# Compilation of an Artillery Fire Table Based on the Loth Model

Team 207, Problem B

November 8, 2025


## Abstract

Our study develops a ballistic model for spherical artillery projectiles consistent with mid-19th century technology. The model integrates gravity, air resistance, and atmospheric variations, employing the Loth drag coefficient model valid from subsonic to supersonic speeds. Its primary contribution is a practical artillery firing table system, providing baseline firing data for three discrete propellant charges and two firing modes, alongside corrections for wind, altitude, and elevation difference. Designed for simplicity and ease of use, this firing table enables commanders to rapidly determine firing solutions under operational conditions where computational equipment is unavailable. Detailed sections of the table and application examples are provided in the appendix. During the compilation of the tables, the R-squared values for all fitted models exceeded 0.95, with some approaching 1.0. Numerical simulation and validation demonstrated that the model's error is less than 0.1% within standard engagement ranges, with no discernible trend of error amplification. These simulation results collectively indicate that the proposed artillery firing table possesses high accuracy and robustness. Calculations successfully demonstrated multiple firing solutions for a 1200-meter target under different charges, proving the table's reliability for rapid battlefield aiming. This work reconstructs historical firing procedures, merging historical authenticity with computational analysis, and highlights the method's utility in resource-limited contexts.


**Key Words: Artillery, Loth models, Artillery firing tables, Aerodynamics**

# Contents

# 1 Introduction

## 1.1 Background

Artillery is a combat system composed of long-range weapons. Unlike infantry small arms, it is characterized by extended range and formidable destructive power. In its early stages, it primarily served as heavy, relatively stationary siege weapons for besieging fortifications. With technological advancements, field artillery has become increasingly portable and maneuverable. The artillery system encompasses multiple types, including cannons with low trajectories and high muzzle velocities, howitzers with high trajectories and variable charge capabilities, and mortars with high-arcing trajectories.

**Figure 1:** Artillery from 1825 to 1875

150 to 200 years ago, artillery development underwent a pivotal transition from muzzle-loading smoothbore cannons to breech-loading rifled guns. Key characteristics of this era included: the introduction of rifling to enhance accuracy and range; the gradual replacement of muzzle-loading mechanisms with breech-loading systems, increasing rate of fire and operational safety; the inability to continuously adjust projectile kinetic energy due to pre-packaged propellant charges of fixed weight, necessitating discrete control through varying the number of charges; Casting materials shifted from cast iron to stronger steel, enhancing durability and load capacity; simultaneously, standardized designs and interchangeable parts facilitated mass production and maintenance. Additionally, explosive shells gained widespread adoption over solid projectiles, significantly increasing lethality. Drawing upon these historical technological characteristics, this paper will establish corresponding ballistic models to reconstruct the firing accuracy and operational procedures of artillery from this era.

## 1.2 Problem Restatement

Building upon the historical context outlined above, this problem requires us to construct a cannon ball trajectory model consistent with mid-19th century technological capabilities. Specifically, we must establish a ballistic model capable of accurately describing the projectile's motion through the air. This model will simulate the flight path of a spherical projectile under specified initial conditions and ultimately determine the firing parameters necessary to achieve target impact.

For the scenario outlined in the first question, where the artillery piece and target share the same elevation and are separated by a horizontal distance of 1200 meters, we must determine the initial velocity and elevation angle of the projectile required for a hit. To achieve this, we first establish a set of dynamic equations describing the projectile's motion. This equation system must comprehensively account for external factors such as gravity, air resistance, and potential wind forces to construct a complete ballistic model.

Building upon this foundation, we address the second question. By setting the target position as boundary conditions, we can solve this dynamic system to determine the combination of exit velocity and elevation angle required to hit the target. Ultimately, based on the model's computational results, we will provide artillery officers with a clear and concise artillery firing table. This will enable them to rapidly estimate and utilize the data effectively in real combat scenarios.

