
2550	32.8	275
2600	34.5	279
2650	32.6	287
2700	35.9	290
2750	33.8	298
2800	33.1	305
2850	32.2	313
2900	34.0	317
2950	33.5	324
3000	32.6	332

Figure 25: Base Firing Table

This Basic Ballistic Table is a comprehensive result derived under windless conditions, the standard condition with an altitude of 500 meters, and the scenario where the artillery firing angle is directly aimed at the target. Since this artillery piece dates back 200 years ago, its maximum firing range did not exceed 3 km at that time; thus, the distance range of the basic table is from 500 m to 3000 m. Artillery officers can use this table to estimate the basic initial firing velocity and elevation angle, thereby providing a reliable basis for rapid estimation.

7.1.2 Correction Table

The Correction Table is a standardized technical document used in conjunction with the Basic Ballistic Table, whose core function is to address the limitations of the latter's fixed standard conditions. It provides precise parameter correction values for various variables that deviate from the standard state (windless conditions, 500m altitude, and the artillery directly aiming at the target) in actual combat, making the firing data more consistent with the actual battlefield environment and improving hit accuracy.

When more known conditions are available, artillery officers can further correct the basic initial fir ing velocity and elevation angle based on the local altitude, wind speed and direction, as well as the altitude difference between the firing position and the target. The Altitude Correction Table,Altitude Difference Table and three Wind Correction Tables are presented as follows:

Alt(m)	Elev Corr(°)	Vel Corr(m/s)
500	-1.2	+2.1
1000	-2.5	+5.4
1500	-3.8	+8.2
2000	-5.1	+11.6
2500	-6.4	+14.1
3000	-7.7	+17.3
3500	-9.0	+20.2
4000	-10.3	+23.3
4500	-11.6	+26.5
5000	-12.9	+29.0

Table 7.1: Altitude Correction Table

Tailwind(m/s)	Elev Corr(°)	Vel Corr(m/s)
2	-0.6	0
4	-1.3	0
6	-2.1	0
8	-2.6	0
10	-3.2	0

Table 7.2: Tailwind Correction Table

Altitude Difference(m)	Vel Corr(m/s)	Elev Corr(°)
-200	+5.7	-1.7
-150	+4.1	-1.2
-100	+2.8	-0.8
-50	+1.5	-0.4
0	0.0	0.0
+50	-1.3	+0.5
+100	-2.6	+0.9
+150	-3.9	+1.4
+200	-5.1	+1.8

Table 7.3: Altitude Difference Table

Headwind(m/s)	Elev Corr(°)	Vel Corr(m/s)
2	+0.8	0
4	+2.0	0
6	+3.1	0
8	+4.7	0
10	+6.9	0

Table 6.3: Headwind Correction Table

Crosswind(m/s)	Azimuth Corr(°)
2	-0.5
4	-1.0
6	-1.3
8	-1.5
10	-1.9

Table 6.4: Crosswind Correction

From this, we derive the quick steps for artillery officers to use the table lookup method as follows:

