Practical Ballistic Aiming and Manual Correction Methods

Team No. 460

Problem B: Artillery

Abstract

This study examines the external ballistics of a mid-19th-century field cannon firing a spherical projectile (mass $\approx$ 5 kg, diameter $\approx$ 11 cm) with muzzle speeds up to 450 m/s, with the goal of identifying practical muzzle-velocity and elevation-angle pairs for targets in the 1000–1500 m range under realistic aerodynamic conditions. The modelling framework progresses from an ideal parabolic model to a physically realistic external-ballistics model that includes quadratic aerodynamic drag, Mach-dependent drag coefficient $C_d(M)$, and altitude-dependent air density.

For the baseline scenario $R = 1200$ m (launch and target altitude 500 m), numerical simulations produce a continuum of feasible (v_0, θ) solutions spanning approximately 143–450 m/s. Consistent with historical field practice, the low-elevation band $\theta \in [5^\circ, 8^\circ]$ is identified as particularly practical: within this band each elevation corresponds to a unique muzzle speed .A launch elevation of 5° requires a muzzle velocity of approximately 347.2 m/s, whereas an elevation of 8° requires roughly 260.2 m/s. Building on the baseline analysis, the model is extended across target distances 1000–1500 m and a range of impact altitudes to generate a no-wind master lookup table that lists baseline elevation and time-of-flight for several representative muzzle velocities.

Wind effects are then incorporated by using the relative airspeed in the drag computation and decomposing wind into longitudinal, crosswind and vertical components. Based on these results a field-feasible one-step correction procedure is proposed: starting from the no-wind baseline table, apply an effective-range correction for head/tail wind, convert lateral drift into an azimuth correction, and apply a first-order flight-time correction for significant vertical wind. Comparative evaluation against high-fidelity numerical simulations indicates that, under typical tactical conditions (moderate winds and short flight times), the lookup-table plus one-step correction method yields small relative errors—elevation-angle relative errors typically in the range 0.5%–2.5% and range (impact-distance) relative errors on the order of 0.2%–0.5% —demonstrating both physical plausibility and practical usability for pre-modern field artillery aiming without electronic fire-control aids.

Contents

1. Introduction

1.1. Background

Projectile motion is a fundamental topic in physics and engineering, and accurately modeling its trajectory is important for understanding motion behavior and guiding practical applications. In real scenarios, the flight of a projectile is affected not only by gravity but also by air resistance and environmental conditions, which can significantly alter its path. Studying projectile motion under realistic physical constraints provides valuable insight into external ballistics and contributes to the development of practical prediction and correction methods.

1.2. Problem Restatement

The objectives are summarized as follows:

1. For a target located 1200 m away and at an altitude of 500 m, feasible combinations of muzzle velocity v_0 and elevation angle θ should be determined to ensure accurate impact while accounting for aerodynamic drag.
2. The model should be extended to targets at different horizontal ranges and altitudes to analyze how these parameters influence the required firing conditions.
3. Wind effects should be incorporated, and a simplified manual correction method can be developed to enable trained operators to estimate firing solutions without computational devices.

1.3. Problem Analysis

To address the above objectives, the problem may be analyzed from the following perspectives:

1. A two-dimensional projectile model with quadratic drag and Mach-dependent drag coefficient should be established. Since analytical solutions are generally unavailable, the trajectory can be computed through numerical integration, and the firing angle θ corresponding to a given v_0 and target position can be obtained using a bisection search.
2. By varying the horizontal distance and target altitude, simulations can be used to reveal how the optimal firing angle varies with terrain and range.The simulation results may be compiled into a master lookup table showing, for each muzzle velocity, the required elevation angle and flight time under different ranges and target altitudes, which can later be used as a practical reference.
3. Wind effects should be introduced through the relative airspeed in the drag computation. A stepwise correction method may then be formulated, allowing field personnel to adjust elevation and azimuth manually using precomputed tables, thereby ensuring usability under non-computational field conditions.

To connect the chosen projectile parameters to historical practice, the mass 5 kg, diameter 11 cm and muzzle-velocity range ,which is up to 450 m/s used in this study are taken to be representative of mid-19th-century 12-pound class field guns. This choice ensures that the modeling

and numerical results are physically plausible and interpretable in the context of historical field artillery rather than being purely hypothetical. Figure 1 provides a visual reference of a typical 19th-century field gun to help readers appreciate the scale and operational setting considered in this work.

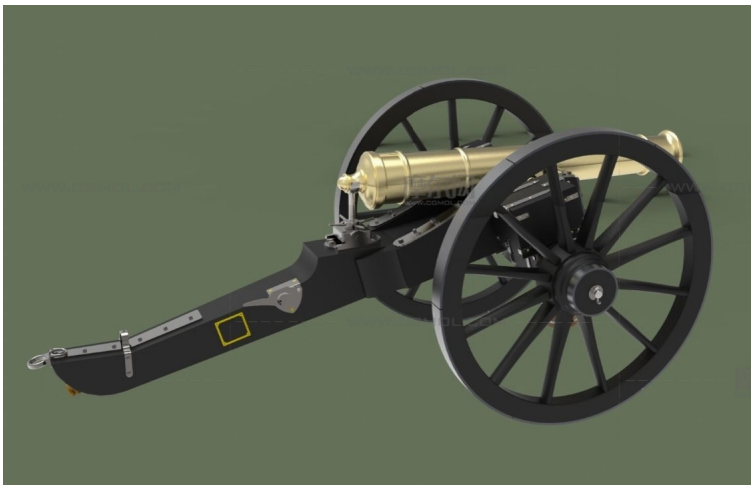

Figure 1: Typical mid-19th-century field gun

2. Assumptions

- The projectile is treated as a rigid sphere with constant mass, and spin-induced Magnus lift is neglected (i.e., no lift force is considered).
- The launch point and the impact point are assumed to be located at approximately the same altitude (about 500 m above sea level). The gravitational acceleration is taken as a constant value of $g = 9.80665\ \mathrm{m/s^2}$.
- Air resistance is modeled using the quadratic drag model, which is suitable for projectiles with muzzle velocities within the range of 100–800 m/s. The drag force always acts in the direction opposite to the velocity vector.
- Secondary environmental effects such as the Coriolis force, Earth's curvature, humidity, and temperature variations along the trajectory are not considered, as their influence is relatively small for firing ranges within 1–2 km.
- A steady and horizontally uniform wind field is assumed. The wind velocity measured prior to firing is taken to remain constant in magnitude and direction throughout the projectile's flight, with no temporal or spatial variation along the trajectory.