**Figure 2:** Shells and propellant charges of artillery

When determining the projectile's exit velocity, it is crucial to note that this velocity cannot be varied continuously. The projectile of the cannon is propelled by the combustion gases generated from the explosion of propellant charges, as illustrated in Figure 2. To alter the exit velocity, one must change the number of propellant charges. While an electromagnetic railgun might allow for continuous adjustment of exit velocity, the conventional artillery system studied in this study can only achieve discrete changes in this parameter.

# 2 Notations

| Symbols | Description | Unit |
|---|---|---|
| $m$ | Shell mass | $kg$ |
| $S$ | Characteristic area | $m^2$ |
| $u$ | Local wind speed | $m/s$ |
| $s$ | Local Sound Speed | $m/s$ |
| $\gamma$ | Specific heat ratio of air | $J/(kg \cdot K)$ |
| $R$ | Universal Gas Constant | $J/(mol \cdot K)$ |
| $v$ | Exit velocity | $m/s$ |
| $\theta$ | Elevation angle | ° |
| $\varphi$ | Azimuth | ° |
| $\rho$ | Air density | $kg/m^3$ |
| $Re$ | Reynolds number | [1] |
| $M_a$ | Mach number | [1] |
| $C_d$ | Drag coefficient | [1] |
| $\mu$ | Kinematic viscosity | $m^2/s$ |

# 3 Assumptions

## 3.1 Circular probability error

The recoil generated when a shell is fired causes the barrel to vibrate, resulting in a slight deviation of the muzzle at the instant the projectile leaves the barrel. This random error causes the actual impact points of the shells to scatter in an elliptical pattern around the target point, known as circular probability error. However, the range considered in this problem is relatively short. The dispersion range caused by recoil will be significantly smaller than that in medium-to-long-range firing. Therefore, in the preliminary theoretical calculations of the model, we will temporarily exclude the circular probability error caused by recoil. That is, we assume that under windless conditions and standard atmospheric conditions, the projectile can precisely hit the target point along an ideal trajectory.

## 3.2 Coriolis force

The Earth's rotation induces the Coriolis effect, generating a Coriolis force—an inertial force perpendicular to the direction of an object's velocity—which causes a deflection in its trajectory. Its expression is:

$$\overrightarrow{F_c} = -2m\left(\vec{\omega}\times\vec{v}\right) \tag{3-1}$$

Where $m$ is the projectile mass, $v$ is the projectile velocity, and $\omega$ is the angular velocity of Earth's rotation.

For the range specified in this problem, along with the maximum initial velocity of the projectile and its corresponding flight time, estimation indicates that the lateral deviation at the impact point caused by the Coriolis force is negligible. Therefore, to balance computational complexity with accuracy requirements, we assume the influence of the Coriolis force can be disregarded in the context of this mission scenario.

## 3.3 Gravitational acceleration gradient

As the altitude of a projectile increases, its distance from the center of the Earth grows. According to the law of universal gravitation, the actual gravitational force acting on it decreases slightly with increasing height, meaning a gravitational gradient exists. Considering the relative change in gravitational acceleration:

$$\frac{\Delta g}{g} \approx -\frac{2h}{R} \tag{3-2}$$

Substituting typical ballistic altitude values reveals that gravitational variation is far less than one-thousandth. Therefore, this model assumes that gravitational acceleration remains constant $9.80665kg\cdot m/s^2$ throughout the entire trajectory of the projectile.

