Problem Chosen
B

Team Control Number
204

Analysis of Artillery Firing Parameters

Summary

This paper develops a two-dimensional exterior-ballistics model which aims to determine, under the constraint of a maximum muzzle velocity of $450\ m/s$, the multiple combinations of muzzle velocity and firing angle that allow cannons from $150-200$ years ago to hit a target at $1200\ m$, and to design a paper-and-pencil procedure for estimating firing parameters that can be used by artillery officers without access to modern computational tools.

In Problem I, starting from a rigid-body trajectory, we use dimensional analysis and order-of-magnitude estimates to demonstrate that buoyancy, Magnus force, and the Coriolis force are negligible, thereby reducing the dynamics to a particle ballistic model. For air resistance, we introduce an exponentially decaying air-density profile with altitude and construct a piecewise model that explicitly captures the rapid rise of shock wave drag in the transonic regime, so that the deceleration of the projectile more closely matches realistic exterior ballistics. We then build a solution framework combining inner Runge–Kutta numerical integration, outer nonlinear root finding, multi-initial-value search, and duplicate-solution elimination, which yields in a single run several physically distinct firing solutions. Among these, by balancing flight time, wind sensitivity, and engineering practicality, we select the small-angle limiting solution with firing angle 4° and muzzle velocity $450\ m/s$ as the recommended optimal solution.

In Problem II, within the same exterior-ballistics framework, we explicitly include horizontal tailwinds and headwinds, and reduce the hit condition for targets at $1000-1500\ m$ with different altitude differences and wind speeds to a linear wind correction for the muzzle velocity in the vicinity of the "recommended small-angle branch", written as$v_0(U) \approx v_0(0) + kU$. Numerical experiments show that, near the preferred solutions with small firing angle and short time of flight, the required muzzle velocity depends almost perfectly linearly on the wind speed. By reintegrating the trajectories and collecting the distribution of altitude errors, we quantitatively evaluate the error of this first-order linear approximation and confirm its validity. Exploiting this property, we precompute three firing tables on a range–altitude grid. An artillery officer needs only to look up the estimated range, altitude difference, and wind speed, and then perform simple calculations to obtain firing parameters.

Keywords: exterior ballistics, particle ballistic model, shock wave drag, wind correction, linear validation, firing-table design

Contents

1 Introduction

1.1 Background

As a historically significant military weapon, artillery has always centered its combat operations around ballistic calculations[1]. From the 19th century to the early 20th century, artillery officers had to rapidly estimate accurate muzzle velocities and firing angles without the aid of computers or calculators, in order to ensure that projectiles hit their targets. This task involved fundamental projectile kinematics as well as the complex effects of air resistance, elevation, and wind conditions. Consequently, developing a ballistic estimation method that was both precise and readily applicable in the field was of significant military value.

1.2 Problem Restatement

This problem considers a cannon from 150-200 years ago, firing spherical projectiles with a mass of $5\ kg$ and a diameter of $11\ cm$, achieving a maximum muzzle velocity of $450\ m/s$.

The core problem consists of two parts:

• The muzzle velocity and elevation angle required to hit a target at a horizontal range of $1200\ m$ and an elevation of $500\ m$ must be determined.

• The development of a practical method is required for artillery officers to rapidly estimate successful firing parameters for various targets at horizontal ranges of $1000 - 1500\ m$, under different elevation and wind speed conditions.

2 Initial Assumptions and Notations

2.1 Initial Assumptions

1. The projectile is a rigid body with symmetrical geometry and mass distribution, and the eccentricity is zero[2]. Only rotational motion about its longitudinal axis is considered, with no attitude oscillations in other directions.

2. The motion is characterized by four degrees of freedom: three translational directions in space (horizontal forward, vertical upward, perpendicular to the trajectory plane) and one rotational direction about the longitudinal axis[3].

3. The angle of attack remains small throughout flight, allowing the attitude to rapidly reach a stable state. The dynamic changes of the actual angle of attack are replaced by an equilibrium angle of attack, disregarding any attitude oscillations or oscillations.

4. Air density decreases gradually with increasing altitude. Near-surface temperature remains constant, maintaining atmospheric stability without complex disturbances such as turbulence or sudden temperature changes.

5. Weather conditions are normal with no strong convection. Horizontal wind speeds remain below $10\,m/s$, and the average vertical wind speed is close to zero.

6. After launch from the gun muzzle, the projectile travels through open space without terrain obstruction or ground effect[4] influence. The initial launch direction exhibits no lateral deviation.

2.2 Notations

Table 1: Notations

Symbol	Definition	Numerical value
g_0	Surface gravitational acceleration	$9.81\ m/s^2$
ρ_0	Air density at sea level	$1.225\ kg/m^3$
m	Mass of the projectile	$5\ kg$
D	Diameter of the projectile	$0.11\ m$
S	Maximum cross-sectional area	$9.5 \times 10^{-3}\ m^2$
$v_{0,max}$	Maximum muzzle velocity	$450\ m/s$
H	Atmospheric scale height	$8400\ m$
C_d	Drag coefficient	Determined by operating conditions and model
C_l	Magnus force coefficient	

3 Physical Analysis

3.1 Force Analysis

3.1.1 Gravity

According to the law of universal gravitation:

$$g(h) = g_0\left(\frac{R}{R+h}\right)^2 \tag{1}$$

In this problem, $h \ll R$, therefore we keep gravity constant, $g_0 = 9.81\ m/s^2$.

3.1.2 Air Buoyancy

According to Archimedes principle, air buoyancy is:

$$F_{float} = \rho(h)Vg(h) \tag{2}$$

In this problem, considering the air density at sea level ($\rho_0 = 1.225 kg/m^3$), it is calculated that $F_{float} = 0.0084N$, which accounts for only $0.017\ \%$ of the gravity, so it can be neglected in the ballistic calculation.

3.1.3 Air Resistance

Due to the symmetry of the sphere, the aerodynamic drag force F_D acts in the direction opposite to the velocity v. The relationship between aerodynamic drag force F_D and the sphere's flight velocity v can be obtained through simplified modeling. In $Figure\ 1$, the

disc represents the sphere's maximum cross-section and is orthogonal to the velocity direction.

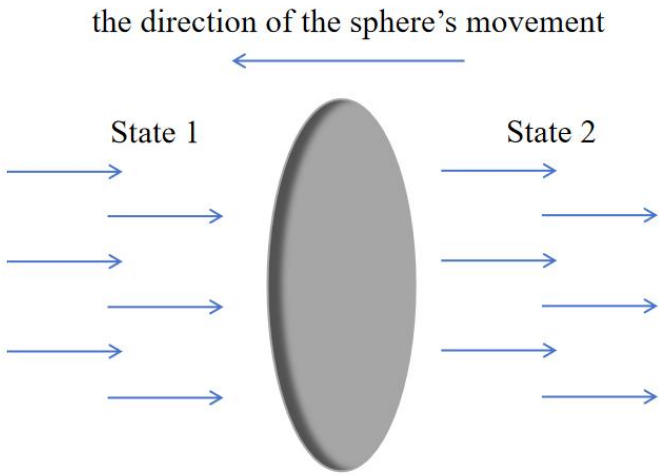

Figure 1: Airflow Condition on Both Sides of the Disc

As shown in $Figure\ 1$, the pressure experienced on both sides of the disc differs. In $State\ 1$, the airflow has not yet passed through the sphere, with pressure p_1 and relative velocity $v_1 = v$. In $State\ 2$, the airflow is completely blocked by the disc, with pressure p_2 and relative velocity $v_2 = 0$. According to Bernoulli's principle:

$$p_1 + \frac{1}{2}\rho v_1^2 = p_2 + \frac{1}{2}\rho v_2^2 \tag{3}$$

From equation (3), the pressure difference across the cross-section is:

$$p_1 - p_2 = \frac{1}{2}\rho v^2 \tag{4}$$

Let the area of the disc be S. The drag force acting on the disc is the pressure difference across its two surfaces:

$$F_D = (p_1 - p_2)S = \frac{1}{2}\rho S v^2 \tag{5}$$

Equation (5) represents the simplified drag relationship for a spherical model. The actual spheres is not a flat disc; air experiences velocity losses, friction forces arise as air flows over the spherical surface, and sphere rotation also affects drag, making real-world scenarios more complex.[5] We introduce the air resistance coefficient C_d to normalize these effects, allowing air resistance to be expressed as:

$$\vec{F}_D = (p_1 - p_2)S = -\frac{1}{2}C_d\rho S v^2 \frac{\vec{v}}{v} \tag{6}$$

Here, based on the problem context, we set C_d to the engineering default value of 0.47.

3.1.4 Coriolis Force

It is evident that the Earth is rotating at a constant angular velocity, denoted ω_0.[6] Consequently, the consideration of the necessity for the Coriolis force arises when selecting a reference frame at rest relative to the Earth：

$$\vec{F}_{\text{cor}} = 2m\vec{v} \times \vec{\omega} \tag{7}$$

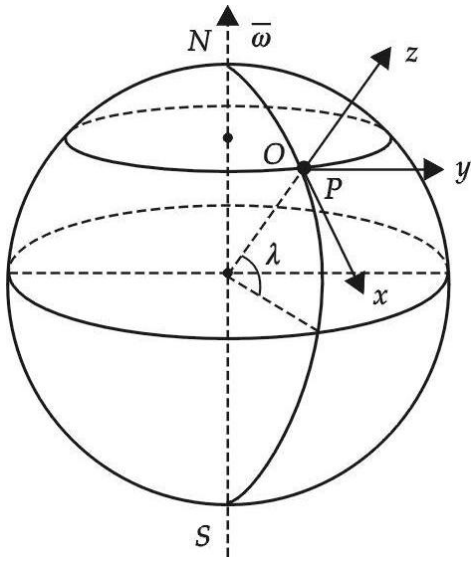

Figure2: The Space Rotation Reference Frame on the Earth's Surface

3.1.5 Magnus Force

The Magnus force generated by a rotating sphere arises from pressure differences caused by varying flow velocities across its surfaces. Here, we consider a case where the rotation axis is perpendicular to the sphere's velocity direction, as shown in Figure 3.