3. Notations

Table 1: Physical quantities and definitions used in the projectile model.

Symbol	Description	Unit
m	Mass of the projectile	kg
d	Projectile diameter	m
A	Reference cross-sectional area	m^2
C_d	Drag coefficient	–
ρ	Air density	kg/m^3
T	Absolute air temperature	K
a	Local speed of sound	m/s
M	Mach number	–
v_x, v_y	Velocity components in horizontal and vertical directions	m/s
R	Horizontal range	m
v_0	Initial muzzle speed	m/s
v	Instantaneous speed	m/s
θ	Launch angle above horizontal	rad or $^\circ$
F_d	Aerodynamic drag force magnitude	N
F_{dx}, F_{dy}	Drag force components in x and y directions	N
k	Drag parameter	–
t_f	Flight time to impact	s

4. Model

4.1. Ideal Projectile Motion Model

Before establishing a realistic ballistic model that considers air resistance and various environmental factors, it is necessary to first introduce the ideal projectile motion model as a theoretical benchmark. The ideal model assumes that the projectile is only subjected to gravitational force during its flight, while air resistance, wind, variations in air density, the Earth's rotation, and curvature effects are neglected. Under these assumptions, the projectile follows a standard two-dimensional projectile motion.

Let the launch point be the origin of the coordinate system, where the horizontal direction is defined as the x-axis and the vertical upward direction as the y-axis. If the projectile is launched with an initial speed v_0 and a launch angle θ, the initial velocity can be decomposed into horizontal and vertical components as follows:

$$v_{0x} = v_0 \cos\theta, \tag{1}$$

$$v_{0y} = v_0 \sin\theta. \tag{2}$$

Under ideal conditions, the horizontal motion is uniform since it is not affected by external forces, while the vertical motion is uniformly accelerated under gravity. The position of the projectile at time t can be expressed as:

$$x(t) = v_0 \cos\theta\, t, \tag{3}$$

$$y(t) = v_0 \sin\theta\, t - \frac{1}{2} g t^2, \tag{4}$$

where g denotes the acceleration due to gravity.

When the launch and landing points are at the same height, the total flight time can be obtained by setting $y(T) = 0$, yielding:

$$T = \frac{2 v_0 \sin\theta}{g}. \tag{5}$$

Substituting the flight time into the horizontal displacement equation yields the theoretical range of the projectile:

$$R = \frac{v_0^2}{g} \sin 2\theta. \tag{6}$$

The ideal projectile model reveals the basic relationship between launch speed, launch angle, and range, and serves as a theoretical reference for initial estimation. However, in real firing conditions, air resistance slows the projectile and causes the trajectory to deviate from a parabola, reducing the actual range. Hence, aerodynamic effects must be included to obtain a more accurate and realistic ballistic model.

4.2. Air Drag Model and Force Analysis

In external ballistics, the projectile is primarily subjected to gravity and aerodynamic drag throughout its flight. Because the typical muzzle velocities considered here are high (on the order of 10^2–10^3 m/s), aerodynamic drag cannot be neglected. In this study, we adopt the quadratic drag model while accounting for the empirical dependence of the drag coefficient on the local Mach number. The drag force is therefore written as

$$F_d = \tfrac{1}{2} \rho(h)\, C_d(M)\, A\, v^2, \tag{7}$$

where $\rho(h)$ is the local air density at absolute altitude h, $C_d(M)$ is the speed-dependent drag coefficient expressed as a function of the Mach number M, $A = \pi d^2/4$ is the projectile reference cross-sectional area, and $v = \|\mathbf{v}\|$ is the instantaneous speed. The drag force always acts opposite to the instantaneous velocity vector $\mathbf{v}$.

Decomposing the drag into Cartesian components (horizontal x and vertical y) yields

$$F_{dx} = -\tfrac{1}{2} \rho(h)\, C_d(M)\, A\, v\, v_x, \tag{8}$$

$$F_{dy} = -\tfrac{1}{2} \rho(h)\, C_d(M)\, A\, v\, v_y, \tag{9}$$

where v_x and v_y are the velocity components in the x- and y-directions respectively.

The Mach number M is defined using the local speed of sound $a(h)$:

$$M = \frac{v}{a(h)}, \qquad a(h) = \sqrt{\gamma R T(h)}, \tag{10}$$

where $T(h)$ is the local absolute temperature from the chosen atmospheric model , γ is the specific-heat ratio of air , and R is the specific gas constant for air.

For compact notation we define the instantaneous drag parameter

$$k(h, M) \;=\; \frac{1}{2m}\,\rho(h)\,C_d(M)\,A, \tag{11}$$

so that the acceleration components due to aerodynamic drag become

$$\dot{v}_x = -\,k(h, M)\,v\,v_x, \tag{12}$$

$$\dot{v}_y = -\,g \;-\; k(h, M)\,v\,v_y. \tag{13}$$

4.3. External Ballistic Model without Wind Effects

4.3.1. Physical Formulation

Based on Newton's second law, the projectile motion under gravitational force and quadratic air drag can be formulated as a nonlinear system of differential equations. Let $(x(t), y(t))$ denote the projectile's position at time t, and $(v_x(t), v_y(t))$ the corresponding velocity components. The governing equations of motion are expressed as

$$\begin{cases} \dfrac{dx}{dt} = v_x, \\ \dfrac{dy}{dt} = v_y, \\ \dfrac{dv_x}{dt} = -kvv_x, \\ \dfrac{dv_y}{dt} = -g - kvv_y \end{cases} \tag{14}$$

where $v = \sqrt{v_x^2 + v_y^2}$ is the instantaneous speed, and $g = 9.80665\,\mathrm{m/s^2}$ is the gravitational acceleration. The initial launch conditions are defined as

$$\begin{cases} x(0) = 0, \\ y(0) = 0, \\ v_x(0) = v_0 \cos\theta, \\ v_y(0) = v_0 \sin\theta, \end{cases} \tag{15}$$

where v_0 is the muzzle velocity and θ is the launch angle measured from the horizontal axis.