## 3.4 Some Other Assumptions

- In this article, sea-level standard atmospheric conditions are used as the reference quantity:
$$T_0 = 288.15K\ ,\quad p_0 = 101.325kPa\ ,\quad \rho_0 = 1.225kg/m^3$$
- The loss of shell mass during launch and flight is not considered.
- Based on the background information, introduce the hypothesis: The chemical energy of each standard propellant charge is converted into the kinetic energy of the projectile's exit velocity at an identical rate, i.e.,
$$E_k = nkE_{chem} \tag{3-3}$$
and thus obtain the velocity formula
$$v = v_{\max}\sqrt{\frac{n}{n_{\max}}} \tag{3-4}$$

Here, $n$ denotes the number of propellant charges, $n_{\max}$ represents the maximum number of propellant charges that can be accommodated(Here we set $n_{\max}=3$), is the conversion ratio (calculated to be independent of v), $E_{chem}$ is the chemical energy per propellant charge, and $v_{\max}$ is the maximum exit velocity at $n_{\max}$ (given $v_{\max}=450m/s$).

Based on the simplified assumptions above, we can construct a ballistic model that is both physically realistic and computationally efficient while maintaining computational accuracy.

# 4 Modelling

## 4.1 Physics Model for Artillery Firing

After establishing the fundamental assumptions, we began constructing a physical model for artillery firing. This model will comprehensively account for the effects of gravity, air resistance, and environmental factors on the trajectory, predicting the projectile 's path by solving the equations of motion. The physical concept diagram of a projectile's flight is as follows:

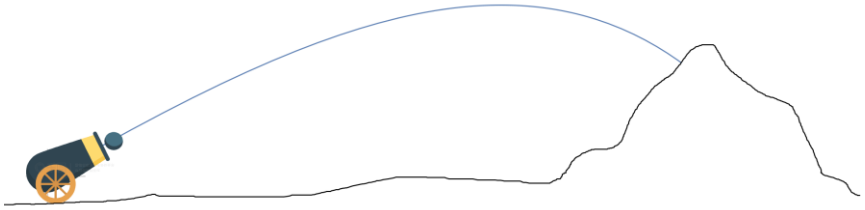


**Figure 3**: Diagram of Projectile Motion in a Parabolic Trajectory

At the same time, we consider the forces acting on the projectile and its motion in the following coordinate system.

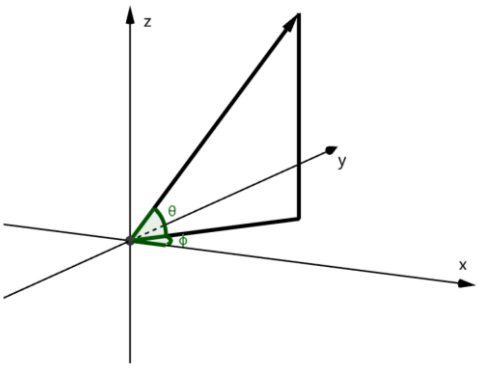



**Figure 4**: Coordinate System Diagram

Here we analyze the forces acting on an object. Objects are subject to gravity acting vertically downward:

$$G=mg \tag{4-1}$$

Additionally, objects experience drag force opposing their velocity. Here we employ a drag model widely used in fluid dynamics and ballistics. The aerodynamic drag acting on a projectile during flight is given by the following formula:

$$\overrightarrow{F_D} = -\frac{1}{2} C_D \rho S v_r \cdot \overrightarrow{v_r} \tag{4-2}$$

Among these, $C_D$ is the drag coefficient, whose value depends on the shape of the projectile and the Reynolds number; $\rho$ is the density of air; $S$ is the characteristic area of the projectile, i.e., its central cross-sectional area; $v_r$ is the projectile's velocity relative to the air.

Using Newton's second law, we can derive the following set fundamental of equations:

$$\begin{cases} \dfrac{d^2x}{dt^2} = -\dfrac{1}{m}\cdot\dfrac{1}{2}\rho\sqrt{\left(\dfrac{dx}{dt}-u_x\right)^2+\left(\dfrac{dy}{dt}-u_y\right)^2+\left(\dfrac{dz}{dt}\right)^2}\cdot\left(\dfrac{dx}{dt}-u_x\right)\cdot C_d S \\ \dfrac{d^2y}{dt^2} = -\dfrac{1}{m}\cdot\dfrac{1}{2}\rho\sqrt{\left(\dfrac{dx}{dt}-u_x\right)^2+\left(\dfrac{dy}{dt}-u_y\right)^2+\left(\dfrac{dz}{dt}\right)^2}\cdot\left(\dfrac{dy}{dt}-u_y\right)\cdot C_d S \\ \dfrac{d^2z}{dt^2} = -\dfrac{1}{m}\cdot\dfrac{1}{2}\rho\sqrt{\left(\dfrac{dx}{dt}-u_x\right)^2+\left(\dfrac{dy}{dt}-u_y\right)^2+\left(\dfrac{dz}{dt}\right)^2}\cdot\dfrac{dz}{dt}\cdot C_d S - g \\ \left.\dfrac{dx}{dt}\right|_{t=0} = v\cos\theta\cos\varphi \\ \left.\dfrac{dy}{dt}\right|_{t=0} = v\cos\theta\sin\varphi \\ \left.\dfrac{dz}{dt}\right|_{t=0} = v\sin\theta \end{cases} \tag{4-3}$$

### 4.1.1 Models for Key Physical Quantities

Hereafter, we present the models and formulas for several key physical quantities discussed in this paper. The relationships they describe are derived from classical and authoritative theories or experiments [1-4].

1) Temperature: Temperature varies with altitude. In this article, the projectile remains within the troposphere throughout. We use the temperature expression from the International Standard Atmosphere (ISA) model:

$$T_{ISA} = T_0 - Lh \tag{4-4}$$

Here, $L$ is the vertical temperature lapse rate, set at $0.0065°C/m$ ; $h$ denotes the elevation difference between the current location and the horizontal plane.

2) Air density: Air density is related to the altitude and the temperature at the current location. We employ an exponential atmospheric model:

$$\rho = \rho_0 \exp\left(\frac{-g_0 h}{RT}\right) \tag{4-5}$$

3）Air viscosity: Viscosity varies with temperature. According to the Sutherland correlation, the viscosity formula is as follows:

$$\mu = \frac{a\sqrt{T}}{1+\frac{b}{T}} \tag{4-6}$$

In the equation above, $a$ and $b$ are constants determined experimentally,Since we employ air in the ISA model, we adopt the following constants in the equation:

$$a = 1.485\times10^{-6} kg/\left(m\cdot s\cdot K^{0.5}\right), b = 110.4K$$

4）Speed of sound: We employ the ideal gas isentropic sound velocity model to calculate the speed of sound, with the formula as follows:

$$s = \sqrt{\frac{\gamma RT}{M}} \tag{4-7}$$

5）dimensionless numbers: The Mach number $Ma$ and the Reynolds number $Re$, defined as follows:

$$Ma = \frac{v}{s}, Re = \frac{\rho l v}{\mu} \tag{4-8}$$

### 4.1.2 Model of the Drag Coefficient

The drag coefficient $C_D$ is the most critical and complex parameter in ballistic calculations. It comprehensively reflects complex flow phenomena such as flow separation and shock formation. Stokes' law or classical incompressible flow resistance formulas no longer apply in transonic and supersonic ranges.

Considering that projectiles may decelerate from supersonic to subsonic speeds during flight, we need a drag model that covers a wide range of Mach numbers. The semi-empirical model proposed by Loth et al. [5], which incorporates compressibility corrections to the classical Clift-Gauvin formula, can accurately describe the drag characteristics of spheres from subsonic to hypersonic speeds. Therefore, it is adopted in this paper. The core of this model lies in modifying the classical Clift-Gauvin formula for incompressible flow resistance to capture the effects of compressibility across different Reynolds number ranges. Specifically, we introduce the Mach number correction factor defined by the formula

$$C_M = 1.65 + 0.65\tanh\left(4M_P - 3.4\right) \quad \text{for } M_p < 1.5 \quad (4\text{-}9)$$

$$C_M = 2.18 - 0.13\tanh\left(0.9M_P - 2.7\right) \quad \text{for } M_p > 1.5 \quad (4\text{-}10)$$

in order to describe the overall trend of the drag coefficient with Mach number in the high Reynolds number region $\left(10,000 < Re_p < 1,000,000\right)$. It represents the ratio of the drag coefficient to the incompressible flow reference value.