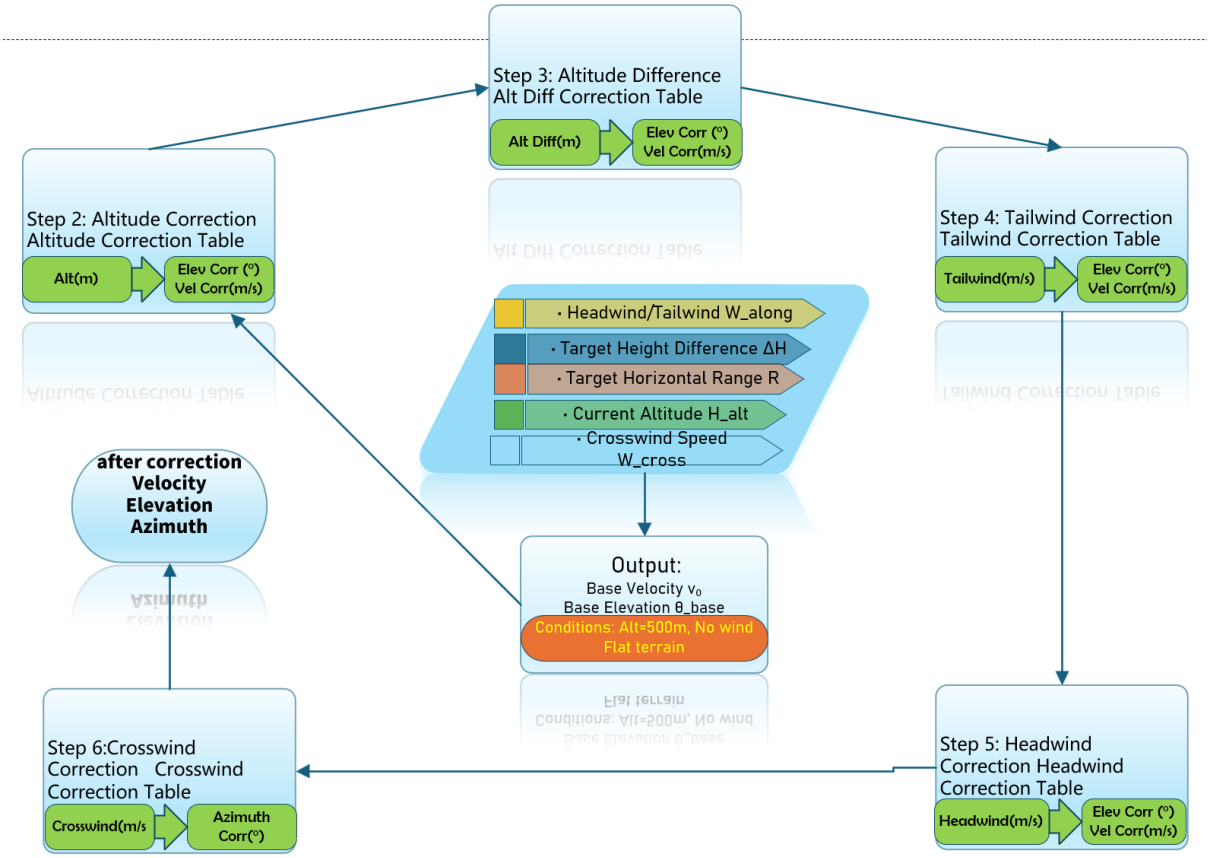


Figure 26: Flowchart for Table Lookup

This procedure enables mathematically trained artillery officers without access to computers or calculators to quickly estimate feasible initial velocities and firing elevation angles, providing preliminary support for rapid response in actual combat operations

7.2 Nomogram

A nomogram is a graphical computational tool that can convert complex multivariate computational problems into a simple line-drawing and reading operation. In the context of this artillery ballistics problem, gunners only need to use a straightedge to connect the known parameter points on the corresponding scales; the intersection point allows direct reading of the required firing parameters, eliminating the need for complex table lookup, interpolation, and cumulative correction calculations. This method is particularly suitable for rapid decision-making in battlefield environments, simplifying the original table lookup and correction process into a single line-drawing and reading step, which significantly improves the fire response speed while reducing the risk of human calculation errors. Designed based on the principles of logarithmic scales and geometric projection, a nomogram can achieve the two-dimensional visual representation of multi-dimensional parameters while maintaining sufficient accuracy.

Based on the basic firing table presented above, a preliminary nomogram relating distance, elevation angle, and initial velocity can be derived as follows:

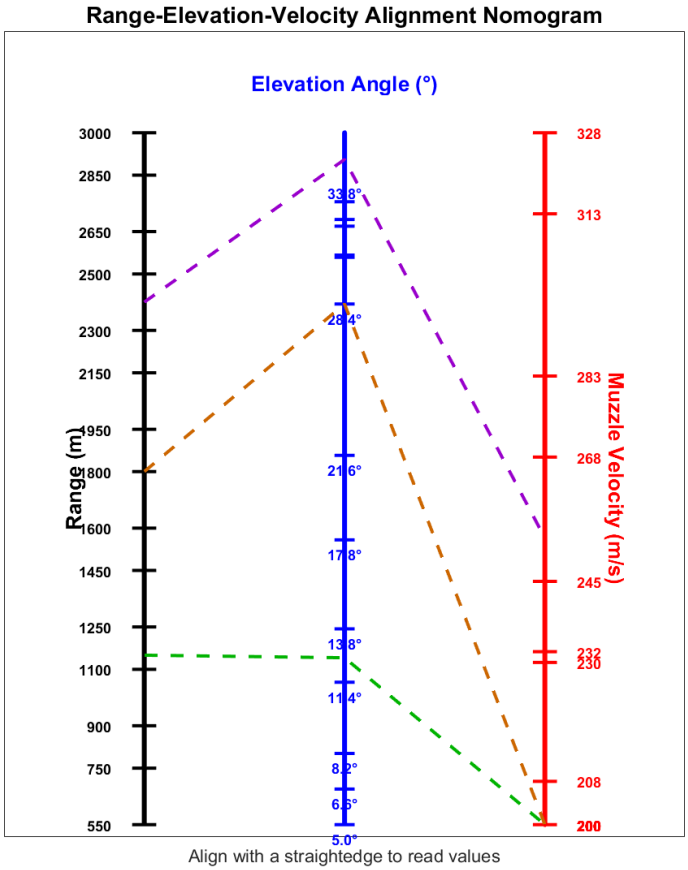


Figure 27: Range-Elevation-Velocity Alignment Nomogram

Artillery officers can quickly determine the required firing parameters by drawing lines with a straightedge and compass. If time permits, they can further make corrections to the initial velocity, elevation angle, and horizontal deflection angle obtained through paper-and-pencil drawing by utilizing the content in the correction table section presented above.

7.3 Rapid Formula Method

The rapid formula method provides closed-form approximations for calculating artillery firing parameters, simplified for mental calculation in field conditions.Since it is not feasible to directly derive a simplified formula via equations in the presence of drag or wind, to obtain a relatively accurate estimation formula, the following section neglects the effects of air drag and wind on the artillery.