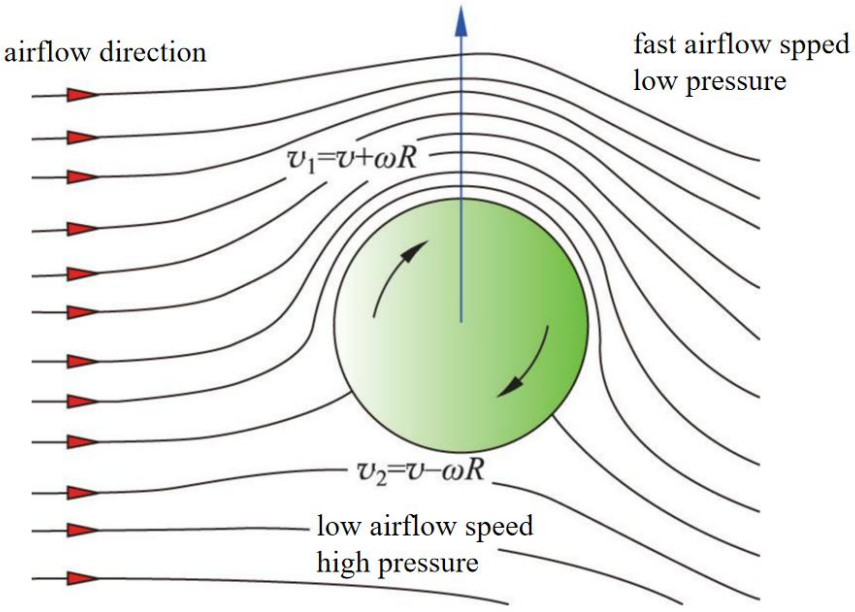

Figure 3: Schematic Diagram of the Magnus Force

When a rotating sphere travels through the air, the flow velocity on the side facing the airflow slows down ($v_2 = v - \omega R$), while the velocity on the side following the airflow increases ($v_1 = v + \omega R$).[5] According to Bernoulli's equation:

$$p_1 + \frac{1}{2}\rho v_1^2 = p_2 + \frac{1}{2}\rho v_2^2 \tag{8}$$

the pressure difference between the two sides generates a Magnus force perpendicular to ω and v, whose magnitude is:

$$\begin{aligned} |\vec{F}_{\mathrm{M}}| &= S(p_2 - p_1) \\ &= \pi R^2 \cdot 2\rho v \omega R \\ &= 2\pi\rho R^3 v\omega \end{aligned} \tag{9}$$

Introducing the coefficient C_l allows the expression for the Magnus force to be consistent with Equation (6). Considering the direction of the Magnus force yields:

$$\vec{F}_{\mathrm{M}} = \frac{1}{2} C_l \rho S v^2 \cdot \frac{\vec{\omega} \times \vec{v}}{|\vec{\omega} \times \vec{v}|} \tag{10}$$

However, after consulting the literature and performing manual calculations, we found that for a model of 19th-century cannon, the effects of rotation and the Magnus force are dispersed and far smaller than those of gravity and drag. Therefore, in subsequent discussions, we will not adopt the initially assumed 4DoF spin model, but instead consider the point particle ballistic model[7].

3.2 Variation of Air Density with Altitude

3.2.1 Static Equilibrium Equation

Consider a thin layer of air in the atmosphere, where the pressure difference is balanced by gravity:

$$dp = -\rho g dh \tag{11}$$

Here, dp represents the change in pressure within a height dh, ρ denotes the air density, and g is the gravitational acceleration.

3.2.2 Ideal Gas State Equation

Assuming air is an ideal gas:

$$p = \frac{\rho R T}{M} \tag{12}$$

Here, R is the gas constant, T is temperature, and M is the molar mass of air.

3.2.3 Isothermal Assumption

When temperature varies insignificantly with altitude (such as in the troposphere), T may be assumed constant. Combining equation (11) (12) yields:

$$\frac{d\rho}{\rho} = -\frac{gdh}{\frac{RT}{M}} = -\frac{gMdh}{RT} \tag{13}$$

3.2.4 Integral Solution

After integration, the exponential decay relationship of air density with altitude is obtained:

$$\rho(h) = \rho_0 e^{-\frac{gMh}{RT}} \tag{14}$$

Here, ρ_0 is the air density at sea level, and h is the altitude.

3.2.5 Introduction of Elevation Parameter

Typically, the atmospheric scale height $H = \frac{RT}{gM}$ is substituted, simplifying equation (14) to:

$$\rho(h) = \rho_0 e^{-\frac{h}{H}} \tag{15}$$

Here, H is approximately 8,400 meters, representing the characteristic altitude at which air density is halved.

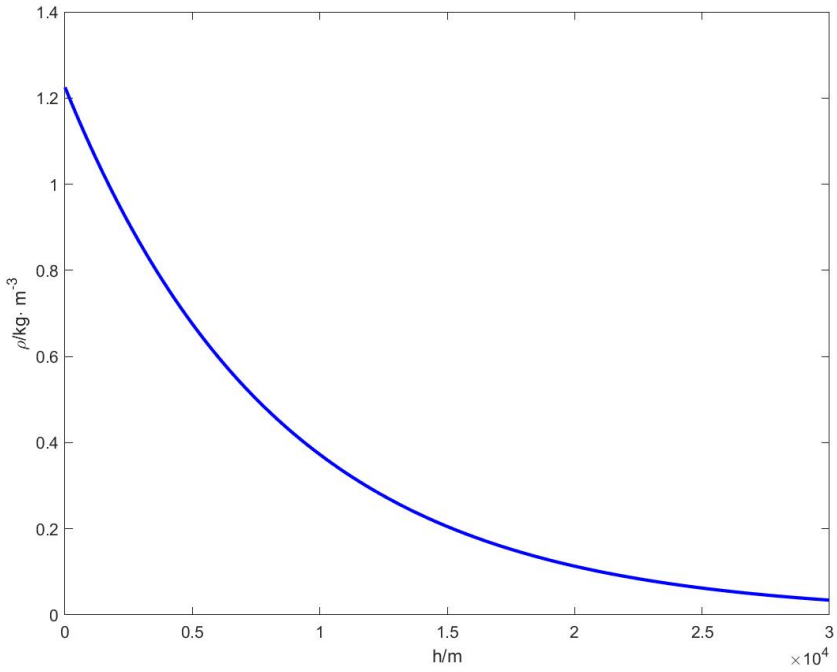

Figure 4: Variation of Air Density with Altitude

3.3 Variation of Drag Coefficient with Mach Number

Upon consulting the literature, we find that the drag coefficient $C_d(M)$ of a spherical projectile varies as a piecewise function with respect to the Mach number M:

$$C_d(M) \approx \begin{cases} 0.45 & , \quad M < 0.7 \\ 0.45 + 0.55 \cdot \dfrac{M - 0.7}{0.4} & , \quad 0.7 \le M < 1.1 \\ 1.0 - 0.2 \cdot \dfrac{M - 1.1}{0.4} & , \quad 1.1 \le M < 1.5 \\ 0.8 & , \quad M \ge 1.5 \end{cases} \tag{16}$$

In reality, the function curve is non-linear, but we have simplified it here by making a linear approximation.

4 Problem I Solution

4.1 Problem Analysis

Given the target point's horizontal distance $v_{tar} = 1200m$, elevation $y_{tar} = 500m$, and the constraint that the maximum muzzle velocity $v_m = 450m/s$, we aim to determine multiple sets of initial velocities v_0 and firing angles θ that enable the projectile to hit the target while accounting for air resistance and air density.

4.2 General Thought

Given the difficulty in obtaining analytical solutions for air density varying with altitude and air resistance varying with both air density and velocity, the problem is approached by numerically integrating $x(v_0, \theta)$ and $y(v_0, \theta)$ to find multiple solutions to the two-dimensional nonlinear system of equations:

$$F(v_0, \theta) = \begin{bmatrix} x(v_0, \theta) - x_{\text{tar}} \\ y(v_0, \theta) - y_{\text{tar}} \end{bmatrix} = 0 \tag{17}$$

We chose a solution structure combining “inner-layer numerical integration + outer-layer nonlinear equation solving + multi-start search”:

• Inner layer: Given a set (v_0, θ), use ordinary differential equation numerical integration to obtain the trajectory and impact point;

• Outer Layer: Treat the deviation between the impact point and the target as the equation residual, and use $fsolve$ to iteratively refine (v_0, θ);

To obtain multiple physically distinct solutions, construct a non-uniform initial guess grid covering low/medium/high elevation angles and initial velocities. Independently call $fsolve$ for each initial point, then perform deduplication and filtering on the obtained solutions.

4.3 Model Building

The projectile is treated as a point mass, subject to gravity and air resistance. Taking the horizontal and vertical coordinates as x and y, respectively, and the velocity components as v_x and v_y , then the quantity of state $y = [x,\ y,\ v_x,\ v_y]^T$, $v = \sqrt{v_x^2 + v_y^2}$.

The acceleration is given by the following formulae:

$$\begin{aligned}\dot{x} &= v_x \\ \dot{y} &= v_y \\ \dot{v}_x &= -\frac{F_D}{m} \cdot \frac{v_x}{v} \\ \dot{v}_y &= -g - \frac{F_D}{m} \cdot \frac{v_y}{v}\end{aligned} \tag{18}$$

Initial conditions are:

$$\begin{aligned}x(0) &= 0 \\ y(0) &= 0 \\ v_x(0) &= v_0 \cos\theta \\ v_y(0) &= v_0 \sin\theta\end{aligned} \tag{19}$$

4.4 Algorithm Design

4.4.1 Solving Framework

We designed a program to solve the aforementioned structure of "inner-layer numerical integration + outer-layer nonlinear equation solving + multi-start search".

We employed MATLAB's adaptive-step Runge–Kutta[8] integrator $ode45$ to perform integration over the time interval $[0,40]s$. Given (v_0, θ), we locate the cutoff point in the integration results where the horizontal coordinate first exceeds the target horizontal distance $x > x_{tar}$. The corresponding (x, y) coordinates at this time are recorded as the approximate landing point (x_{final}, y_{final}). Ultimately, we construct the residual function:

$$\begin{aligned}F_1 &= x_{\text{final}} - x_{\text{tar}} \\ F_2 &= y_{\text{final}} - y_{\text{tar}}\end{aligned} \tag{20}$$

Call $fsolve$ to find a solution where both $F1$ and $F2$ approach zero simultaneously for the variables (v_0, θ).

Additionally, the residual function imposes a penalty for exceeding the maximum initial velocity ($v_0 > v_{0,max}$) by making the residual proportional to the excess quantity, ensuring the iteration process consistently satisfies the initial velocity constraint.

4.4.2 Multi-start Search and Initial Grid Design

Due to the highly nonlinear relationship between (v_0, θ) and the impact point in the presence of air resistance and a variable-density atmosphere, a single target often corresponds to multiple ballistic trajectories. To extract multiple physical solutions from the locally convergent $fsolve$, the program employs a "multi-start" approach:

• Firing angles are divided into three physically meaningful intervals: low angle ($5° - 30°$), medium angle ($31° - 50°$), and high angle ($51° - 85°$). Within each interval, initial values are sampled at increased density to form a non-uniform angular grid, ensuring coverage of both low- and high-velocity trajectories.