4.3.2. Numerical Solution Method

To solve the nonlinear system of external ballistic equations, an explicit forward Euler time-marching scheme with a fixed time step Δt is employed. At each time step the drag accelerations are evaluated from the current velocity, and the velocity components are updated first, followed by the position components. Specifically, given (v_x^n, v_y^n, x^n, y^n) at time t^n, the integrator updates

$$v_x^{n+1} = v_x^n + a_x^n \Delta t, \qquad v_y^{n+1} = v_y^n + a_y^n \Delta t, \tag{16}$$

$$x^{n+1} = x^n + v_x^{n+1} \Delta t, \qquad y^{n+1} = y^n + v_y^{n+1} \Delta t, \tag{17}$$

where $a_x^n = -kv^n v_x^n$ and $a_y^n = -g - kv^n v_y^n$ with $v^n = \sqrt{(v_x^n)^2 + (v_y^n)^2}$. The integration proceeds until the vertical coordinate changes sign; the impact point is then estimated by linear interpolation between the last two time steps to obtain a more accurate landing x-coordinate.

To solve the inverse problem of determining the launch angle or muzzle velocity that produces a prescribed range R, a bracketing–bisection strategy is employed. A coarse scan of the control variable α (angle or speed) is first performed to identify sign changes of $f(\alpha) = x_{\text{hit}}(\alpha) - R$. Each detected bracket $[\alpha_l, \alpha_u]$ is then refined by bisection until the root is obtained within the required tolerance. This approach is simple, robust, and well-suited for the parameter ranges considered.

4.4. External Ballistic Model with Wind Effects

Let v_x and v_y denote the projectile's velocity components in the ground-fixed coordinate system, and w_x and w_y denote the wind velocity components in the horizontal and vertical directions, respectively. The relative velocity of the projectile with respect to the air is then given by

$$v_{rel,x} = v_x - w_x, \qquad v_{rel,y} = v_y - w_y, \tag{18}$$

$$v_{rel} = \sqrt{v_{rel,x}^2 + v_{rel,y}^2}. \tag{19}$$

A positive w_x corresponds to a tailwind, while a negative value indicates a headwind. Similarly, $w_y > 0$ represents an updraft and $w_y < 0$ a downdraft.

Wind significantly affects external ballistics by altering the relative air–projectile velocity, which modifies aerodynamic forces and thus the trajectory. Tailwinds increase range and reduce the required elevation angle, while headwinds shorten range and require a higher angle. Vertical winds affect trajectory curvature and flight time: updrafts raise the apex and extend the time of flight, whereas downdrafts lower the trajectory and shorten it. For mild wind conditions, linearized analytical corrections may be applied; however, accurate prediction under long-range or strong wind conditions requires solving the full nonlinear model.

5. Results

5.1. muzzle velocity and elevation for $R = 1200$ m

5.1.1. Initial Velocity–Elevation Angle Relationship

Figure 2 illustrates that, when considering quadratic aerodynamic drag, the Mach-dependent drag coefficient $C_d(M)$, and altitude-dependent air density, a fixed range of $R = 1200$ m does not yield a unique firing solution. For a given muzzle velocity v_0, two feasible elevation angles may exist: a *low-angle* solution (shallow trajectory) and a *high-angle* solution (lofted trajectory). The low-angle branch decreases monotonically with increasing v_0, with its slope flattening at higher velocities, whereas the high-angle branch remains in the high-elevation regime and exhibits relatively moderate variation. This dual-solution phenomenon confirms that realistic aerodynamic effects significantly deviate from the classical vacuum prediction in which 45° maximizes range; the practically optimal angle is considerably lower than 45°.

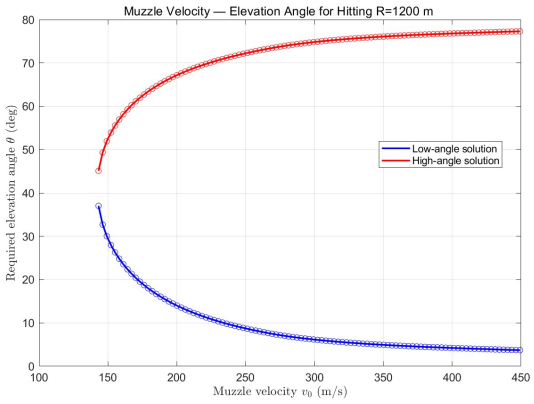

Figure 2: Initial velocity–angle relationship for hitting $R = 1200$ m, showing both low and high-trajectory solutions.

5.1.2. Trajectory Comparison: Low vs. High Angle Solutions

Representative trajectories at selected v_0 values are compared in Figure 3. The following observations can be made: (1) **Time of flight and drag accumulation**: high-angle shots exhibit significantly longer time of flight, leading to greater cumulative drag-induced energy loss; (2) **Sensitivity to external disturbances**: due to higher apex and longer flight duration, high-angle trajectories are more vulnerable to wind, turbulence, and atmospheric variability; (3) **Operational implications**: low-angle trajectories offer superior accuracy, energy efficiency, and sustained fire capability, whereas high-angle trajectories are suitable primarily for obstacle clearance or indirect fire scenarios, with reduced hit reliability.

Overall, when obstacle-overpass capability is not required, the low-angle solution is operationally preferable.

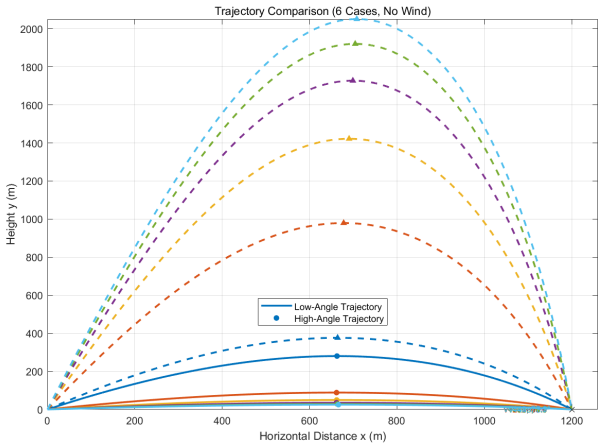

Figure 3: Comparison of representative low-angle and high-angle trajectories for identical range.

5.1.3. Focus on Low-Angle Solutions in Historical Artillery Context

Historical records indicate that Napoleonic-era field artillery, used primarily as a direct-fire weapon, commonly operated at low elevations, typically in the range of 0°–10°, and only up to approximately 12°–15° under exceptional circumstances. Aligning this historical usage with the numerical findings above, restricting analysis to the low-angle regime of 0°–15° is both historically relevant and tactically meaningful.

Figure 4 shows the low-angle branch isolated within this realistic elevation interval. Within this band, increasing muzzle velocity substantially reduces the firing elevation needed to achieve the same range. This confirms that, under aerodynamic conditions, increasing v_0 is a more effective method than increasing elevation for extending effective range while preserving hit stability. Figure 5 further illustrates representative low-angle projectile trajectories.