7.3.1 Initial Velocity Calculation

Starting from vacuum baseline $v_0 = \sqrt{gR}$ with $g = 10$ m/s^2, we obtain $v_0 = \sqrt{10R}$. Progressive corrections yield:

Drag correction (range in km):

$$v_0 \approx \sqrt{10R} \times 1.2$$

General form with all corrections:

$$\boxed{v_0 \approx 3\sqrt{R} \times 1.2} \tag{7.1}$$

Mental calculation rule: For $R = 1000$ m, $v_0 \approx 3 \times 32 \times 1.2 \approx 115$ m/s. Scale linearly: double distance requires $\sqrt{2} \approx 1.4$ times velocity.

7.3.2 Elevation Angle Calculation

To address the rapid estimation of initial velocity and launch angle for artillery officers with mathematical training but without computational tools, we first establish the **vacuum trajectory model** as a theoretical basis. In an ideal vacuum condition where air resistance, altitude difference, and wind speed are neglected, the projectile undergoes projectile motion, whose motion equations can be decomposed into horizontal and vertical directions:

- In the **horizontal direction**, it performs uniform linear motion, with the velocity component $v_x = v_0 \cos\theta$ and the range $R = v_x t = v_0 \cos\theta \cdot t$.
- In the **vertical direction**, it undergoes uniformly variable linear motion, with the velocity component $v_z = v_0 \sin\theta - gt$. When the projectile lands, the vertical displacement is zero, so the flight time satisfies $0 = v_0 \sin\theta \cdot t - \frac{1}{2}gt^2$.

From the vertical motion equation, we solve for the **flight time**:

$$t = \frac{2v_0 \sin\theta}{g}$$

Substituting this flight time into the horizontal range expression, we **derive the range formula**:

$$R = v_0 \cos\theta \cdot \frac{2v_0 \sin\theta}{g} = \frac{v_0^2 \cdot 2\sin\theta\cos\theta}{g} = \frac{v_0^2 \sin(2\theta)}{g}$$

To **invert the elevation angle** θ, we rearrange the range formula:

$$\sin(2\theta) = \frac{Rg}{v_0^2}$$

Taking the arcsine of both sides and simplifying, we obtain:

$$\boxed{\theta = \frac{1}{2}\arcsin\left(\frac{Rg}{v_0^2}\right)} \tag{7.2}$$

This is the theoretical relationship between the elevation angle, range, and initial velocity under vacuum conditions.

7.3.2.1 Taylor Expansion with Air Resistance

Expand equation (7.2) using Taylor series expansion: $\theta = 45^\circ + k_1 \cdot (R/\beta) + k_2 \cdot (R/\beta)^2 + \ldots$

(1). Benchmark Solution: Optimal Angle in Vacuum

When the range R satisfies $\frac{Rg}{v_0^2} = 1$ (i.e., $v_0 = \sqrt{Rg}$), $\sin(2\theta) = 1$, and solving it gives $\theta = 45^\circ$, which is the optimal angle under vacuum conditions.

(2). Introducing Air Resistance Correction

When air resistance exists in reality, the ballistic coefficient $\beta = \frac{m}{\rho C_d A}$ (where m is the projectile mass, ρ is the air density, C_d is the drag coefficient, and A is the cross-sectional area) characterizes the projectile's resistance to drag. Drag causes range loss, so the optimal angle needs to be corrected.

(3). Dimensionless Parameter and Taylor Expansion

Define the dimensionless parameter $\epsilon = \frac{R}{\beta}$ (the ratio of range to ballistic coefficient, characterizing the degree of drag influence). Taking the vacuum optimal angle 45° as the benchmark, expand θ in a Taylor series at $\epsilon = 0$:

$$\theta(\epsilon) = \theta_0 + \theta_0'\epsilon + \frac{1}{2}\theta_0''\epsilon^2 + \ldots$$

where:

- $\theta_0 = 45^\circ$ (vacuum optimal angle)
- The first-order derivative term $\theta_0' = k_1$, and the second-order derivative term $\frac{1}{2}\theta_0'' = k_2$. The coefficients are determined by drag models (such as perturbation analysis, numerical simulation, or experimental data fitting).

(4). Expansion Result

Substituting $\epsilon = \frac{R}{\beta}$, we get:

$$\theta = 45^\circ + k_1 \cdot \left(\frac{R}{\beta}\right) + k_2 \cdot \left(\frac{R}{\beta}\right)^2 + \ldots$$

where k_1 and k_2 are constants related to the projectile's aerodynamic characteristics (for example, through drag work analysis or experimental fitting, $k_1 \approx 0.01^\circ \cdot \beta/\text{m}$, and the specific values need to be determined based on actual parameters). By neglecting the second-order small quantities, we obtain the final calculation formula:

$$\boxed{\theta = 45^\circ + k_1 \cdot \left(\frac{R}{\beta}\right)} \tag{7.3}$$

7.3.3 Azimuth Deviation Estimation

In the presence of wind, it is not feasible to derive a valid formula to reasonably estimate horizontal deflection. Thus, in practical situations, it can only be achieved by the correction table method mentioned above or by correcting the artillery firing direction based on experience.

7.3.4 Ultra-Simplified Field Formulas

For rapid mental calculation under typical conditions.The approximate formulas for initial velocity and elevation angle are as follows:

$$v_0 \approx 3\sqrt{R} \times 1.2 \tag{7.4}$$

$$\theta = 45^\circ + k_1 \cdot \left(\frac{R}{\beta}\right) \tag{7.5}$$

It is worth noting that this simplified formula is an empirical formula derived under the condition of completely neglecting the effects of air drag and wind on the artillery. It may have a relatively large deviation from actual conditions, and part of the errors can be corrected through the correction tables mentioned earlier.