• Initial velocities are sampled equidistantly within the range $[250, 450]\ m/s$, enabling the search to cover scenarios from medium-high velocities to near-maximum initial velocities.

• By combining the above angles and initial velocities via $meshgrid$, hundreds of initial guess points are generated. $fsolve$ is invoked independently for each point. Any result that successfully converges with a total landing deviation not exceeding $5mm$ is temporarily saved as a candidate solution.

This design of "non-uniform grid + multi-start local solving" enhances search efficiency while capturing multiple solution sets with significant differences across a large parameter space.

4.4.3 Solution Screening

Among the candidate solutions obtained through multi-start search, multiple sets of results often exist that are numerically very close and share identical physical meaning. To address this, the program first sorts all candidate solutions by (v_0, θ), then sequentially compares the differences between adjacent solutions:

• If the initial velocity difference does not exceed $2\ m/s$ and the firing angle difference does not exceed 5°, they are deemed duplicate solutions and discarded;

• Only when at least one of these two conditions shows a significant difference is the pair retained as a new valid solution.

The deduped solution set ensures that any two retained solutions differ by at least approximately 5° in angle and exhibit distinguishable initial velocity differences. This provides several distinctly different firing options for subsequent tactical selection. For these final solutions, the program performs ballistic integration again, plotting trajectories within the $0 - 1500m$ range and marking target points to visually validate the algorithm's results.

4.4.4 Result Presentation

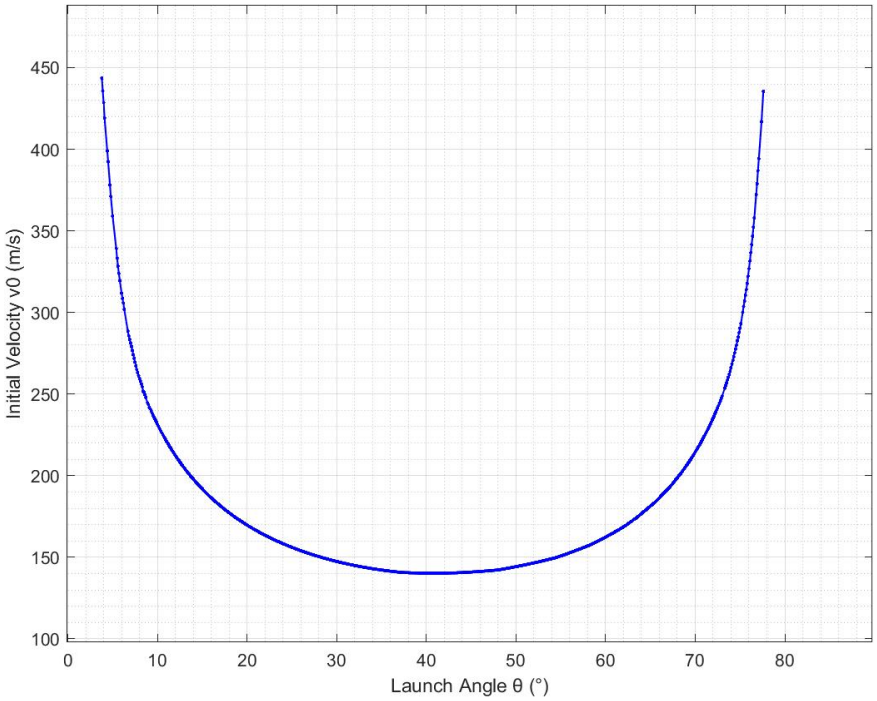

Figure 5: Relationship Between Firing Angle and Initial Velocity

When wind effects are disregarded and the altitude difference remains constant in Problem I, the relationship between the initial velocity and the firing angle can be obtained as shown in Figure 5 (where θ starts from 25°). It can be observed that multiple firing angles may correspond to different velocities, and at extreme angles ($\theta > 80°$ *or* $\theta < 30°$), the velocity becomes highly sensitive to changes in angle.

Considering that targets may deviate in practical scenarios, a set of solutions with the shortest airborne flight time (i.e., the fastest interception) is selected:

$$\theta = 4°, v_0 = 450 m/s$$

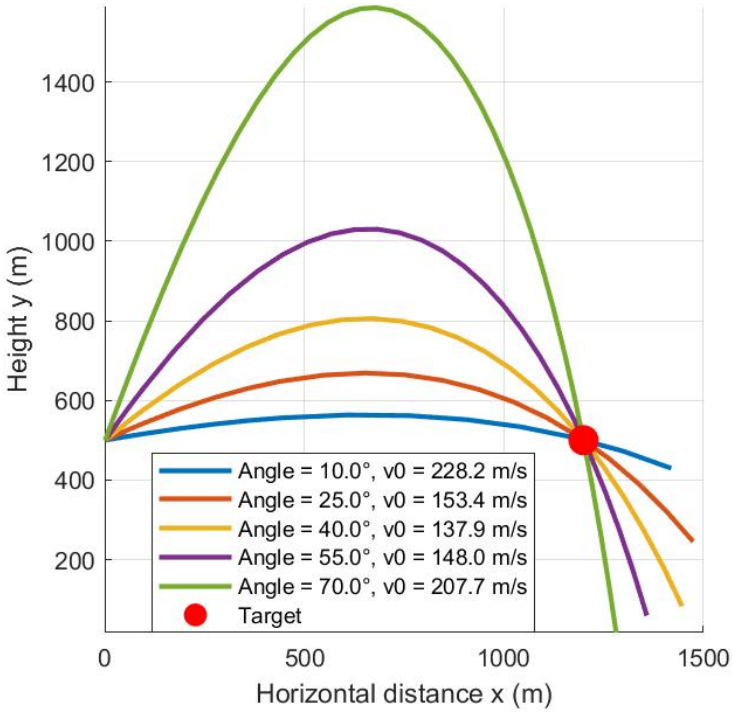

Figure 6: Trajectories for Five Fixed Firing Angles

4.5 Error Analysis of Coriolis Force

In the preceding numerical model, we assumed the projectile's motion was confined to a vertical plane containing both the firing point and target point, thereby neglecting the Coriolis force generated by Earth's rotation. To verify the validity of this assumption, we estimated five representative firing conditions and analyzed the lateral deviation caused by the Coriolis force.

To obtain an upper bound for this deviation, we employed a simplified model:

• Assuming the direction of the Earth's rotational angular velocity is perpendicular to the direction of the projectile's velocity, the magnitude of the Coriolis acceleration is at its maximum.

• Consider only the acceleration perpendicular to the plane generated by the horizontal velocity component $v_x = v_0 cos\theta$, which means the reduction of v due to drag is ignored, yielding

$$a_z = 2\omega v_x = 2\omega v_0 \cos\theta \tag{21}$$

• The projectile's flight time remains calculated using the analytical result without considering Earth's rotation, assuming identical starting and ending altitudes:

$$t \approx \frac{2v_0 \sin\theta}{g} \tag{22}$$

• During this time interval, treat a_z as constant, yielding lateral displacement:

$$\Delta z = \frac{1}{2} a_z t^2 = \omega v_0 \cos\theta \left(\frac{2v_0 \sin\theta}{g}\right)^2 \tag{23}$$

Substituting the five sets of firing parameters from Figure 6 into the above equation yields the lateral displacement caused by the Coriolis force (absolute value only):

Table 2: Lateral Displacement Caused by Coriolis Force

Number	$v_0/(m \cdot s^{-1})$	$\theta/°$	$\lvert\Delta z\rvert/m$
1	228.2	10.0	0.8
2	153.4	25.0	1.3
3	137.9	40.0	1.9
4	148.0	55.0	3.0
5	207.7	70.0	6.7

As can be seen, at the 1.2 km range in this problem, the lateral deviation caused by the Coriolis force is generally within 10 m and increases with the firing angle. When compared to the horizontal range, the maximum deviation is approximately

$$\frac{|\Delta z|_{max}}{x_{tar}} = \frac{6.7}{1200} \times 100\% = 0.56\% \tag{24}$$

Considering air resistance effects would further reduce the relative error between $|\Delta z|$ and x. This indicates that the error introduced by the Coriolis force is significantly smaller than errors from air resistance models and factors like actual wind fields, having negligible impact on the required muzzle velocity and firing angle.

Therefore, under the accuracy requirements of this problem, simplifying the motion to a two-dimensional problem and neglecting the Coriolis force's effect on the projectile trajectory is reasonable. The actual lateral deviation can be compensated for in real firing by making small-angle corrections to the aiming direction, without significantly altering our primary conclusions regarding the initial velocity and firing angle.

5 Problem II Solution

5.1 Problem Analysis

Problem II builds upon Problem I by introducing constraints closer to real combat scenarios. In more general combat situations, the target's horizontal range lies between 1000 m and 1500 m, its altitude between 200 m and 800 m, and there exists varying degrees of tailwind or headwind (wind speed range approximately $-10\ m/s$ to $10\ m/s$). We are tasked with providing a method for an artillery officer without access to

computers or calculators to rapidly estimate the appropriate firing angle and muzzle velocity under varying wind conditions. This estimation should be achievable on paper using tables and minimal mental calculations.

To avoid redundant derivations, we directly adopt the external ballistic model and numerical solution framework established in Problem I for Problem II. Specifically:

• We continue to use the two-dimensional rectangular coordinate system established in Problem I, with the horizontal direction as the x-axis, and the vertical upward direction as the y-axis.

• The projectile is subject only to gravity and air resistance. Air resistance is approximated as proportional to the square of the relative airflow velocity, with the drag coefficient and frontal area consistent with those in Problem I.

• The empirical formula for air density variation with altitude, the numerical integration method (e.g., Runge–Kutta-based time stepping), and the subroutine for "calculating impact coordinates given launch parameters" all adopt the results from Problem I.

Additionally, when establishing non-standard external ballistic models, only the effects of longitudinal and lateral wind variations are typically considered. Since wind changes generally occur parallel to the ground, their influence on the vertical plane is negligible.[9] Building upon this foundation, the primary distinction in Problem II lies in the need to explicitly account for the impact of horizontal wind.

We define the wind direction as collinear with the x-axis (positive for tailwind, negative for headwind), with wind speed denoted as U. In this case, the air resistance experienced by the projectile depends on the difference between the projectile velocity and the wind velocity, that is, the relative airflow velocity $\vec{u} = \vec{v} - \vec{U}$. In numerical calculations, simply replacing the velocity $\vec{v}$ from Problem I with $\vec{u}$ yields the ballistic equation incorporating wind speed effects. This approach maintains consistency in the model structure while facilitating expansion upon the original program.