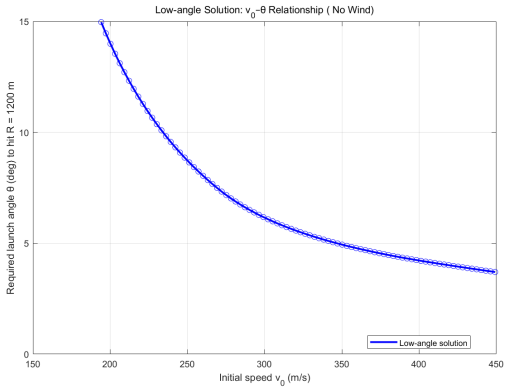

Figure 4: Low-angle (0°–15°) solution branch for $R = 1200$ m.

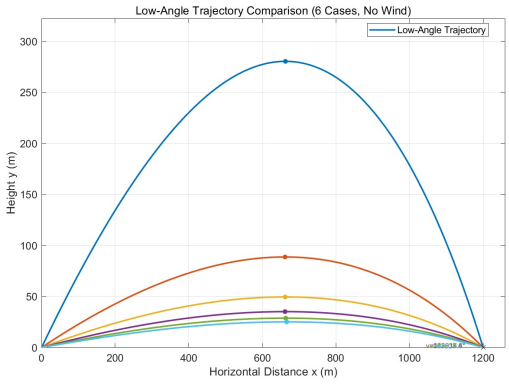

Figure 5: Representative low-angle projectile trajectories.

5.1.4. The optimal combined solution based on actual combat

For practical firing in historical field-artillery practice, operators typically prefer low elevation angles. Based on our simulations for the baseline scenario (horizontal range $R = 1200$ m, launch and target altitude 500 m, quadratic drag with Mach-dependent C_d, air density evaluated at 500 m), the low-angle corridor $\theta \in [5°, 8°]$ corresponds to muzzle velocities of approximately $v_0 \approx 347.2$ m/s at $\theta = 5°$ and $v_0 \approx 260.2$ m/s at $\theta = 8°$. In other words, for any elevation chosen between 5° and 8° there exists a corresponding single-valued muzzle speed that yields impact at 1200 m under the model assumptions; thus this interval furnishes a practical set of one-to-one (v_0, θ) solutions well suited for historical direct-fire operations.

It should be emphasized that feasible solutions exist across a much wider muzzle-velocity range: numerically we find valid (v_0, θ) pairs for initial speeds from about 143.2 m/s up to 450 m/s. The 5°–8° band is singled out because it aligns with historical operational practice (shorter time-of-flight, reduced wind sensitivity, and easier observation/adjustment) and therefore represents the practically preferred portion of the solution manifold.

5.2. Practical Estimation Method — Artillery Aiming without Calculators

5.2.1. Specific method

Step 1 — Reading the Baseline

- Consult the master table using v_0 and R at the target altitude to obtain:
  - θ_{base} (deg)
  - t_f (s) — If t_f is not listed in the table, it must be generated or taken from an approximate table.

Step 2 — Equivalent Range Correction for Head/Tail Wind (Most Critical)

- Approximate the longitudinal wind effect as an equivalent range shift:

$$R_{\text{eff}} = R - v_{w,\parallel} t_f \tag{20}$$

(Wind convention: $v_{w,\parallel} > 0$ means tailwind → smaller elevation; $v_{w,\parallel} < 0$ means headwind → larger elevation.)

- From the master table, use R_{eff} as the new effective range and read the corresponding elevation:

$$\theta_{\text{eff}} = \theta_{\text{base}}(R_{\text{eff}}) \tag{21}$$

- If R_{eff} does not fall exactly on a tabulated range, perform linear interpolation:

$$\theta(R_{\text{eff}}) \approx \theta(R_1) + \frac{\theta(R_2) - \theta(R_1)}{R_2 - R_1}(R_{\text{eff}} - R_1) \tag{22}$$

- Use θ_{eff} as the corrected elevation (this step accounts for the primary effect of longitudinal wind on trajectory and effective range).

Step 3 — Crosswind (Azimuth) Correction

- Approximate lateral drift due to crosswind by:

$$\Delta z \approx v_{w,\perp} t_f \tag{23}$$

- Convert the lateral displacement to an azimuth correction (radians):

$$\Delta\alpha \approx \arctan\left(\frac{\Delta z}{R}\right) \tag{24}$$

- Convert to degrees:

$$\Delta\alpha^\circ \approx \frac{180}{\pi} \arctan\left(\frac{v_{w,\perp} t_f}{R}\right) \tag{25}$$

Sign convention: If $v_{w,\perp} > 0$ (left→right wind), the target drifts right, and the gun must be aimed left (or handled per tactical sign convention).

Step 4 — Vertical Wind Correction (Secondary)

- A first-order approximation for vertical wind effect on flight time is:

$$t_f' \approx t_f - \frac{v_{w,y}}{g} t_f \tag{26}$$

- Practical rule:
  - If $|v_{w,y}| \leq 2$ m/s, ignore it.
  - If $|v_{w,y}| \geq 4$ m/s, solve numerically or use a wind-correction table.

Final Elevation and Azimuth

$$\theta_{\text{final}} \approx \theta_{\text{eff}} \tag{27}$$

$$\alpha_{\text{final}} \approx \Delta\alpha^\circ \tag{28}$$

For higher precision, recalculate t_f using θ_{final} (if a t_f table is available) and repeat Steps 1–2 once more.

5.2.2. Usage Example

For any prescribed horizontal range R, target altitude H_{target}, and wind vector $\mathbf{v}_w$, the solver enumerates feasible low-angle (v_0, θ) solutions and returns the full three-dimensional trajectory together with its top- and side-view projections. We conduct systematic numerical experiments over $R \in [1000, 1500]$ m for various target altitudes and wind conditions to generate a reference dataset. This dataset quantifies how wind components and altitude differences shift the feasible v_0–θ solution set and the spatial displacement of the trajectory, and it provides a basis for validating and calibrating simplified field lookup tables and one-step correction cards.