8 Conclusions

This study establishes a theoretical framework for artillery projectile trajectory modeling based on simplified aerodynamic and kinematic assumptions, systematically describing the flight behavior of projectiles under constrained conditions. By treating the projectile as a rigid sphere, adopting the quadratic drag law, and approximating atmospheric parameter variations, the model effectively captures core trajectory characteristics within its valid scope—including the relationship between drag and velocity, as well as altitude-dependent aerodynamic effects—providing a foundational tool for analyzing the dynamics of short-to-medium range artillery.

For such a spherical projectile with a maximum muzzle velocity of 450 m/s, a mass of 5 kg, and a diameter of 11 cm, where the target is located at a horizontal distance of 1200 meters and both the artillery and the target are situated at an altitude of 500 meters above sea level, the relationship between feasible initial velocities and launch elevation angles is illustrated in the following figure:

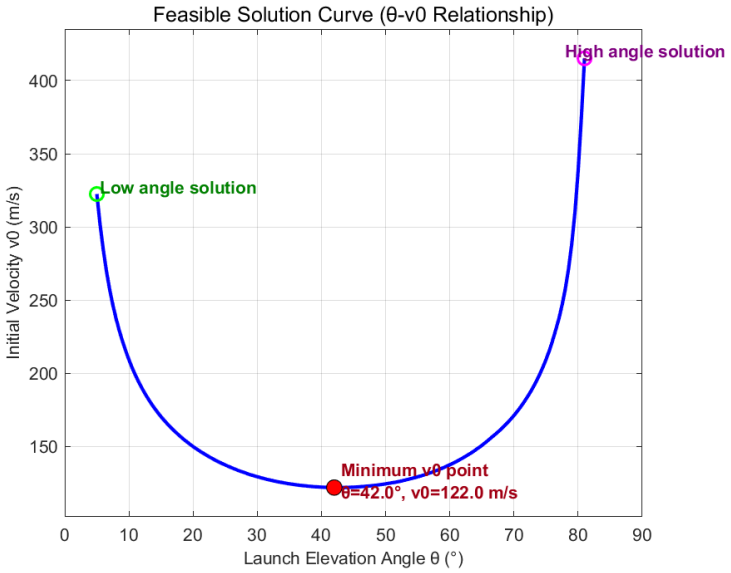


Figure 28: Feasible Solution Curve

Considering comprehensively different horizontal distances, varying altitudes, and diverse wind effects, we have derived the final projectile flight model for specific practical scenarios. First, we obtained preliminary base firing tables and correction tables through simulation calculations. Subsequently, a nomogram suitable for manual plotting by artillery officers was developed based on the initial firing tables. Finally, simplified formulas under specific conditions were derived from the simplified model to assist in mental calculations, with corrections supplemented by practical experience and the correction tables.

9 Model Limitations and Deficiencies

☹**Simplification of Projectile Geometry and Material:** The model treats the projectile as a rigid sphere, while actual artillery projectiles have diverse shapes and material properties. It ignores shape-induced aerodynamic lift and material deformation under high-speed flight, leading to deviations in drag coefficient and trajectory prediction for projectiles with complex flight attitudes.

☹**Idealization of Atmospheric Parameter Variation:** Atmospheric parameters are assumed to vary regularly with altitude, but real-world atmospheres involve turbulence, sudden temperature changes and non-uniform wind fields. These factors disrupt the predicted regularity of air density and dynamic viscosity, causing inaccuracies in aerodynamic drag calculation and trajectory correction.

☹**Neglect of Long-Range Geophysical Effects:** The model neglects Coriolis force and Earth's curvature. For long-range artillery or projectiles with extended flight time, their effects become non-negligible and may lead to significant target misses.

☹**Validity Range of Quadratic Drag Law:** The quadratic drag law is assumed to hold universally, but the drag coefficient depends on Reynolds number and Mach number. Drastic changes in projectile speed cause nonlinear variations in the drag coefficient, reducing the law's accuracy and introducing errors in resistance-related calculations.

☹**Quasi-Static Process Approximation:** Treating trajectory evolution as a quasi-static process ignores transient dynamic effects. Aerodynamic systems require finite time to establish force and thermal balance, and this approximation may underestimate unsteady aerodynamic forces for projectiles with rapid speed changes, affecting real-time trajectory adjustment precision.

10 Model Improvement Directions

☺**Refine Projectile Aerodynamic and Material Modeling**Incorporate actual projectile shapes and material properties. Establish non-spherical aerodynamic models and integrate high-speed deformation mechanisms, calibrated with experimental or simulation data.

☺**Optimize Atmospheric Environment Dynamic Simulation**Integrate turbulence, non-uniform wind and sudden temperature changes. Add real-time meteorological feedback to dynamically adjust atmospheric parameters.

☺**Incorporate Long-Range Geophysical Effects**Include Coriolis force and Earth's curvature for long-range scenarios. Derive location-dependent geophysical correction formulas based on latitude and flight parameters.