5.2 General Thought

To address the requirement of "accounting for wind while enabling rapid estimation without calculators", we adopted the approach of "Analyzing single-point → Establishing first-order wind correction → Generating a firing table on a range–altitude grid". The overall steps can be summarized as follows:

1. Structural Analysis of Multiple Solutions for Single Target Points

First, select typical target points specified in the problem. Scan the firing angle at multiple wind speeds and use the numerical framework from Problem 1 to solve for the initial velocity required for a hit, yielding multiple "initial velocity–firing angle" curves. Comparison reveals: in the region of smaller angles and higher initial velocities, the curves for different wind speeds converge closely, indicating low sensitivity to wind speed variations.

2. Flight Time and Solution Optimization

Under windless conditions, flight times for the same target point at different firing angles were calculated, revealing that flight time increases with angle. Therefore, among multiple hit solutions for the same target, we prioritize the solution with the smallest angle and shortest flight time as the "recommended solution" for subsequent research.

3. First-Order Linear Wind Correction Model

To satisfy the engineering constraint of "no calculator", we approximate the effect of wind speed on the required initial velocity as a first-order linear correction near the recommended angle:

$$v_0(U) \approx v_0^{(0)} + kU \tag{25}$$

Here, v_0 represents the initial velocity in calm conditions, and k denotes the first-order wind correction coefficient.

We selected multiple wind speed points within the range $[-10, 10]\ m/s$. For each wind speed value, we calculated the precise $v_0(U)$ required for a hit. Subsequently, we determined k using least-squares linear regression and verified that the deviation from the target caused by this linear approximation was sufficiently small.

4. Generation of Firing Tables on Distance–Altitude Grids

After confirming that first-order linear corrections provide sufficient accuracy, we extend the above method to a two-dimensional grid covering ranges of $1000 - 1500m$ and target altitudes of $200 - 800m$. For each grid point (x, y), we first determine the recommended solution under windless conditions that satisfies the "small angle, short flight time" principle, recording the firing angle $\theta^*(x, y)$ and windless initial velocity $v_0^{(0)}(x, y)$. Then, under wind conditions of $U = \pm 10\ m/s$, we numerically solve the equation to estimate the wind correction coefficient $k(x, y)$ using finite differences. This ultimately produces three tables: the firing angle table $\theta^*(x, y)$, the windless initial velocity table $v_0^{(0)}(x, y)$, and the wind correction coefficient table $k(x, y)$. Artillery officers need only consult these tables and perform one multiplication and one addition according to the formula $v_0 = v_0^{(0)}(x, y) + k(x, y)U$. The firing angle is directly taken as a, thereby completing the correction for wind speed effects.

5.3 Structural Analysis of Multiple Solutions for Single Target Points

5.3.1 Correction of $v_0 - \theta$ Curves and Drag Models at Multiple Wind Speeds

To analyze the impact of wind speed on the solution structure, we first selected the typical target point specified in the problem statement: a horizontal distance $R = 1200m$, with the target altitude matching the cannon altitude (both at $500m$). Based on this, we defined a set of horizontal wind speeds $U = -10, -5, 0, 5, 10m/s$, representing strong headwind, weak headwind, calm conditions, weak tailwind, and strong tailwind, respectively.

For each given wind speed U, we scanned the launch angle within a specific range (e.g., $3° \leq \theta \leq 85°$) at fixed increments. Using the numerical solution framework established in Problem I, we invoked the solver for each angle to determine the initial velocity $v_0(\theta, U)$ required to hit the target. This process yielded multiple "initial velocity versus firing angle" curves for different wind speeds, as shown in Figure 7.

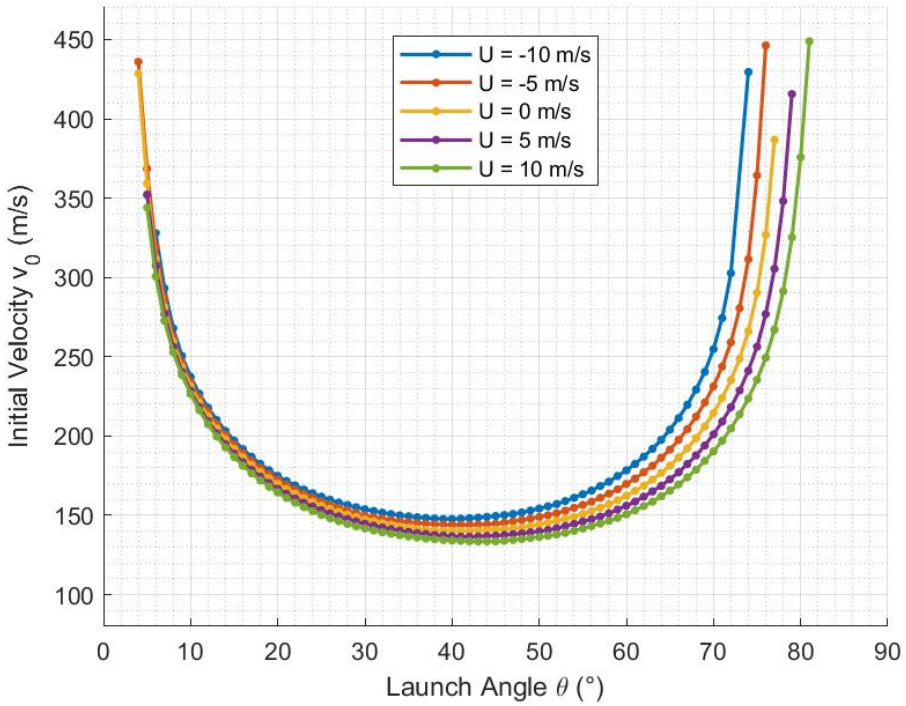

Figure 7: $v_0 - \theta$ Curves Under Different Horizontal Wind Speeds

The curves in the figure exhibit a typical "U-shaped" structure: near smaller and larger firing angles, the initial velocity required to hit the target increases significantly, while a minimum initial velocity point exists near medium firing angles. Curves for different wind speeds share similar overall shapes but are shifted along the velocity axis. Specifically, within the "small angle–high initial velocity" region of interest, the curves are relatively close together. This indicates that solutions within this angular range exhibit low sensitivity to wind speed variations, providing preliminary justification for selecting small-angle solutions in subsequent analyses.

5.3.2 Relationship Between Flight Time and Firing Angle

After defining the $v - \theta$ multi-solution structure under multiple wind speeds, we further investigated the relationship between flight time and firing angle under windless conditions. Specifically: With the target point fixed, we scanned the firing angle θ in specified increments. For each angle, we solved for the initial velocity $v_0(\theta, 0)$ required for impact. This initial velocity was then substituted into the external ballistic differential equation for integration, recording the first time $T(\theta)$ at which the projectile reached $x = R$.

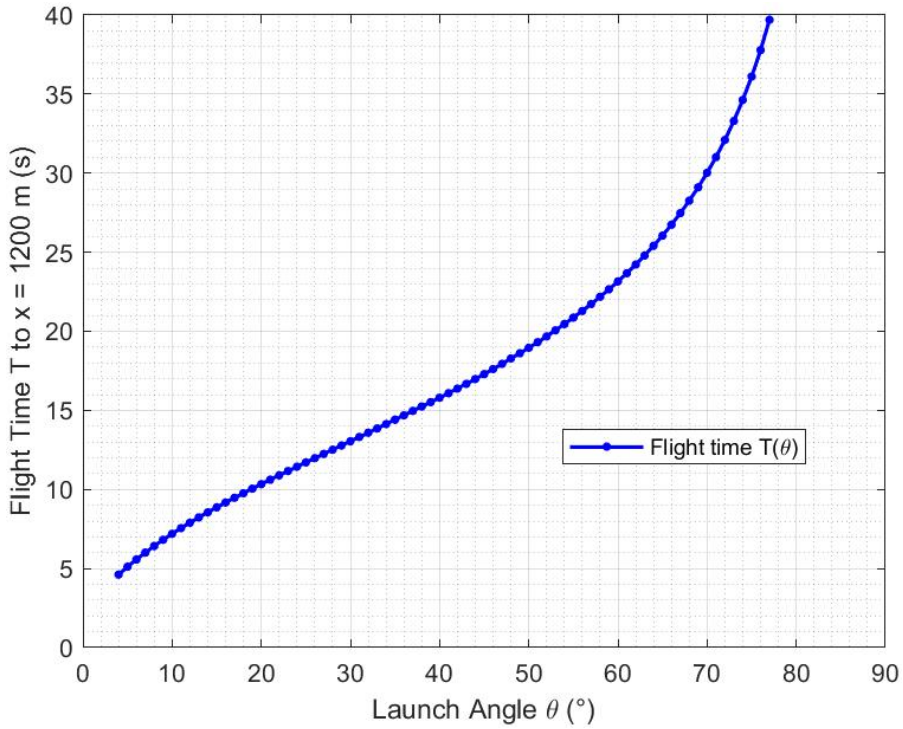

Figure 8: $T-\theta$ Curve Under No-wind Condition

Numerical results indicate that within the physically feasible solution range, the flight time T generally increases monotonically with the firing angle: the greater the angle, the longer the projectile remains airborne. For the same target point, solutions with smaller firing angles correspond to shorter flight times, while solutions with larger firing angles correspond to longer flight times.

5.3.3 Justification for Selecting the Small-Angle Solution

Based on the analysis above, we adopt the convention in subsequent modeling and trajectory generation that for each target point, the hit solution from the "small-angle branch" shall be prioritized as the recommended solution. This choice is not only grounded in the physical intuition of shorter flight times and reduced sensitivity to wind speed, but also gains further support from the following two engineering perspectives:

1. The impact of initial velocity control error on hit deviation is relatively minor.

In actual combat, artillery units struggle to precisely control the initial launch velocity to a specific theoretical value. As shown in Figure 7, within the selected "small angle–high initial velocity" range, the spacing between different wind velocity curves is narrow. Even when initial velocity fluctuates within a certain range, the corresponding solution remains within the "clustered" region where multiple curves overlap, resulting in relatively mild changes in the impact point. In contrast, in the large-angle region, the spacing between wind velocity curves significantly increases. An initial velocity error of the same magnitude will amplify into a larger point of impact deviation. Therefore, from the perspective of robustness to initial velocity errors, small-angle solutions offer greater safety.

2. Small-angle solutions exhibit higher accuracy under linear wind correction

In subsequent verification of first-order linear wind correction, representative recommended solutions were selected at both small and large angles. Linear fitting was

applied to recalculate the ballistic trajectory, evaluating the vertical deviation caused by linear approximation. Results indicate that in the small-angle region, the corrected initial velocity obtained using the linear relationship $v_0(U) \approx v_0^{(0)} + kU$ generally produces smaller height deviations at the point of impact compared to the large-angle region, with significantly smaller maximum vertical deviations. This demonstrates that the linear wind correction assumption holds more strongly in the small-angle branch, further validating the rationality of selecting the small-angle solution.