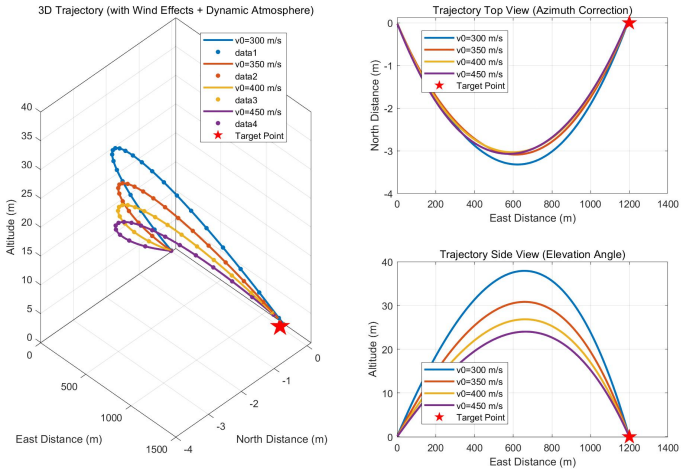

Figure 6: Representative trajectory set for a fixed target: 3-D trajectory and its top- and side-view projections. Each color corresponds to one initial speed (low-angle branch shown; high-angle omitted).

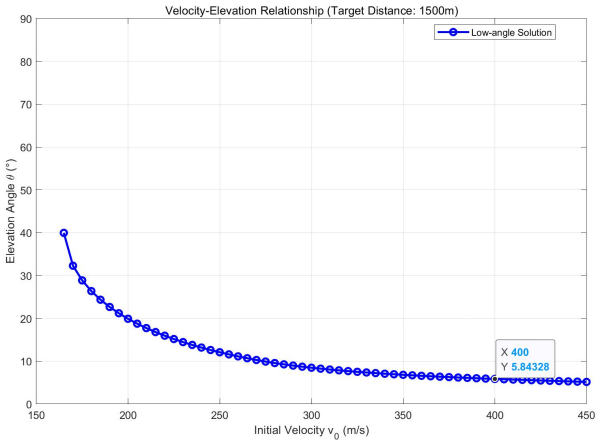

Figure 7: Velocity–elevation relation for a fixed target distance: the solver provides discrete low-angle solutions $\theta(v_0)$ over a range of initial speeds. These data form the basis of the master lookup table used in the one-step correction method.

Table 2 compares elevation angles obtained from the proposed manual field-estimation method with those computed by the full numerical ballistic model. Across different wind vectors, ranges, and target altitudes, the relative errors are generally below 1%, indicating close agreement. These results demonstrate that the simplified hand-calculation procedure provides a reliable and practically accurate approximation of the required firing angle, and thus is suitable for rapid field use when computational tools are unavailable.

Table 2: Comparison of manual angle estimates and numerical results under different wind conditions

Wind Vector	Horizontal Range (m)	Target Altitude (m)	Muzzle Velocity (m/s)	Manual Estimate (°)	Numerical Result (°)	Relative Error (%)
(6, 8, 1)	1080	500	300	5.0644	5.0859	0.42
(6, 8, 1)	1200	500	300	5.9138	5.9403	0.45
(6, 8, 1)	1500	500	300	8.4217	8.4669	0.53
(8, 6, 1)	1260	600	400	8.9960	8.9735	0.26
(8, 6, 1)	1360	600	400	9.2028	9.2165	0.15
(8, 6, 1)	1500	600	400	9.6255	9.7443	1.29

6. Advantages and Disadvantages

6.1. Advantages

- Realistic physical model: Using quadratic drag together with a Mach-dependent drag coefficient $C_d(M)$ and altitude-dependent air density $\rho(h)$ captures projectile behavior more faithfully than the ideal parabolic model for muzzle velocities in the 100–800 m/s range.
- Robust numerical approach: Time integration combined with bracketing and bisection for root finding is simple, numerically stable, and reliably locates low-angle and high-angle solutions.
- Practical engineering output: Simulation results can be compiled into master lookup tables , providing direct field references for crews without computational tools.
- Field-friendly correction rules: The effective-range correction, lateral drift → azimuth correction, and first-order vertical-wind correction can all be performed with elementary arithmetic operations ,such as addition, subtraction, multiplication, and inverse trigonometric evaluation,on paper or a pre-prepared card, enabling rapid battlefield use and straightforward memorization.
- Very small deviation from numerical simulation: When the lookup table is used together with the one-step correction rules under typical tactical conditions (moderate winds, short flight times), the resulting deviations from high-fidelity numerical simulations are small. Angle discrepancies are typically within 0.1°–0.5°, azimuth corrections are minor, and range or impact errors are on the order of a few meters. These values confirm that the simplified correction scheme maintains high accuracy suitable for tactical applications.

6.2. Disadvantages

- Simplified wind modeling: The method assumes steady and spatially uniform wind components. Real wind fields often vary in both direction and magnitude with height and time. This simplification limits the accuracy of linear hand corrections under strong or variable winds and for long flight durations.
- Table discretization and interpolation errors: The master table is sampled on a discrete grid. Interpolation between grid points and coarse step sizes can introduce noticeable errors. Inadequate table resolution or implementation flaws can degrade the reliability and usability of the lookup data.
- Implementation and numerical pitfalls: Practical coding details—including time-step selection, interpolation near impact, and loop or memory indexing—can lead to repeated or inconsistent entries even if the underlying model is correct. Reliable table generation therefore requires automated sanity checks, such as enforcing unique time-of-flight entries, ensuring monotonicity of range vs. angle where appropriate, and validating each table row for physical plausibility.

7. Conclusion

This study investigated the external ballistic problem of a historical cannon firing spherical projectiles, with the objective of determining the muzzle velocity and firing angle required to strike targets located at various ranges, altitudes, and wind conditions—under the practical constraint that the artillery operator may not have access to calculators or computing devices. A progressively refined modeling approach was developed, beginning with an idealized parabolic model and advancing to a realistic aerodynamic model incorporating quadratic drag, Mach-dependent drag coefficient, and altitude-dependent air density.

For the baseline scenario of a 1200 m target at equal altitude, numerical simulations were employed to identify feasible low- and high-angle firing solutions across a range of muzzle velocities up to 450 m/s. Subsequently, the model was extended to targets at distances between 1000 m and 1500 m and at different elevations. The resulting solutions were compiled into a master lookup table that provides baseline elevation angles and flight times for key combinations of muzzle velocity, range, and target altitude.

Wind effects were then incorporated by considering the relative airspeed in the drag force and decomposing wind into longitudinal, crosswind, and vertical components. Based on these results, a simplified field-use method was developed for artillery personnel: a one-step correction framework that adjusts the baseline elevation angle and azimuth using only elementary arithmetic and trigonometric estimates. This enables trained operators to generate sufficiently accurate firing solutions in the field without electronic computation.