☺**Upgrade Drag Coefficient and Resistance Law Modeling**Refine drag coefficient's nonlinear relationship with Reynolds and Mach numbers. Adopt a segmented resistance model for transonic and supersonic regimes.

☺**Develop Unsteady Aerodynamic Simulation Framework**Replace quasi-static approximation with unsteady models. Capture transient dynamic effects and force-thermal balance response time for rapid speed changes.

References

[1] Zhao, B. Y., & Chen, Z. H. (2020). Computational simulation of the flight trajectory of rotating spheres based on aerodynamics [Jiyu kongqi donglixue de xuanzhuan qiuti feixing guiji de jisuan moni] (in Chinese). *Physics and Engineering*, 30(3), 50-54.

[2] Zhou, H., Shen, W., Wang, T. F., & Lu, Z. Y. (2025). Study on the influence of Coriolis force on the launch of long-range projectiles [Keli'oli li dui yuan shecheng paodan fashe de yingxiang yanjiu] (in Chinese). *Physics Bulletin*, 2025(6), 138-141.

[3] Fan, L., & Zhou, L. (2024). Research on parallel computation of multiple trajectories based on the fourth-order Runge-Kutta method [Jiyu sijie Longge-Kuta fa de duo daodan bingxing jisuan yanjiu] (in Chinese). *Aeronautical Science and Technology*, 35(9), 101-110.