5.4 Numerical Verification of the Linear Assumption for First-Order Wind Correction

To verify the validity of the first-order wind correction model $v_0(U) \approx v_0^{(0)} + kU$, we conducted specialized numerical experiments at representative target points. To represent typical scenarios for small-angle branches, this section fixes the firing angle θ^* at $15°$ with the muzzle altitude matching the target altitude($R = 1200\ m, y_{cannon} = y_{target} = 500\ m$. We solved the "exact required initial velocity" $v_0(U)$ point-by-point within the wind speed range $[-10,10] m/s$, then performed linear fitting and deviation assessment based on these results.

First, for each given wind speed U, we used $fsolve$ to determine the initial velocity required for target hit at a fixed angle θ^*. To enhance convergence efficiency, the code provided an approximate analytical solution for the initial velocity based on the analytical projectile model as the initial condition:

$$v_0^2 \approx \frac{gR^2}{2\cos^2\theta^*(R\tan\theta^* - \Delta y)},\quad \Delta y = y_{\text{target}} - y_{\text{cannon}} \tag{26}$$

When the target is level with the muzzle:

$$v_0^2 \approx \frac{gR}{\sin(2\theta^*)} \tag{27}$$

Using this as the initial value, the numerically meaningful "true" solution $v_0(U)$ is obtained by minimizing the deviation between the landing point and the target point in both the x and y directions, and the corresponding flight time $T(U)$ is recorded.

After obtaining a set of data points $\{U_i, v_0(U_i)\}$, we fit the relationship $v_0^{\text{lin}}(U) = a + bU$ using the least squares method to obtain the coefficients a and b of the first-order linear model. The "$v_0(U) - U$ curve" shows that the black discrete points represent the numerically solved $v_0(U)$, while the red dashed line is the fitted straight line. The two almost perfectly coincide, indicating that under conditions $\theta^* = 15°,\ U \in [-10,10] m/s$ the required initial velocity exhibits a nearly strictly linear relationship with wind speed. The corresponding residuals are negligible, with errors well below 1% relative to typical initial velocities of approximately $190\ m/s$.

To further evaluate the impact of this linear approximation on hit accuracy, we employed the fitted $v_0^{lin}(U)$ as the initial launch velocity. Under the same wind speed U, we re-integrated the external ballistic equation, using linear interpolation[10] to determine the shell's altitude at $x = 1200\ m$ and calculated the vertical deviation $\Delta y(U) = y(x = R, U) - y_{\text{target}}$. Numerical results indicate that across the entire wind speed range, most vertical deviations fall within $\pm 0.5\ m$. Only near extreme tailwind conditions do deviations slightly increase, yet their absolute values remain below approximately $1 \sim 1.5\ m$, which is entirely acceptable.

In summary, at the representative small angle $\theta^* = 15°$, the initial velocity $v_0(U)$ required to hit the same target exhibits a highly linear relationship with horizontal wind speed. The additional deviation introduced by first-order wind correction is only a small quantity on the order of meters. Both theoretical fitting and practical deviation data confirm that the first-order linear wind correction model is reliable and sufficiently accurate within the wind speed range of this problem. This provides robust numerical support for the simplified method of "referencing tables for $v_0^{(0)}$ and k, then applying linear correction" in subsequent firing tables.

5.5 Construction Methods and Results of the Firing Table

5.5.1 Range–Altitude Grid Design and Parameter Constraints

Following the selection of small-angle branches and validation of the first-order wind correction model in the preceding section, this section extends the methodology to broader the range and altitude to generate firing tables for artillery reference. As specified in the problem statement, target positions are sampled on the following two-dimensional grid:

$$\text{Range: } x \in [1000,1500]m, \Delta x = 50\ m;$$

$$\text{Altitude: } y \in [200,800]m, \Delta y = 50\ m.$$

This yields several range and altitude points, totaling $N_x N_y$ target grid points. In all calculations, the muzzle altitude is fixed at $y_{cannon} = 500\ m$, so the grid encompasses both downward ($\Delta y < 0$) and upward ($\Delta y > 0$) firing scenarios.

To ensure engineering relevance, we impose an upper limit on initial velocity: when searching for the baseline solution under windless conditions, $v_0^{(0)} \le 400\ m/s$ is required, representing the conventional initial velocity achievable with standard propellant charges. When incorporating wind corrections, the initial velocity is expanded to $v_0^{(0)} \le 450\ m/s$ to allow for first-order corrections.

Simultaneously, the hit deviation threshold is set to $\|\Delta r\| \le 5 \times 10^{-3}\ m$ to ensure the calculated firing parameters can be considered numerically "accurate hits".

5.5.2 Small-Angle Foundation Solution Search under Windless Conditions

At each grid point (x, y), we first search for the fundamental solution of the "small-angle branch" under windless conditions ($U = 0$). To achieve this, we scan the firing angle point-by-point across the angular range $\theta \in [5°, 30°]$, $\Delta\theta = 1°$. For each angle θ, we approximate the initial velocity using the analytical solution for vacuum projectile motion as the initial guess for $fsolve$:

Equation (27) is used for nearly level firing (where $|\Delta y|$ is small), while Equation (26) is used for significant altitude differences, where R is the target's range.

Subsequently, under the constraint of $[v_{0,min}, v_{0,max}^{base}] = [100, 400]m/s$ and at a fixed angle, the residual function is invoked to solve for the initial velocity of the projectile hitting the target at $U = 0$. Internally, this solver integrates the ballistic equation using the Runge–Kutta-type $ode45$ solver and reconstructs the altitude at $x = R$ via linear interpolation, thereby returning the deviations of the impact point from the target point in both the x and y directions. All solutions satisfying the conditions of $v_0 \in [100,400]m/s$ and $\|\Delta r\| \le 5 \times 10^{-3}\ m$ are recorded as candidate solutions $(\theta_i, v_{0,i})$.

5.5.3 Calculation of Small-Angle Foundation Solutions and Wind Correction Coefficient

For each target point, we select a "recommended base solution" from the set of candidate solutions obtained in the previous step. To reflect the principle of "prioritizing smaller angles while staying as close as possible to 15°", the selection rule proceeds in two steps: First, search for candidate solutions within the preferred angle range $\theta \in [10°, 20°]$. If solutions exist, select the set closest to the center angle $\theta_c \approx 15°$. If no solutions exist within the preferred range, select the set with the smallest angle from all candidate solutions.

The obtained recommended solutions are denoted as $\theta^*(x, y)$ and $v_0^{(0)}(x, y)$, stored respectively in the angle table and the zero-wind initial velocity table.

After obtaining the base solution, we estimate the first-order wind correction coefficient $k(x, y)$ for this grid point at the same angle $\theta^*(x, y)$. The specific procedure is as follows: At wind speeds $U = -10\ m/s$ and $U = +10\ m/s$, respectively, invoke the solver with $v_0^{(0)}(x, y)$ as the initial value (allowing the initial velocity upper limit to be raised to $450\ m/s$) to obtain the initial velocities v_0^- and v_0^+ required to hit the target. If both solutions succeed and satisfy the accuracy and speed constraints, the wind correction coefficient is given by the central difference:

$$k(x,y) = \frac{v_0^+ - v_0^-}{U_+ - U_-} = \frac{v_0^+ - v_0^-}{20} \tag{28}$$

The resulting $k(x, y)$ values are entered into the wind correction coefficient table. For individual target points where no convergent solution can be found within the specified velocity range at a wind speed of $\pm 10\ m/s$, mark that point as NaN. This indicates that the point should be avoided in practical applications or that a more conservative interpolation strategy should be adopted.

5.5.4 Three firing Tables and Their Usage Methods

After completing calculations for all grid points, we obtained three mutually complementary two-dimensional tables:

Table 3: Firing Table $\theta^*(x, h)$

h \ x	1000m	1050m	1100m	1150m	1200m	1250m	1300m	1350m	1400m	1450m	1500m
200m	3	3	3	3	3	3	3	3	3	3	3
250m	3	3	3	3	3	3	3	3	3	3	3
300m	3	3	3	3	3	3	3	3	3	3	3
350m	3	3	3	3	3	3	4	3	3	3	3
400m	3	3	3	4	3	3	3	3	3	4	4
450m	3	3	3	3	4	3	4	4	3	6	4
500m	3	4	4	4	4	4	5	6	7	7	6
550m	7	7	6	6	8	7	8	8	9	9	9
600m	10	9	9	10	9	9	9	9	9	11	11
650m	13	12	13	11	12	12	13	11	11	12	12
700m	15	16	15	15	14	17	14	14	14	14	14
750m	17	17	17	19	16	18	17	16	16	16	16
800m	20	20	19	19	20	20	18	18	19	18	17

Table 4: Windless Initial Velocity Table $v_0^{(0)}(x,h)$

h \ x	1000m	1050m	1100m	1150m	1200m	1250m	1300m	1350m	1400m	1450m	1500m
200m	141.87	150.49	158.79	166.59	174.58	183.50	191.90	199.98	209.34	218.12	226.93
250m	153.04	162.14	171.04	179.37	187.77	197.30	206.22	214.82	224.74	233.92	243.88
300m	167.41	177.13	186.74	195.71	204.66	214.88	224.39	233.83	244.27	255.43	267.92
350m	186.86	197.35	207.84	217.57	227.40	238.38	236.80	260.84	274.37	289.19	306.26
400m	215.27	226.87	238.33	234.95	262.26	275.94	291.29	309.99	329.94	315.22	332.44
450m	266.43	281.35	297.55	318.89	298.36	360.88	334.25	354.15	434.35	315.06	424.04
500m	426.72	364.55	386.67	409.31	426.62	449.47	401.01	362.89	334.49	349.91	415.74
550m	337.95	349.07	435.03	448.29	328.39	401.42	353.73	369.91	337.93	349.49	365.92
600m	330.02	402.15	403.86	341.03	413.73	418.26	427.74	436.41	445.08	350.06	361.80
650m	321.28	378.94	319.53	445.13	371.03	373.71	331.28	444.53	449.58	400.64	408.19
700m	373.79	305.35	345.54	339.01	395.09	276.39	392.92	392.51	392.99	398.48	403.21
750m	437.65	403.39	379.89	283.69	415.33	310.47	346.71	399.00	394.97	397.28	400.86
800m	410.36	372.55	415.95	390.34	323.99	318.99	411.47	401.73	347.72	396.14	447.80