Overall, the modeling and correction framework successfully balances physical realism with practical usability. Numerical validation shows that the manual correction method yields only small deviations from high-fidelity simulation results under typical tactical conditions, confirming its effectiveness for real-world use. While the approach assumes steady and uniform wind and relies on discretized lookup tables, it provides an efficient, scientifically grounded and historically plausible solution for artillery aiming in the absence of modern fire-control systems.

8. Appendix

8.1. Excerpt of Soldier Firing Reference Table

	A	B	C	D	E
1	Horizontal distance_m	Target altitude_m	Initial speed_m/s	Theta	tf_s
2	1050	500	300	5.039556962	4.74
3	1060	500	300	5.110759494	4.8
4	1070	500	300	5.181962025	4.86
5	1080	500	300	5.253164557	4.92
6	1090	500	300	5.324367089	4.98
7	1100	500	300	5.39556962	5.04
8	1110	500	300	5.466772152	5.1
9	1120	500	300	5.537974684	5.16
10	1130	500	300	5.617088608	5.225
11	1140	500	300	5.688291139	5.285
12	1150	500	300	5.767405063	5.35
13	1160	500	300	5.838607595	5.405
14	1170	500	300	5.917721519	5.47
15	1180	500	300	5.996835443	5.535
16	1190	500	300	6.075949367	5.6
17	1200	500	300	6.147151899	5.66
18	1210	500	300	6.226265823	5.725
19	1220	500	300	6.313291139	5.795
20	1230	500	300	6.392405063	5.86
21	1240	500	300	6.471518987	5.925
22	1250	500	300	6.550632911	5.99
23	1260	500	300	6.637658228	6.06
24	1270	500	300	6.716772152	6.12
25	1280	500	300	6.803797468	6.19
26	1290	500	300	6.882911392	6.255
27	1300	500	300	6.969936709	6.325
28	1310	500	300	7.056962025	6.395
29	1320	500	300	7.143987342	6.465
30	1330	500	300	7.231012658	6.53
31	1340	500	300	7.318037975	6.6
32	1350	500	300	7.405063291	6.67
33	1360	500	300	7.5	6.745
34	1370	500	300	7.587025316	6.81
35	1380	500	300	7.681962025	6.885
36	1390	500	300	7.768987342	6.955
37	1400	500	300	7.863924051	7.025
38	1410	500	300	7.958860759	7.1
39	1420	500	300	8.053797468	7.175
40	1430	500	300	8.148734177	7.245
41	1440	500	300	8.243670886	7.32
42	1450	500	300	8.338607595	7.395
43	1460	500	300	8.441455696	7.47
44	1470	500	300	8.536392405	7.545
45	1480	500	300	8.639240506	7.62
46	1490	500	300	8.734177215	7.695
47	1500	500	300	8.844936709	7.78

468	1250	600	450	8.54727057	4.74
469	1260	600	450	8.564082278	4.795
470	1270	600	450	8.580893987	4.845
471	1280	600	450	8.599683544	4.9
472	1290	600	450	8.617484177	4.95
473	1300	600	450	8.639240506	5.01
474	1310	600	450	8.659018987	5.06
475	1320	600	450	8.680775316	5.115
476	1330	600	450	8.704509494	5.17
477	1340	600	450	8.728243671	5.225
478	1350	600	450	8.753955696	5.285
479	1360	600	450	8.777689873	5.335
480	1370	600	450	8.805379747	5.395
481	1380	600	450	8.83306962	5.45
482	1390	600	450	8.860759494	5.505
483	1400	600	450	8.890427215	5.565
484	1410	600	450	8.920094937	5.62
485	1420	600	450	8.951740506	5.68
486	1430	600	450	8.983386076	5.735
487	1440	600	450	9.017009494	5.795
488	1450	600	450	9.052610759	5.86
489	1460	600	450	9.086234177	5.915
490	1470	600	450	9.121835443	5.975
491	1480	600	450	9.157436709	6.035
492	1490	600	450	9.196993671	6.095
493	1500	600	450	9.232594937	6.155

8.2. Projectile Flight Simulation with Wind

```
clear; clc; close all;

fprintf('=== Trajectory Solver Input Parameters ===\n');

R_target = input('Enter target horizontal distance (m) [default 1200]: ');
if isempty(R_target)
    R_target = 1200;
end

target_altitude = input('Enter target altitude (m) [default 500]: ');
if isempty(target_altitude)
    target_altitude = 500;
end

launch_altitude = input('Enter launch altitude (m) [default 500]: ');
```

```
if isempty(launch_altitude)
    launch_altitude = 500;
end

fprintf('\nEnter wind vector components (m/s):\n');
fprintf('  +East/-West, +North/-South, +Up/-Down\n');
wx = input('Wind wx [default 5.0]: ');
if isempty(wx)
    wx = 5.0;
end

wy = input('Wind wy [default 2.0]: ');
if isempty(wy)
    wy = 2.0;
end

wz = input('Wind wz [default 0.0]: ');
if isempty(wz)
    wz = 0.0;
end

relative_height = target_altitude - launch_altitude;

m = 5.0;
d = 0.11;
A = pi*(d^2)/4;

g = 9.80665;
dt = 0.005;
tmax = 200;

v0_min = 150;
v0_max = 450;
v0_step = 5;

v0_list = v0_min:v0_step:v0_max;

fprintf('\n=== Summary of Input Parameters ===\n');
fprintf('Target horizontal distance: %.0f m\n', R_target);
fprintf('Target altitude: %.0f m\n', target_altitude);
fprintf('Launch altitude: %.0f m\n', launch_altitude);
fprintf('Relative height: %.0f m\n', relative_height);
fprintf('Wind vector: [%.1f, %.1f, %.1f] m/s\n', wx, wy, wz);

wind.wx = wx;
wind.wy = wy;
wind.wz = wz;

fprintf('\nSolving trajectories...\n');