A Appendix

A.1 A.1 Python Code Excerpt

```
import numpy as np
from scipy.integrate import solve_ivp
from scipy.optimize import least_squares
g0 = 9.80665
R_earth = 6371000.0
rho0 = 1.225
H_scale = 8500.0
omega_earth = 7.2921e-5
m = 5.0
d = 0.11
A = np.pi * (d / 2) ** 2
Cd = 0.47
def gravity(z, lat_deg):
    lat_rad = np.deg2rad(lat_deg)
    g_lat = g0 * (1 + 0.0053024 * np.sin(lat_rad) ** 2 -
  0.0000058 * np.sin(2 * lat_rad) ** 2)
    r = R_earth + z
    return g_lat * (R_earth / r) ** 2
def air_density(z):
    return rho0 * np.exp(-z / H_scale)
def ode_system(t, state, vw, lat_deg):
```

```
    x, y, z, vx, vy, vz = state
    if z < 0:
        return [0, 0, 0, 0, 0, 0]
    v_rel_x = vx - vw[0]
    v_rel_y = vy - vw[1]
    v_rel_z = vz - vw[2]
    v_rel = np.sqrt(v_rel_x**2 + v_rel_y**2 + v_rel_z**2)
    if v_rel < 1e-6:
        return [vx, vy, vz, 0, 0, -gravity(z, lat_deg)]
    rho = air_density(z)
    F_drag = 0.5 * Cd * rho * A * v_rel**2
    ax_drag = -(F_drag / m) * (v_rel_x / v_rel)
    ay_drag = -(F_drag / m) * (v_rel_y / v_rel)
    az_drag = -(F_drag / m) * (v_rel_z / v_rel)
    g_local = gravity(z, lat_deg)
    ax = ax_drag
    ay = ay_drag
    az = az_drag - g_local
    return [vx, vy, vz, ax, ay, az]
def solve_v0_theta_phi(target_xyz, launch_xyz, vw, lat_deg,
  v0_max=550.0):
    x0, y0, z0 = launch_xyz
    xt, yt, zt = target_xyz
    dx, dy, dz = xt - x0, yt - y0, zt - z0
    horizontal_dist = np.hypot(dx, dy)
    phi_init = np.rad2deg(np.arctan2(dy, dx))
    g_local_at_launch = gravity(z0, lat_deg)
    if dz > 50:
        v0_guess = v0_max * 0.95
        theta_guess = 45.0
    else:
        v0_guess = v0_max * 0.85
        avg_horizontal_speed = v0_guess *
  np.cos(np.deg2rad(30)) * 0.6
        t_guess = horizontal_dist / avg_horizontal_speed if
  avg_horizontal_speed > 0 else 10.0
        if horizontal_dist > 0:
            theta_guess_rad = np.arctan((dz + 0.5 *
  g_local_at_launch * t_guess**2) / horizontal_dist)
        else:
            theta_guess_rad = np.pi / 2
        theta_guess = np.rad2deg(theta_guess_rad)
    theta_guess = np.clip(theta_guess, 10.0, 85.0)
    v0_bounds = (50.0, v0_max)
    theta_bounds = (5.0, 85.0)
    phi_bounds = (phi_init - 45, phi_init + 45)
    def scaled_residuals(params):
        v0, theta_deg, phi_deg = params
        theta_rad = np.deg2rad(theta_deg)
        phi_rad = np.deg2rad(phi_deg)
```

```
        vx0 = v0 * np.cos(theta_rad) * np.cos(phi_rad)
        vy0 = v0 * np.cos(theta_rad) * np.sin(phi_rad)
        vz0 = v0 * np.sin(theta_rad)
        state0 = [x0, y0, z0, vx0, vy0, vz0]
        max_flight_time = max(2 * v0 / g_local_at_launch +
50, 5.0)
        def hit_event(t, state, *args):
            if state[2] < 0:
                return 0
            if state[5] > 0:
                return 1.0
            return state[2] - zt
        hit_event.terminal = True
        hit_event.direction = -1
        try:
            sol = solve_ivp(
                ode_system, [0, max_flight_time], state0,
args=(vw, lat_deg),
                events=hit_event, dense_output=True,
max_step=0.05, rtol=1e-6, atol=1e-6
            )
        except:
            return [1e6, 1e6, 1e6]
        if sol.status == 1 and sol.t_events and
sol.t_events[0]:
            try:
                final_state = sol.sol(sol.t_events[0][0])
            except:
                return [1e5, 1e5, 1e5]
        else:
            final_state = sol.y[:, -1]
            max_z_reached = np.max(sol.y[2, :])
            if max_z_reached < zt:
                height_deficit = zt - max_z_reached
                final_x, final_y = final_state[0],
final_state[1]
                horizontal_error = np.hypot(final_x - xt,
final_y - yt)
                scale_x = max(abs(dx), 1.0)
                scale_y = max(abs(dy), 1.0)
                scale_z = max(abs(dz), 1.0)
                return [
                    horizontal_error / scale_x + 20,
                    horizontal_error / scale_y + 20,
                    height_deficit / scale_z + 200
                ]
            else:
                return [1e4, 1e4, 1e4]
        if not np.all(np.isfinite(final_state)):
            return [1e5, 1e5, 1e5]
```

```
        error_x = final_state[0] - xt
        error_y = final_state[1] - yt
        error_z = final_state[2] - zt
        scale_x = max(abs(dx), 1.0)
        scale_y = max(abs(dy), 1.0)
        scale_z = max(abs(dz), 1.0)
        return [error_x / scale_x, error_y / scale_y, error_z
/ scale_z]
    res = least_squares(
        scaled_residuals, x0=[v0_guess, theta_guess,
phi_init],
        bounds=([v0_bounds[0], theta_bounds[0],
phi_bounds[0]],
                [v0_bounds[1], theta_bounds[1],
phi_bounds[1]]),
        ftol=1e-8, xtol=1e-8, gtol=1e-8, max_nfev=400,
verbose=0
    )
    final_params = res.x
    final_v0 = final_params[0]
    check_flight_time = max(2 * final_v0 / g_local_at_launch
+ 30, 2.0)
    theta_rad = np.deg2rad(final_params[1])
    phi_rad = np.deg2rad(final_params[2])
    vx0 = final_v0 * np.cos(theta_rad) * np.cos(phi_rad)
    vy0 = final_v0 * np.cos(theta_rad) * np.sin(phi_rad)
    vz0 = final_v0 * np.sin(theta_rad)
    state0 = [x0, y0, z0, vx0, vy0, vz0]
    def event_check(t, state, *args):
        if state[2] < 0:
            return 0
        if state[5] > 0:
            return 1.0
        return state[2] - zt
    event_check.terminal = True
    event_check.direction = -1
    try:
        sol_check = solve_ivp(
            ode_system, [0, check_flight_time], state0,
args=(vw, lat_deg),
            events=event_check, dense_output=True
        )
    except:
        raise RuntimeError()
    physical_error_norm = 1e6
    if sol_check.status == 1 and sol_check.t_events and
sol_check.t_events[0]:
        try:
            final_state_check =
sol_check.sol(sol_check.t_events[0][0])
```

```
            if np.all(np.isfinite(final_state_check)):
                physical_error_norm =
    np.linalg.norm(final_state_check[:3] - np.array([xt, yt,
    zt]))
        except:
            pass
if __name__ == "__main__":
    launch_xyz = [0, 0, 500]
    target_xyz = [1200, 0, 600]
    vw = [0, -10, 0]
    lat_deg = 45.0
    v0_max = 450.0
    try:
        v0, theta_deg, phi_deg =
    solve_v0_theta_phi(target_xyz, launch_xyz, vw, lat_deg,
    v0_max)
        theta_rad = np.deg2rad(theta_deg)
        phi_rad = np.deg2rad(phi_deg)
        vx0 = v0 * np.cos(theta_rad) * np.cos(phi_rad)
        vy0 = v0 * np.cos(theta_rad) * np.sin(phi_rad)
        vz0 = v0 * np.sin(theta_rad)
        state0 = [launch_xyz[0], launch_xyz[1],
    launch_xyz[2], vx0, vy0, vz0]
        def hit_event_final(t, state, *args):
            if state[2] < 0:
                return 0
            if state[5] > 0:
                return 1.0
            return state[2] - target_xyz[2]
        hit_event_final.terminal = True
        hit_event_final.direction = -1
        max_flight_time_final = max(2 * v0 /
    gravity(launch_xyz[2], lat_deg) + 50, 5.0)
        sol_final = solve_ivp(
            ode_system, [0, max_flight_time_final], state0,
    args=(vw, lat_deg),
            events=hit_event_final, dense_output=True,
    max_step=0.01, rtol=1e-7, atol=1e-7
        )
        plot_trajectory_3d(sol_final, target_xyz)
        if sol_final.status == 1 and sol_final.t_events and
    sol_final.t_events[0]:
            impact_time = sol_final.t_events[0][0]
            impact_state = sol_final.sol(impact_time)
    except RuntimeError as e:
        print()
```

Listing 1: Python Code for Ballistic Trajectory Calculation

B Report on the Use of AI

B.1 Used AI System

The AI systems used in this work include **GPT-5** (a generative AI system developed by OpenAI) and **Claude 4.5 Sonnet** (a large language model developed by Anthropic),all accessed via their web interfaces.