Table 5: Wind Correction Coefficient Table $k(x,h)$

h \ x	1000m	1050m	1100m	1150m	1200m	1250m	1300m	1350m	1400m	1450m	1500m
200m	−0.55	−0.59	−0.58	−0.61	−0.66	−0.71	−0.74	−0.77	−0.81	−0.87	−0.92
250m	−0.54	−0.59	−0.58	−0.60	−0.66	−0.71	−0.73	−0.77	−0.83	−0.93	−1.04
300m	−0.53	−0.59	−0.58	−0.58	−0.65	−0.71	−0.71	−0.80	−0.98	−1.12	−1.37
350m	−0.51	−0.58	−0.57	−0.56	−0.64	−0.76	−0.75	−1.13	−1.41	−1.63	−1.89
400m	−0.49	−0.57	−0.56	−0.58	−0.94	−1.21	−1.46	−1.74	−2.00	−1.87	−2.30
450m	−0.77	−0.80	−1.25	−1.60	−1.34	−2.10	−1.89	−2.16	−2.18	−1.75	−2.47
500m	−1.22	−1.42	−1.57	−1.63	−1.48	−1.78	−1.61	−2.05	−1.83	−2.15	−1.83
550m	−1.08	−1.17	−1.21	−1.40	−1.02	−1.56	−1.75	−1.88	−1.80	−2.01	−2.00
600m	−0.96	−0.94	−0.99	−1.01	−1.22	−0.69	−1.18	−1.60	−1.35	−1.91	−2.18
650m	−0.78	−0.75	−0.69	−0.74	−1.16	−1.56	−1.29	−0.94	−1.19	−1.71	−1.61
700m	−0.27	−0.75	−0.45	−0.70	−0.87	−0.66	−1.56	−1.09	−1.04	−1.34	−1.05
750m	0.30	0.05	−0.44	−0.48	−0.16	−0.81	−0.84	−1.10	−0.89	−1.38	−1.41
800m	0.64	0.04	0.32	−0.27	−0.57	−0.69	−0.03	−0.57	−1.07	−1.06	−1.61

$k(x,h)$ is the amount by which the initial velocity must be adjusted for each 1 m/s change in wind speed, with the unit being m/s.

Artillery officers need only:

1. Based on rough range and altitude estimates, locate the grid point (x,h) closest to the target in the three tables.
2. Read the recommended firing angle θ^* from the firing table.
3. Read $v_0^{(0)}$ and k respectively from the windless initial velocity table and wind correction factor table. For a given horizontal wind speed U, perform one multiplication and one addition according to Equation (25) to obtain the corrected initial firing velocity. Directly use θ^* as the firing angle. If the target position falls between grid points, simple interpolation between adjacent grid points or rounding to the nearest value may be used.

Additionally, we utilized three tables to generate color distribution maps for "Recommended firing angle–Target position", "Windless initial velocity–Target position", and "Wind correction coefficient–Target position". These maps provide an intuitive overview of the overall trends of each variable as range and altitude change. This also indirectly demonstrates that the firing table exhibits good spatial continuity, making it suitable for interpolation to achieve finer distance and elevation resolution.

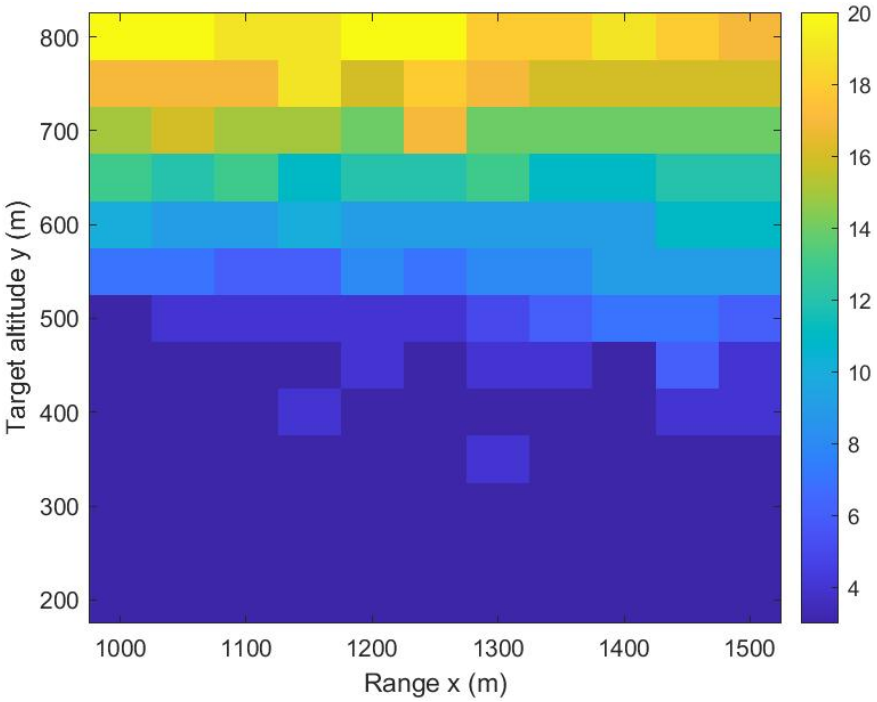

Figure 9: Recommended Firing Angle θ^*

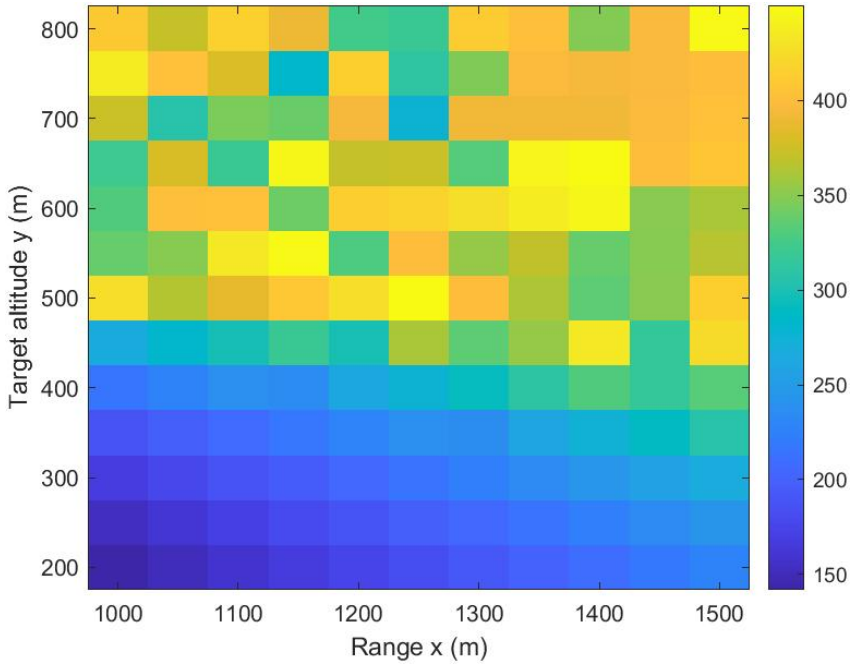

Figure 10: Required Muzzle Velocity $v_0^{(0)}$

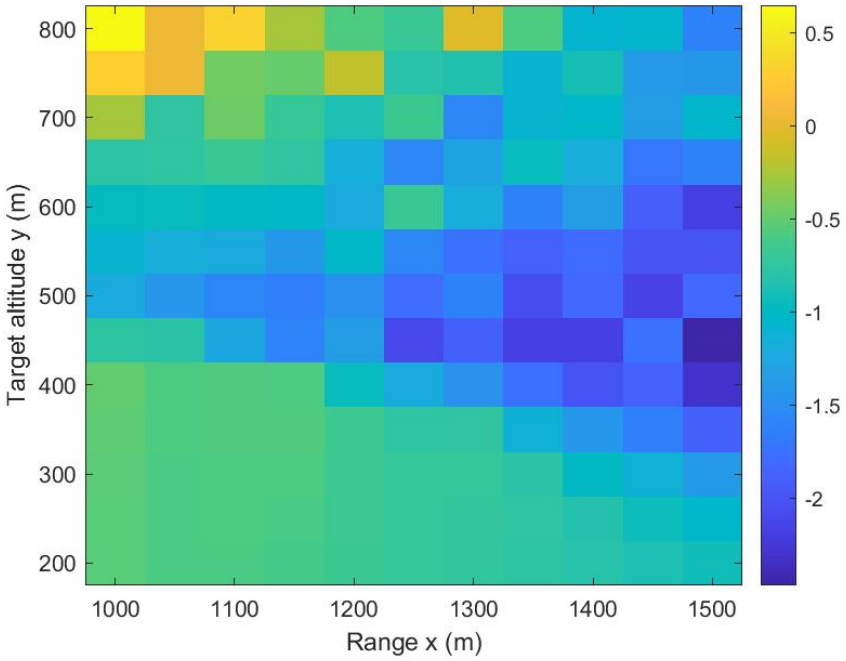

Figure 11: Wind Correction Coefficient k

6 Conclusion and Discussion

6.1 Conclusion

This paper addresses the ballistic calculation problem for 19th-century artillery by constructing a ballistic model based on the particle assumption, followed by numerical solution and result analysis. First, by simplifying the battlefield environment and projectile motion characteristics while neglecting minor factors such as the Magnus force and Coriolis force, the core influencing factors were identified as gravity, air resistance (including altitude-dependent density changes), and horizontal wind. Subsequently, a piecewise linearization approach was applied to the drag coefficient. Combined with the Runge-Kutta method, this enabled numerical ballistic calculations under multiple initial conditions, yielding firing tables and wind correction coefficients covering ranges from 1000 to 1500 meters.

6.2 Discussion

6.2.1 Strengths

1. Physically Accurate and Streamlined Modeling

Starting from rigid-body external ballistics, we employ order-of-magnitude analysis to eliminate secondary effects such as the Magnus force, buoyancy, and Coriolis force, resulting in a particle ballistic model incorporating only gravity and air resistance. Concurrently, we retain critical effects like altitude-dependent atmospheric density and the sharp increase in shock resistance near the speed of sound, balancing computational simplicity with physical fidelity.

2. Systematic and Practical Numerical Solution Framework

A unified framework combining “Runge-Kutta numerical integration + multi-initial-value nonlinear solver + solution deduplication screening” enables systematic exploration of multiple solutions for the same objective. By integrating flight time, wind sensitivity, and

engineering feasibility, a low-angle limit solution with an firing angle of approximately 4° and initial velocity of 450 m/s is selected as the recommended optimal solution for Problem I, demonstrating the integration of model analysis with engineering criteria.

3. Practical Tool with Outstanding Value Translation

In Problem II, the framework not only delivers precise numerical solutions across varying wind speeds but also validates the assumption that "initial velocity approximates wind speed linearly" while quantifying the upper bound of error. Based on this, three two-dimensional tables were constructed: a "Firing Table", a "Windless Initial Velocity Table", and a "Wind Correction Coefficient Table". This transformation from complex computational models to paper-based lookup algorithms demonstrates strong practicality and scalability.

6.2.2 Weaknesses

1. Numerous Idealized Assumptions Regarding the Environment and Projectile

• A simplified atmospheric model is employed, omitting explicit consideration of temperature variations with altitude, turbulence, gusts, and other random factors.