Building upon this foundation, to extend the compressibility effects to a broader Reynolds number range $\left(45 < Re_p < Re_{p,crit}\right)$,we adopted the modified Clift-Gauvin formula proposed by them, as follows:

$$C_D = \frac{24}{Re_p}\left(1 + 0.15Re_p^{\,0.687}\right)H_M + \frac{0.42C_M}{1 + \left(42,500 / Re_p^{\,1.16C_M}\right) + \left(G_M / Re_p^{\,0.5}\right)} \quad (4\text{-}11)$$

This formula incorporates corrections via two functions, $G_M$ and $H_M$ .Function $G_M$ primarily corrects the empirical term in the denominator of the drag formula under low-to-medium Reynolds number conditions, precisely capturing the increase in drag with the rising Mach number in this range; function $H_M$ serves as a pre-factor, mainly correcting the plateau value of the drag coefficient under high Reynolds number conditions. Functions Gm and Hm are defined as follows:

$$G_M = 166M_P^{\,3} + 3.29M_P^{\,2} - 10.9M_P + 20 \quad \text{for } M_p < 0.8 \quad (4\text{-}12)$$

$$G_M = 5 + 40M_P^{\,-3} \quad \text{for } M_p > 0.8 \quad (4\text{-}13)$$

$$H_M = 0.0239M_P^{\,3} + 0.212M_P^{\,2} - 0.074M_P + 1 \quad \text{for } M_p < 1 \quad (4\text{-}14)$$

$$H_M = 0.93 + \frac{1}{3.5 + M_P^{\,5}} \quad \text{for } M_p > 1 \quad (4\text{-}15)$$

Therefore, this study will comprehensively employ the model system formed by the aforementioned equations as the core basis for calculating the drag force experienced by spherical projectiles in compressible flow regions.

## 4.2 Model Solution

For Question 1, substitute into the system of equations (4-3) the horizontal distance $\Delta x = 1200m$ and vertical distance $\Delta z = 0m$ between the target and the artillery piece, the calculated exit velocity-elevation angle phase diagram is shown in Figure 5. The projectile trajectory diagrams for different charge configurations are shown in Figures 6 through 8.