B.2 The Query

The exact wording input into the AI tool was:

Table B.1: AI Models and Corresponding Queries Used in This Work

AI Model	Query
GPT-5 (OpenAI)	*"Could you please introduce the ODE45 computational module in MATLAB? ."*
Claude 4.5 Sonnet (Anthropic)	*"How to conduct result analysis and visualization for Step 6?"*

B.3 The Full Output

The complete output generated by the AI tool is as follows:

B.3.1 GPT-5 output

```
Overview of the Principles, Usage, Common Options, and
    Considerations of ode45 in MATLAB for Standardized Use
    in Ballistic or Other Physical Model Numerical
    Integration
Introduction and Algorithm Principles
$ode45$ implements the Dormand Prince 4(5) explicit
    Runge Kutta method (also known as RK45). It estimates
    the local truncation error by simultaneously computing
    the differences between the 4th and 5th-order
    solutions and adaptively adjusts the step size to
    satisfy given relative (RelTol) and absolute (AbsTol)
    error tolerances.
It is suitable for "non-stiff" ordinary differential
    equation (ODE) systems. For problems exhibiting rapid
    decay scales or stiffness behavior, stiff solvers (such
    as ode15s) should be considered.
Basic Usage
Typical call:
[t, Y] = ode45(@odefun, tspan, y0, options);
```

```
The form of odefun: dYdt = odefun(t, Y) or dYdt = odefun(t,
    Y, params) (parameters can be passed via an anonymous
    function).
tspan = [t0 tf] or specify a vector of specific moments to
    obtain solutions at particular time points.
Returns t as a vector of unequally spaced time points
    determined by the adaptive step size, and Y as the
    corresponding state matrix (size(Y) = length(t)
    length(y0)).
Common odeset Configurations and Functions
Relative/absolute error control:
options = odeset('RelTol',1e-6,'AbsTol',1e-8);
Event detection (termination conditions, zero crossing):
options = odeset('Events',@eventsFcn);
The event function can return [value,isterminal,direction]
    to detect conditions such as landing or crossing and
    halt integration.
Maximum step size/initial step size control:
options = odeset('MaxStep',0.1,'InitialStep',1e-3);
Output function/monitoring:
'OutputFcn' is used for callbacks at each step,
    facilitating visualization or statistical collection (e
    .g., step size distribution).
Vectorization hint:
options = odeset('Vectorized','on') (if odefun supports
    vectorized evaluation, it can enhance speed).
Numerical Characteristics and Practical Recommendations
Adaptive step size: Step size may significantly decrease at
    points of drastic dynamics change or nonlinear
    transitions (e.g., shock wave transitions) to ensure
    local accuracy, but it may increase the number of
    function calls.
Robust parameter passing: Avoid using global variables;
    prefer passing parameters via anonymous functions:
    ode45(@(t,y) odefun(t,y,params), tspan, y0, options);
Trade-offs between accuracy and efficiency: Smaller RelTol/
    AbsTol results in more function calls; during parameter
    sweeps (multiple repeated integrations), it is
    advisable to relax tolerances or perform coarse
    filtering first.
Output handling: Since t comprises adaptive points, if post
    -processing on a fixed time grid is needed, deval(sol,
    t_query) can be used or tspan can be specified as the
    desired time vector during the ode45 call (though this
    enforces interpolation at these points while still
    integrating internally with adaptive step sizes).
Stiff problems and alternatives: For stiff systems (rapidly
    decaying modes), stiff solvers such as ode15s, ode23s,
    etc., should be used, and numerical Jacobians can be
    provided to accelerate convergence.
```

```
Event detection accuracy: Event detection utilizes internal
    interpolation to locate zero crossings, usually more
    accurate than coarse sampling, but can be improved by
    adjusting tolerances and MaxStep.
Brief Example (Including Events)
Integration with a landing event (pseudocode):
matlab
function [t,Y] = run_ballistic(params)
  odefun = @(t,y) ballistic_ode(t,y,params);
  options = odeset('Events',@(t,y) landing_event(t,y,params
    ), 'RelTol',1e-6,'AbsTol',1e-8);
  [t,Y] = ode45(odefun, [0 100], y0, options);
end
Common Problems and Debugging
Integration failure or step size approaching zero: Check
    for singularities or discontinuities in the model (e.g
    ., Cd discontinuity at critical Ma), transitioning to
    smooth variations or reducing the minimum step size
    limit.
Solutions "skipping" critical events at expected time
    points: Use Events or reduce MaxStep.
Unstable results or oscillations: This may arise from
    overly wide tolerances or equation stiffness; it may be
    necessary to use a stiff solver or adjust the error
    control strategy.
Conclusion
ode45 is a versatile and efficient non-stiff ODE solver in
    MATLAB, well-suited for most physical modeling problems
    (such as ballistic simulations) due to its adaptive
    step size and local error estimation. Practical usage
    should combine event detection, error control, and
    performance optimization strategies to ensure the
    physical reliability and computational efficiency of
    the results. If you need example code or optimization
    suggestions tailored to your model, I can provide more
    specific guidance based on your existing code (e.g.,
    your uploaded ballistic_ode/landing_event).
```