• Wind speed is assumed constant and distributed solely along the horizontal trajectory direction, disregarding crosswinds, vertical winds, and temporal variations.

• Engineering randomness such as initial velocity dispersion and manufacturing tolerances for mass and diameter is not incorporated, introducing additional dispersion in actual firing scenarios.

2. Accuracy Limitations at the Dynamic Level

Projectiles are treated as non-spinning particles undergoing only two-dimensional motion, ignoring effects like attitude changes, Magnus force, and three-dimensional trajectory deflection. Accuracy is limited at longer ranges or under rigid body models.

3. Limitations of Firing Tables and Validation

• Tables are pre-calculated for specified grids; targets outside these grids require interpolation or extrapolation, potentially introducing additional errors.

• Lacks systematic validation against historical field data or established external ballistics software.

References

[1] Han, Z. P. (2016). External Ballistics of Projectiles and Arrows [M]. Beijing Institute of Technology Press, 230-247.

[2] Hui, J. H., Wang, Y. S., & Wen, Q. (2025). Initial Conditions for Numerical Solution of Differential Equation Systems in External Ballistics of Rotating Projectile Rigid Bodies. Journal of Ballistics, 37(01), 36-43.

[3] Xiang F., Wang Y. S., Wen Q. & Wang G. Y. (2021). Review of Applications in Rigid Body External Ballistics of Uncontrolled Projectiles. Journal of Detection and Control, 43(04), 14-26.

[4] Lai C. G., Wen S. H., Zhai G. T., Chen J., & Guo Y. F. (2025). Aerodynamic Characteristics of Vertical Landing for Flying Cars Under Ground Effect. Journal of Chongqing University of Technology (Natural Science), 39(05), 45-54.

[5] Zhao, B. Y., & Chen, Z. H. (2020). Computational Simulation of Rotating Sphere Flight Trajectories Based on Aerodynamics. Physics and Engineering, 30(03), 50-54.

[6] Zhou H., Shen W., Wang T. F., & Lu Z. Y. (2025). Study on the Influence of Coriolis Force on Long-Range Projectile Launching. Bulletin of Physics, (06), 138-141.

[7] Liao G. H. (2012). Application of Ballistic Models in Simulating the Stability of Rotating Projectiles. Journal of Sichuan Ordnance Engineering, 33(05), 11-13.

[8] Pan T., Yang F. B., Zhu Y. & Yao X. C. (2015). FPGA Implementation for Solving Ballistic Differential Equations Based on Runge-Kutta. Computer Measurement and Control, 23(12), 4217-4220.

[9] Wang J. Q., Zhao Y. J., & Li P. F. (2025). Analysis and Calibration Study of Tank Armor Anti-Target Deviation Based on Wind Field [J]. Journal of Artillery Firing and Control, 46(02), 24-30.

[10] Gkritzapis D.N., Panagiotopoulos E. E., & Margaris D. P. (2008). Computational atmospheric trajectory simulation analysis of spin-stabilized projectiles and small bullets[J]. International Journal of Computing Science and Mathematics, 2(1–2), 53–72.

Appendices

```
% External ballistics multi-solution solver
% Launch and target altitude are both 500 m.

clear; clc; close all;

%% 1. Physical and geometric parameters
m    = 5;          % projectile mass (kg)
d    = 0.11;       % projectile diameter (m)
S    = pi*(d/2)^2; % reference area (m^2)

g    = 9.81;       % gravity (m/s^2)
rho0 = 1.225;      % air density at sea level (kg/m^3)
H    = 8400;       % atmospheric scale height (m)
c    = 340;        % speed of sound (m/s)

% Fire mission: 1200 m range, equal altitude 500 m
launch_y = 500;    % launch altitude (m)
target_x = 1200;   % horizontal range (m)
target_y = 500;    % target altitude (m)

% Allowed muzzle-velocity interval
v0_min = 100;      % lower bound (m/s)
```

```
v0_max = 450;      % upper bound (m/s)

%% 2. Solver settings
theta_list    = 0:1:90;   % scan firing angle (deg)
dev_threshold = 5e-3;     % tolerance for impact error (m)

options = optimoptions('fsolve', ...
    'Display','off', ...
    'FunctionTolerance',1e-5, ...
    'MaxFunctionEvaluations',2e4, ...
    'MaxIterations',1e3);

%% 3. Search over firing angles (fixed theta, solve for v0)
all_solutions = [];   % rows: [v0, theta, dx, dy]

for theta = theta_list
    theta_rad = theta * pi / 180;

    % Analytic guess without drag, equal launch/target height:
    % R = v0^2 * sin(2*theta) / g  ->  v0 = sqrt(g*R/sin(2*theta))
    sin2theta = sin(2 * theta_rad);
    if abs(sin2theta) <= 1e-3
        v0_guess = v0_max - 30;        % avoid singularity near 0° / 90°
    else
        v0_guess = sqrt(g * target_x / sin2theta);
    end

    % For very small or very large angles, push the guess towards v0_max
    if theta < 5 || theta > 85
        v0_guess = max(v0_guess, v0_max - 30);
    end

    % Clamp initial guess into admissible interval
    v0_guess = min(max(v0_guess, v0_min), v0_max);

    % Residual function: [dx; dy] at range x = target_x
    fun = @(v0) projectile_residual(v0, theta, m, S, g, rho0, H, c, ...
        target_x, target_y, launch_y, v0_min, v0_max);

    [v0_sol, fval, exitflag] = fsolve(fun, v0_guess, options);

    total_dev = norm(fval);
    if exitflag > 0 && v0_sol >= v0_min && v0_sol <= v0_max ...
            && total_dev <= dev_threshold
```

```
        all_solutions = [all_solutions; v0_sol, theta, fval(1), fval(2)];
    end
end

%% 4. Sort and print all valid solutions
solutions_sorted = sortrows(all_solutions, 2);   % sort by angle (deg)

fprintf('Number of valid solutions: %d\n', size(solutions_sorted,1));
for k = 1:size(solutions_sorted,1)
    fprintf(['Sol %2d: theta = %5.1f deg,  v0 = %7.2f m/s,  ', ...
             'dx = %+8.6f m,  dy = %+8.6f m\n'], ...
        k, solutions_sorted(k,2), solutions_sorted(k,1), ...
        solutions_sorted(k,3), solutions_sorted(k,4));
end

%% 5. Angle-velocity curve
figure('Position', [100, 100, 800, 600]);
plot(solutions_sorted(:,2), solutions_sorted(:,1), '.-', 'LineWidth', 1);
xlabel('Launch angle \theta (deg)');
ylabel('Initial velocity v_0 (m/s)');
title('Angle-velocity relation (launch & target altitude 500 m)');
grid on;
xlim([0, 90]);
ylim([v0_min - 20, v0_max + 20]);

%% ==================== Local functions ====================

function F = projectile_residual(v0, theta, m, S, g, rho0, H, c, ...
    target_x, target_y, launch_y, v0_min, v0_max)
% Residual [dx; dy] at the target range for a fixed firing angle.

    % Velocity constraint (penalty outside [v0_min, v0_max])
    if v0 < v0_min || v0 > v0_max
        F = [1e6 * (v0 - v0_min); 1e6 * (v0 - v0_max)];
        return;
    end

    theta_rad = theta * pi / 180;

    % State vector: y = [x; y; vx; vy]
    y0 = [0, launch_y, v0 * cos(theta_rad), v0 * sin(theta_rad)];

    % Integration time long enough to pass x = target_x
    t_end = 300;
```

```
    [~, y] = ode45(@(t,Y) projectile_ode(t, Y, m, S, g, rho0, H, c), ...
                   [0, t_end], y0);

    % Interpolate height when trajectory crosses x = target_x
    idx = find(y(:,1) >= target_x, 1);
    if isempty(idx)
        x_final = y(end,1);
        y_final = y(end,2);
    else
        x1 = y(idx-1,1); y1 = y(idx-1,2);
        x2 = y(idx,1);   y2 = y(idx,2);
        y_final = y1 + (target_x - x1) * (y2 - y1) / (x2 - x1);
        x_final = target_x;
    end

    F = [x_final - target_x; y_final - target_y];
end

function dy = projectile_ode(~, y, m, S, g, rho0, H, c)
% 2D point-mass model with exponential atmosphere and Mach-dependent Cd.

    vx = y(3);
    vy = y(4);
    v  = hypot(vx, vy);

    if v == 0
        dy = [vx; vy; 0; -g];
        return;
    end

    M   = v / c;                        % Mach number
    Cd  = Cd_of_Mach(M);                % drag coefficient
    rho = rho0 * exp(-y(2) / H);        % density vs altitude

    Fd  = 0.5 * rho * S * Cd * v^2;     % drag magnitude
    Fdx = -Fd * vx / v;
    Fdy = -Fd * vy / v;

    ax = Fdx / m;
    ay = -g + Fdy / m;

    dy = [vx; vy; ax; ay];
end