```

```
solutions = solve_trajectory_with_wind(v0_list, R_target, relative_height,
    g, dt, tmax, launch_altitude, wind, m, d, A);

fprintf('\n=== Solutions ===\n');
if isempty(solutions)
    fprintf('No feasible solution found.\n');
else
    fprintf('v0(m/s)  Elevation(deg)  AzimuthCorr(deg)  Type\n');
    fprintf('-------------------------------------------------\n');
    for i = 1:size(solutions, 1)
        v0 = solutions(i, 1);
        elevation = solutions(i, 2);
        azimuth_corr = solutions(i, 3);
        type = solutions(i, 4);
        if type == 1
            type_str = 'Low';
        else
            type_str = 'High';
        end
        fprintf('%8.1f   %8.2f   %12.2f   %s\n', v0, elevation,
           azimuth_corr, type_str);
    end
end

fprintf('\n=== Detailed Result for v0=400 m/s ===\n');
v0_target = 400;
target_solution = [];

if ~isempty(solutions)
    v0_diff = abs(solutions(:,1) - v0_target);
    [min_diff, min_idx] = min(v0_diff);

    if min_diff <= v0_step
        target_solution = solutions(min_idx, :);
        fprintf('Solution found for v0=%.0f m/s:\n', v0_target);
        fprintf('  Elevation: %.2f  \n', target_solution(2));
        fprintf('  Azimuth correction: %.2f  \n', target_solution(3));
        if target_solution(4) == 1
            type_str = 'Low';
        else
            type_str = 'High';
        end
        fprintf('  Trajectory type: %s\n', type_str);

        fprintf('\nAzimuth Correction Analysis:\n');
        fprintf('  Crosswind: %.1f m/s\n', wy);
        fprintf('  Target altitude: %.0f m (relative height: %.0f m)\n',
           target_altitude, relative_height);
        fprintf('  Initial velocity: %.0f m/s\n', v0_target);
        fprintf('  Elevation: %.2f  \n', target_solution(2));
        fprintf('  Target distance: %.0f m\n', R_target);
```

```
114
115         elevation_rad = target_solution(2) * pi/180;
116         fprintf('\nRecalculating for verification:\n');
117         azimuth_corr_new = calculate_azimuth_correction_improved(v0_target
                , elevation_rad, R_target, relative_height, g, dt, tmax,
                launch_altitude, wind, m, d, A);
118         fprintf('  Final azimuth correction: %.2f  \n', azimuth_corr_new);
119
120         if azimuth_corr_new > 0
121             fprintf('  Correct to the right\n');
122         elseif azimuth_corr_new < 0
123             fprintf('  Correct to the left\n');
124         else
125             fprintf('  No correction needed\n');
126         end
127
128     else
129         fprintf('No exact solution for v0=%.0f m/s, closest is %.0f m/s\n'
                , v0_target, solutions(min_idx,1));
130         fprintf('Closest elevation: %.2f  , azimuth correction: %.2f  \n',
                solutions(min_idx,2), solutions(min_idx,3));
131     end
132 else
133     fprintf('No available solutions.\n');
134 end
135
136 if ~isempty(solutions)
137     plot_representative_trajectories(solutions, R_target, relative_height,
            g, dt, tmax, launch_altitude, wind, m, d, A);
138     plot_velocity_elevation_relation(solutions, R_target);
139 end
140
141 function solutions = solve_trajectory_with_wind(v0_list, R_target,
        relative_height, g, dt, tmax, launch_altitude, wind, m, d, A)
142     solutions = [];
143
144     for idx = 1:length(v0_list)
145         v0 = v0_list(idx);
146         elevation_min = 0.1;
147         elevation_max = 85.0;
148         elev_samples = linspace(elevation_min, elevation_max, 60);
149         range_errors = zeros(size(elev_samples));
150
151         for i = 1:length(elev_samples)
152             elevation = elev_samples(i) * pi/180;
153             [x_final, ~, success] = simulate_trajectory(v0, elevation, 0,
                    g, dt, tmax, relative_height, launch_altitude, wind, m, d,
                    A);
154             if success
155                 range_errors(i) = x_final - R_target;
156             else
```

```
                range_errors(i) = NaN;
            end
        end

        valid_idx = find(~isnan(range_errors));
        if length(valid_idx) < 2
            continue;
        end

        range_errors_valid = range_errors(valid_idx);
        elev_samples_valid = elev_samples(valid_idx);

        sign_changes = [];
        for i = 1:length(range_errors_valid)-1
            if range_errors_valid(i) * range_errors_valid(i+1) < 0
                sign_changes = [sign_changes, i];
            end
        end

        if length(sign_changes) >= 1
            iL = sign_changes(1);
            elevL_low = elev_samples_valid(iL);
            elevL_high = elev_samples_valid(iL+1);
            elevation_low = bisect_elevation(v0, elevL_low, elevL_high,
               R_target, relative_height, g, dt, tmax, launch_altitude,
               wind, m, d, A);

            if ~isnan(elevation_low) && elevation_low > 0 && elevation_low
                < 45
                azimuth_corr = calculate_azimuth_correction_improved(v0,
                   elevation_low*pi/180, R_target, relative_height, g, dt,
                    tmax, launch_altitude, wind, m, d, A);
                solutions = [solutions; v0, elevation_low, azimuth_corr,
                   1];
            end
        end
    end
end
function [x_final, y_final, success, trajectory] = simulate_trajectory(v0,
    elevation, azimuth, g, dt, tmax, relative_height, launch_altitude,
   wind, m, d, A)
    vx0 = v0 * cos(elevation) * cos(azimuth);
    vy0 = v0 * cos(elevation) * sin(azimuth);
    vz0 = v0 * sin(elevation);

    x = 0; y = 0; z = 0;
    vx = vx0; vy = vy0; vz = vz0;

    success = false;
    trajectory = [];

```

```
    for t = 0:dt:tmax
        current_altitude = launch_altitude + z;
        [rho, ~, a] = atmosphere_ISA(current_altitude);
        v = sqrt(vx^2 + vy^2 + vz^2);
        M = v / a;
        Cd = Cd_vs_Mach(M);
        k = 0.5 * rho * Cd * A / m;

        vrel_x = vx - wind.wx;
        vrel_y = vy - wind.wy;
        vrel_z = vz - wind.wz;
        vrel = sqrt(vrel_x^2 + vrel_y^2 + vrel_z^2);

        ax = -k * vrel * vrel_x;
        ay = -k * vrel * vrel_y;
        az = -g - k * vrel * vrel_z;

        vx = vx + ax * dt;
        vy = vy + ay * dt;
        vz = vz + az * dt;

        x_prev = x; y_prev = y; z_prev = z;

        x = x + vx * dt;
        y = y + vy * dt;
        z = z + vz * dt;

        if nargout > 3
            trajectory = [trajectory; x, y, z];
        end

        if z <= relative_height && t > 0 && z_prev > relative_height
            frac = (relative_height - z_prev) / (z - z_prev);
            x_final = x_prev + frac * (x - x_prev);
            y_final = y_prev + frac * (y - y_prev);
            success = true;
            return;
        end
    end

    x_final = x;
    y_final = y;
    success = false;
end

function elevation_deg = bisect_elevation(v0, elev_low_deg, elev_high_deg,
    R_target, relative_height, g, dt, tmax, launch_altitude, wind, m, d, A
   )
    max_iter = 30;
    tol = 0.1;