B.3.2 Claude 4.5 Sonnet output

```
Step 6: Result Analysis and Visualization Process
As a scientist proficient in various physical models and
    skilled in using MATLAB for modeling, I will describe
    the result analysis and visualization process in Step
    6. This phase focuses on handling multiple solutions (
    effective launch angle    and initial speed v0),
    including filtering, statistical analysis, typical
    solution extraction, tabular output, and multi-figure
```

```
    visualization. Based on MATLAB code, it ensures that
    the analysis reflects physical significance (such as
    energy minimization, low/high ballistic characteristics
    ) and visualization is intuitive (such as U-shaped
    curves). Below is a concise description of the process,
     emphasizing logic and implementation.
Preparation Phase: Filtering and Typical Solution
    Extraction
Filtering Valid Solutions: Extract convergent and non-NaN
    solutions from the scanning results, obtaining
    theta_valid and v0_valid arrays. Calculate the number
    of valid solutions and success rate (e.g., fprintf
    output "Number of valid solutions: %d / %d (Success
    rate: %.1f%%)"), used to assess the robustness of the
    scan.
Typical Solution Extraction:
Minimum Energy Solution: Calculate initial kinetic energy E
     = 0.5 * m * v0^2, identify    and v0 corresponding to
    the minimum E (physically represents the optimal
    efficiency launch).
Low Ballistic Example: Select the minimum    (short flight
    time, suitable for quick strikes).
High Ballistic Example: Select the maximum    (high
    trajectory, suitable for obstacle clearance).
Flight Time Calculation: Loop through and call
    compute_trajectory to obtain t_flight for each valid
    solution, providing progress feedback every 20 outputs
    (to avoid long calculation blind spots).
Analysis Phase: Statistical Analysis and Tabular Output
Statistical Analysis: Calculate and print key metrics, such
     as minimum/maximum feasible angles (fprintf outputs
    angle ranges), average/median initial speeds, etc.
    Physical significance: reveals the distribution of
    feasible solutions, e.g., boundary limits leading to no
     solution due to angle exceeding limits.
Tabular Output: Use fprintf to print a list of solutions,
    including serial number,   , v0, energy E, flight time
    t, range error error, and label typical solutions (e.g
    ., "    Minimum energy solution"). The table header
    includes explanations of energy definitions and the
    meaning of error. Ensure the error <1 m to validate
    accuracy.
Physical Insights: Analyze the differences between low/high
     ballistic trajectories (short flight time for low
    ballistic, strong obstacle clearance for high ballistic
    ), supporting tactical applications.
Visualization Phase: Multi-Subplot Layout
Overall Setup: Create a figure (position
    [100,100,1400,800], white background), using a 2 3
    subplot layout for increased professionalism.
```

```
Subplot 1: v0- Curve (U-shaped curve) : plot(theta_valid,
    v0_valid, 'b-'), marking the minimum energy point (red
    circle), low ballistic (green square), high ballistic
    (purple diamond). Title "Variation of Initial Speed
    with Launch Angle (U-shaped Curve)" and add legend
    explanations. Physical significance: shows the U-shaped
    distribution of multiple solutions, with minimum v0
    corresponding to energy optimal.
Subplots 2-3: Other Analysis Graphs (Optional Extensions) :
    Such as flight time vs. launch angle (plot(theta_valid
    , t_flight, 'g-'), marking the minimum energy point
    with a red star). Title "Variation of Flight Time with
    Launch Angle." Use grid on to enhance readability.
Subplots 4-6: Comparison of Typical Trajectories: Combine
    into a larger subplot, plotting three trajectories (Y
    (:,1) vs Y(:,2)):
Minimum Energy: Solid blue line.
Low Ballistic: Dashed green line.
High Ballistic: Dotted purple line.
Reintegrate each trajectory using ode45 (with event
    detection), line width 2.5, and add labels like "
    Minimum Energy:  =%.1f , v_0=%.1f m/s, E=%.1f kJ".
    Use hold on to overlay, with legends for explanation.
    Physical significance: visually compares trajectory
    shapes, range consistency (around 1200 m), and height
    differences.
Style Optimization: Uniform xlabel/ylabel (font size 12),
    title (bold 13), varied colors/line styles to avoid
    monotony.
Post-Processing and Verification
Time Recording: Use toc to calculate total time, output via
    fprintf to ensure efficiency (around seconds).
Validation: Check range errors, physical reasonableness of
    trajectories (e.g., peak height of high ballistic > low
    ballistic). If extended, could add energy distribution
    heat maps or 3D views.
MATLAB Implementation Tips: All calculations are loop-
    friendly for parallelism, and if solutions are numerous
    (>1000), vectorize for acceleration. Support output
    export (e.g., saveas(gcf, 'ballistic_analysis.png')).
This process seamlessly integrates from data filtering to
    insightful visualization, expected to enhance
    understanding of the ballistic multiple solutions. If
    specific code snippets or parameter adjustments are
    needed, please let me know!
```