```

```
function Cd = Cd_of_Mach(M)
% Piecewise drag coefficient Cd(M) for a sphere.

    Cd = zeros(size(M));

    % Subsonic: M < 0.7
    idx1 = (M < 0.7);
    Cd(idx1) = 0.45;

    % Transonic: 0.7 <= M < 1.1, Cd: 0.45 -> 1.0
    idx2 = (M >= 0.7) & (M < 1.1);
    Cd(idx2) = 0.45 + 0.55 * (M(idx2) - 0.7) / (1.1 - 0.7);

    % Supersonic: 1.1 <= M < 1.5, Cd: 1.0 -> 0.8
    idx3 = (M >= 1.1) & (M < 1.5);
    Cd(idx3) = 1.0 - 0.2 * (M(idx3) - 1.1) / (1.5 - 1.1);

    % Hypersonic: M >= 1.5
    idx4 = (M >= 1.5);
    Cd(idx4) = 0.8;
end
```

```
% Firing table generator (Problem B, Mach-dependent Cd, small-angle branch)
% - 2D grid in (range x, target altitude y)
% - For each (x,y): scan elevation, keep the smallest-angle solution (no wind)
% - Then compute first-order wind correction v0(U) ≈ v0^(0) + k U

clear; clc; close all;

%% 1. Physical parameters
m    = 5;              % projectile mass (kg)
d    = 0.11;           % projectile diameter (m)
S    = pi*(d/2)^2;     % reference area (m^2)
g    = 9.81;           % gravity (m/s^2)
rho0 = 1.225;          % sea-level air density (kg/m^3)
H    = 8400;           % scale height (m)

gun_y  = 500;          % gun altitude (m)
v0_min = 100;          % minimum muzzle velocity (m/s)
v0_max = 450;          % maximum muzzle velocity (m/s)

dev_threshold = 5e-3;  % tolerance for impact error (m)
```

```
021
022 % fsolve settings
023 options = optimoptions('fsolve', ...
024     'Display','off', ...
025     'Algorithm','levenberg-marquardt', ...
026     'FunctionTolerance',1e-5, ...
027     'MaxFunctionEvaluations',2e4, ...
028     'MaxIterations',1e3);
029
030 %% 2. Range–altitude grid
031 x_list = 1000:50:1500;   % horizontal range (m)
032 y_list = 200:50:800;     % target altitude (m)
033
034 Nx = numel(x_list);
035 Ny = numel(y_list);
036
037 theta_table = NaN(Ny, Nx);   % recommended launch angle θ* (deg)
038 v0_table    = NaN(Ny, Nx);   % base muzzle speed v0^(0) without wind (m/s)
039 kwind_table = NaN(Ny, Nx);   % first-order wind coefficient k (m/s per m/s)
040
041 % angle scan range
042 theta_scan = 3:1:80;         % degrees
043
044 % candidate wind speeds for estimating k
045 U0_candidates = [10, 8, 6, 4, 2];  % m/s
046
047 fprintf('Generating firing tables (Mach-dependent Cd, minimal-angle branch)...\n');
048
049 %% 3. Loop over all grid points (x,y)
050 for ix = 1:Nx
051     target_x = x_list(ix);
052
053     for iy = 1:Ny
054         target_y = y_list(iy);
055         delta_y  = target_y - gun_y;
056
057         %% 3.1 No-wind solutions: scan angles, collect all valid (theta, v0)
058         candidates = [];  % each row: [theta_deg, v0]
059
060         for theta = theta_scan
061             theta_rad = theta * pi/180;
062             tan_theta = tan(theta_rad);
063             cos_theta = cos(theta_rad);
064
```

```
065             % analytic estimate for initial guess (vacuum trajectory)
066             if abs(cos_theta) < 1e-3
067                 v0_guess = v0_max - 10;
068             else
069                 if abs(delta_y) < 1e-6 && theta > 0.5
070                     % approximately equal height: R = v0^2 * sin(2θ) / g
071                     denom = sin(2*theta_rad);
072                     if denom <= 1e-3
073                         v0_guess = 0.5*(v0_min + v0_max);
074                     else
075                         v0_guess = sqrt(g * target_x / denom);
076                     end
077                 else
078                     % unequal height: v0^2 = g R^2 / [2 cos^2θ (R tanθ - Δy)]
079                     denom = target_x * tan_theta - delta_y;
080                     if denom <= 1e-3
081                         v0_guess = 0.5*(v0_min + v0_max);
082                     else
083                         v0_guess = sqrt( g * target_x^2 / (2 * cos_theta^2 * denom) );
084                     end
085                 end
086             end
087
088             v0_guess = max(v0_min, min(v0_guess, v0_max));
089
090             % fixed theta, U = 0, solve v0 that hits (target_x, target_y)
091             fun = @(v0) projectile_solver_fixed_theta_wind_CdM( ...
092                 v0, theta, 0, m, S, g, rho0, H, ...
093                 gun_y, target_x, target_y, v0_min, v0_max);
094
095             [v0_sol, fval, exitflag] = fsolve(fun, v0_guess, options);
096             total_dev = norm(fval);
097
098             if exitflag > 0 && v0_sol >= v0_min && v0_sol <= v0_max ...
099                             && total_dev <= dev_threshold
100                 candidates = [candidates; theta, v0_sol];
101             end
102         end
103
104         % no feasible solution within v0 <= 450 m/s
105         if isempty(candidates)
106             continue;
107         end
108
```

```
        %% 3.2 Choose minimal-angle branch as base solution
        [theta_best, idx_best] = min(candidates(:,1));
        v0_best                = candidates(idx_best, 2);

        theta_table(iy, ix) = theta_best;
        v0_table(iy, ix)    = v0_best;

        %% 3.3 First-order wind correction k(x,y)
        % v0(U) ≈ v0^(0) + k U, solved at fixed theta = theta_best
        k_wind = NaN;

        for U0 = U0_candidates
            U_neg = -U0;
            U_pos =  U0;

            % negative wind
            fun_neg = @(v0) projectile_solver_fixed_theta_wind_CdM( ...
                v0, theta_best, U_neg, m, S, g, rho0, H, ...
                gun_y, target_x, target_y, v0_min, v0_max);

            [v0_neg, fval_neg, exit_neg] = fsolve(fun_neg, v0_best, options);
            success_neg = (exit_neg > 0) && (norm(fval_neg) <= dev_threshold) ...
                          && v0_neg >= v0_min && v0_neg <= v0_max;

            % positive wind
            fun_pos = @(v0) projectile_solver_fixed_theta_wind_CdM( ...
                v0, theta_best, U_pos, m, S, g, rho0, H, ...
                gun_y, target_x, target_y, v0_min, v0_max);

            [v0_pos, fval_pos, exit_pos] = fsolve(fun_pos, v0_best, options);
            success_pos = (exit_pos > 0) && (norm(fval_pos) <= dev_threshold) ...
                          && v0_pos >= v0_min && v0_pos <= v0_max;

            if success_neg && success_pos
                % central difference
                k_wind = (v0_pos - v0_neg) / (U_pos - U_neg);
                break;
            elseif success_pos || success_neg
                % one-sided difference
                if success_pos
                    k_wind = (v0_pos - v0_best) / U_pos;
                else
                    k_wind = (v0_best - v0_neg) / (-U_neg);
                end
```

```
                break;
            end
        end

        kwind_table(iy, ix) = k_wind;   % remains NaN if all attempts failed
    end
end

fprintf('Firing table computation finished.\n');

%% 4. Visualisation: angle / base speed / wind coefficient

figure;
imagesc(x_list, y_list, theta_table);
set(gca,'YDir','normal'); colorbar;
xlabel('Range x (m)');
ylabel('Target altitude y (m)');
title('Recommended launch angle \theta^* (deg, minimal-angle branch)');

figure;
imagesc(x_list, y_list, v0_table);
set(gca,'YDir','normal'); colorbar;
xlabel('Range x (m)');
ylabel('Target altitude y (m)');
title('Required launch speed v_0^{(0)} (m/s, no wind, v_0 \le 450)');

figure;
imagesc(x_list, y_list, kwind_table);
set(gca,'YDir','normal'); colorbar;
xlabel('Range x (m)');
ylabel('Target altitude y (m)');
title('Wind correction coefficient k (m/s per m/s)');

%% 5. Export CSV tables

x_labels = strcat("x_", string(x_list), "m");   % column names: range
y_labels = strcat("h_", string(y_list), "m")';  % row names: altitude

v0_table_round    = round(v0_table,    2);
kwind_table_round = round(kwind_table, 2);

theta_tbl = array2table(theta_table, ...
    'VariableNames', cellstr(x_labels), ...
    'RowNames',      cellstr(y_labels));
```

```
writetable(theta_tbl, 'table_theta_deg_MachCd.csv', 'WriteRowNames', true);

v0_tbl = array2table(v0_table_round, ...
    'VariableNames', cellstr(x_labels), ...
    'RowNames',      cellstr(y_labels));
writetable(v0_tbl, 'table_v0_nowind_ms_MachCd.csv', 'WriteRowNames', true);

kwind_tbl = array2table(kwind_table_round, ...
    'VariableNames', cellstr(x_labels), ...
    'RowNames',      cellstr(y_labels));
writetable(kwind_tbl, 'table_kwind_ms_per_ms_MachCd.csv', 'WriteRowNames', true);

%% 6. Local functions

function F = projectile_solver_fixed_theta_wind_CdM(v0, theta, U, ...
    m, S, g, rho0, H, gun_y, target_x, target_y, v0_min, v0_max)
% Residual [dx; dy] at the target for fixed theta and wind U.

    % velocity bounds (large penalty outside range)
    if v0 < v0_min || v0 > v0_max
        F = [1e6*(v0 - v0_min); 1e6*(v0 - v0_max)];
        return;
    end

    theta_rad = theta*pi/180;
    % state: [x; y; vx; vy]
    y0 = [0, gun_y, v0*cos(theta_rad), v0*sin(theta_rad)];
    t_span = [0, 200];

    % NOTE: initial condition must be y0, not y
    [~, Y] = ode45(@(t,Y) projectile_ode_wind_CdM(t, Y, U, m, S, g, rho0, H), ...
                   t_span, y0);

    % linear interpolation at x = target_x
    idx = find(Y(:,1) > target_x, 1);
    if isempty(idx)
        x_final = Y(end,1);
        y_final = Y(end,2);
    else
        x1 = Y(idx-1,1); y1 = Y(idx-1,2);
        x2 = Y(idx,1);   y2 = Y(idx,2);
        y_final = y1 + (target_x - x1) * (y2 - y1)/(x2 - x1);
        x_final = target_x;
    end
```

```
241
242     F = [x_final - target_x; y_final - target_y];
243 end
244
245
246 function dy = projectile_ode_wind_CdM(~, y, U, m, S, g, rho0, H)
247 % 2D trajectory with horizontal wind U and Mach-dependent Cd.
248
249     % state: y = [x; y; vx; vy]
250     vx = y(3);
251     vy = y(4);
252
253     % velocity relative to air (wind along +x)
254     ux = vx - U;
255     uy = vy;
256     v_rel = sqrt(ux^2 + uy^2) + 1e-8;   % avoid division by zero
257
258     % Mach number and Cd(M)
259     a   = 340;               % speed of sound (m/s)
260     M   = v_rel / a;
261     CdM = Cd_of_Mach(M);
262
263     % altitude-dependent density
264     rho = rho0 * exp(-y(2)/H);
265
266     % quadratic drag acceleration
267     k  = 0.5 * rho * S * CdM * v_rel / m;
268     ax = -k * ux;
269     ay = -g - k * uy;
270
271     dy = [vx; vy; ax; ay];
272 end
273
274 function Cd = Cd_of_Mach(M)
275 % Empirical piecewise drag coefficient Cd(M).
276
277     Cd = zeros(size(M));
278
279     % M < 0.7
280     idx1 = (M < 0.7);
281     Cd(idx1) = 0.45;
282
283     % 0.7 <= M < 1.1 : 0.45 -> 1.0
284     idx2 = (M >= 0.7) & (M < 1.1);
```

```
    Cd(idx2) = 0.45 + 0.55 * (M(idx2) - 0.7) / (1.1 - 0.7);

    % 1.1 <= M < 1.5 : 1.0 -> 0.8
    idx3 = (M >= 1.1) & (M < 1.5);
    Cd(idx3) = 1.0 - 0.2 * (M(idx3) - 1.1) / (1.5 - 1.1);

    % M >= 1.5
    idx4 = (M >= 1.5);
    Cd(idx4) = 0.8;
end
```