```

```
    for iter = 1:max_iter
        elev_mid_deg = 0.5 * (elev_low_deg + elev_high_deg);
        [x_mid, ~, success_mid] = simulate_trajectory(v0, elev_mid_deg*pi
            /180, 0, g, dt, tmax, relative_height, launch_altitude, wind, m
            , d, A);

        if ~success_mid
            elev_high_deg = elev_mid_deg;
            continue;
        end

        error_mid = x_mid - R_target;
        [x_low, ~, success_low] = simulate_trajectory(v0, elev_low_deg*pi
            /180, 0, g, dt, tmax, relative_height, launch_altitude, wind, m
            , d, A);

        if success_low
            error_low = x_low - R_target;
            if error_low * error_mid <= 0
                elev_high_deg = elev_mid_deg;
            else
                elev_low_deg = elev_mid_deg;
            end
        else
            elev_low_deg = elev_mid_deg;
        end

        if abs(error_mid) < tol
            break;
        end
    end

    elevation_deg = elev_mid_deg;
end

function azimuth_correction = calculate_azimuth_correction_improved(v0,
    elevation, R_target, relative_height, g, dt, tmax, launch_altitude,
    wind, m, d, A)
    max_iter = 20;
    tol = 0.1;
    azimuth_low = -15 * pi/180;
    azimuth_high = 15 * pi/180;

    for iter = 1:max_iter
        azimuth_mid = 0.5 * (azimuth_low + azimuth_high);
        [x_mid, y_mid, success_mid] = simulate_trajectory(v0, elevation,
            azimuth_mid, g, dt, tmax, relative_height, launch_altitude,
            wind, m, d, A);

        if ~success_mid
            if azimuth_mid > 0
```

```
                azimuth_high = azimuth_mid;
            else
                azimuth_low = azimuth_mid;
            end
            continue;
        end

        error_mid = y_mid;

        [~, y_low, success_low] = simulate_trajectory(v0, elevation,
           azimuth_low, g, dt, tmax, relative_height, launch_altitude,
           wind, m, d, A);

        if success_low
            error_low = y_low;
            if error_low * error_mid <= 0
                azimuth_high = azimuth_mid;
            else
                azimuth_low = azimuth_mid;
            end
        else
            azimuth_low = azimuth_mid;
        end

        if abs(error_mid) < tol
            break;
        end
    end

    azimuth_correction = azimuth_mid * 180/pi;
    flight_time = calculate_flight_time(v0, elevation, azimuth_mid,
       relative_height, g, dt, tmax, launch_altitude, wind, m, d, A);

    fprintf('  Azimuth correction detailed analysis:\n');
    fprintf('    Relative height: %.1f m\n', relative_height);
    fprintf('    Estimated flight time: %.2f s\n', flight_time);
    fprintf('    Crosswind: %.1f m/s\n', wind.wy);
    fprintf('    Theoretical drift: %.1f m\n', wind.wy * flight_time);
    fprintf('    Initial velocity: %.0f m/s\n', v0);
    fprintf('    Elevation: %.1 f  \n', elevation * 180/pi);
end

function flight_time = calculate_flight_time(v0, elevation, azimuth,
   relative_height, g, dt, tmax, launch_altitude, wind, m, d, A)
    [~, ~, success, trajectory] = simulate_trajectory(v0, elevation,
       azimuth, g, dt, tmax, relative_height, launch_altitude, wind, m, d,
        A);

    if success && ~isempty(trajectory)
        flight_time = size(trajectory, 1) * dt;
    else
```

```
        vz0 = v0 * sin(elevation);
        if relative_height == 0
            flight_time = 2 * vz0 / g;
        else
            flight_time = (vz0 + sqrt(vz0^2 + 2*g*relative_height)) / g;
        end
    end
end

function [rho, T, a] = atmosphere_ISA(H_m)
    p0 = 101325; T0 = 288.15; L = 0.0065; R = 287.05287; g0 = 9.80665;
       gamma = 1.4;
    if H_m < 0, H_m = 0; end
    if H_m < 11000
        T = T0 - L*H_m;
        p = p0 * (T/T0)^(g0/(R*L));
    else
        T = T0 - L*11000;
        p = p0 * (T/T0)^(g0/(R*L)) * exp(-g0*(H_m-11000)/(R*T));
    end
    rho = p/(R*T);
    a   = sqrt(gamma*R*T);
end

function Cd_eff = Cd_vs_Mach(M)
    if M < 0.8
        Cd_eff = 0.47;
    elseif M < 1.2
        Cd_eff = 0.47 + (1.10-0.47)*(M-0.8)/(1.2-0.8);
    elseif M < 2.0
        Cd_eff = 1.10 - (1.10-0.95)*(M-1.2)/(2.0-1.2);
    elseif M < 3.0
        Cd_eff = 0.95 - (0.95-0.90)*(M-2.0)/(3.0-2.0);
    else
        Cd_eff = 0.90;
    end
end
```

References

[1] Zhang, Y. *Tactical Missile Flight Mechanics Design*. Beijing: Yuhang Publishing House, 1998.

[2] Wang, Z., & Zhou, W. *Exterior Ballistics Design Theory and Methods*. Beijing: Science Press, n.d., pp. 7–8.

[3] China Meteorological News Agency. Classification of Wind Vectors and Wind Force Levels [EB/OL]. (2018-07-17) [2024-11-09].

[4] Li, H., & Ye, P. Simulation Tests on the Influence of Reentry-Phase Wind Field on Ballistic Missile Accuracy. *Tactical Missile Control Technology*, 2006(4): 46.

[5] Zheng, X., & He, Y. Modeling and Simulation of Missile under Wind Disturbance. *Tactical Missile Control Technology*, 2013, 30(1): 48.

[6] Zang, G., & Li, S. *Aerodynamics of Ballistic Projectiles*. Beijing: Ordnance Industry Press, 1989.

[7] Yan, J., Li, Z., & Dong, L. Meteorological Corrections for Ballistic Missile Reentry. *Launch Vehicle Technology*, 2001(2): 1–2.

[8] Guo, X. Oblique Projectile Motion with Air Resistance Proportional to the Square of Velocity. *Physics Bulletin*, 2017(6): 63–66.

[9] Li, W., Yan, H., & Zhang, W. Ballistic Simulation of a Certain Projectile under Wind Disturbance. *Computer Technology and Development*, 2011, 21(1): 247–248.