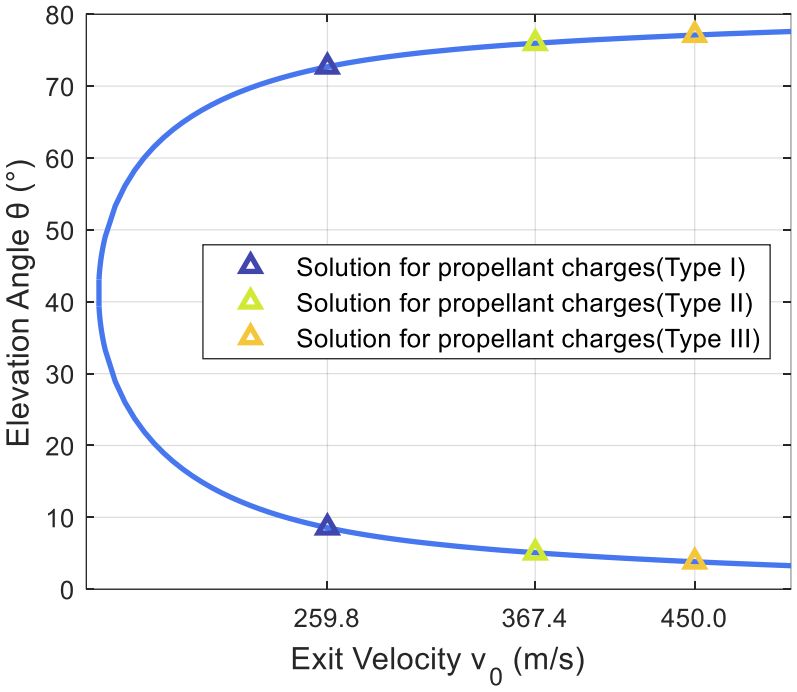



**Figure 5**: Exit Velocity-Elevation Angle Diagram

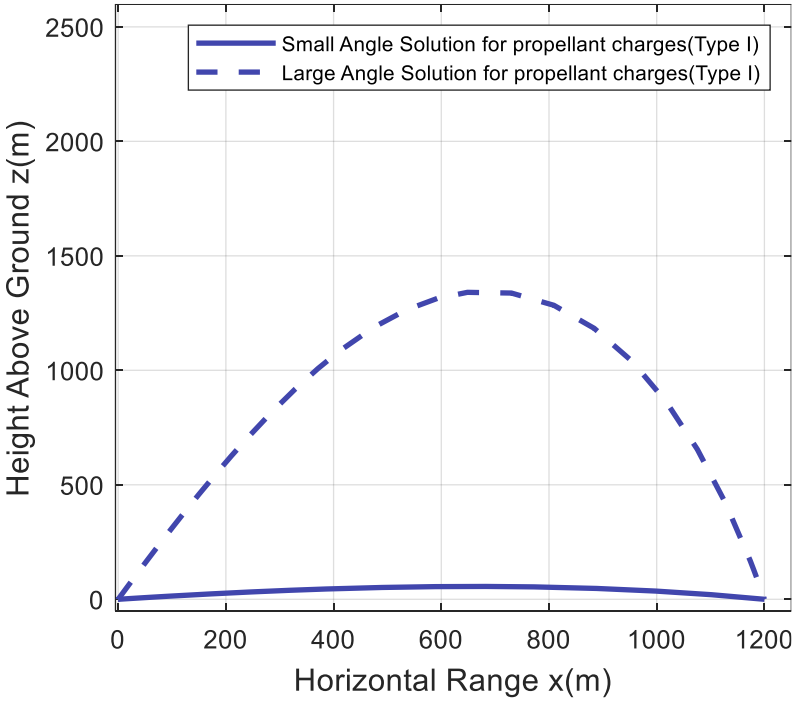



**Figure 6**: Ballistic trajectory of the projectile under Loading Method I

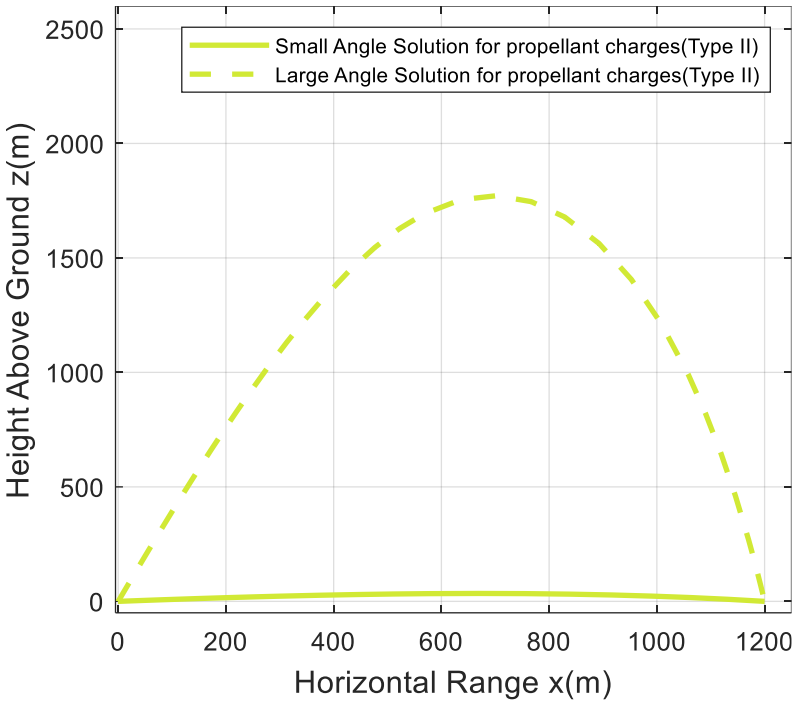



**Figure 7**: Ballistic trajectory of artillery shells under Loading Method II

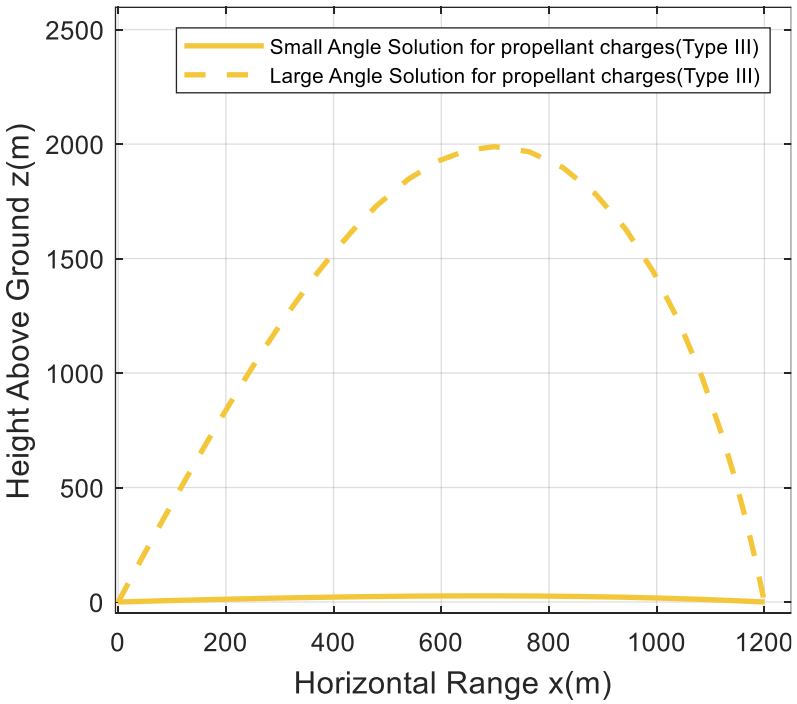



**Figure 8**: Ballistic trajectory of artillery shells under Loading Method III

According to our assumptions, due to the discrete nature of the propellant charge, the velocity can only take discrete values, specifically:

$$v_1 = 259.8076\text{m/s}\,,\quad v_2 = 367.4235\text{m/s}\,,\quad v_3 = 450.0000\text{m/s}$$

The Elevation angle solutions corresponding to each speed are shown in the table below.

**Table 1**: The exit velocity and Elevation angle obtained from Problem 1

| $v/\left(m\cdot s^{-1}\right)$ | $\theta_1/°$ | $\theta_2/°$ |
|---|---|---|
| $v_1$ | 8.5623 | 72.6924 |
| $v_2$ | 5.0794 | 75.9605 |
| $v_3$ | 3.8231 | 77.1028 |

Among these, the small angle $\theta_1$ and the large angle $\theta_2$ can both hit the target.

## 4.3 Artillery Fire Table Method

### 4.3.1 The Development and Rationality of Artillery Fire Tables

The use of artillery tables dates back to the early days of artillery, when gunners relied on accumulated experience and rough estimates for aiming. By the mid-19th century, the advent of rifled barrels and standardized propellants made the need for systematic aiming data increasingly apparent. Early tables were mostly compiled through costly and time-consuming live-fire trials. By the late 19th century, ballisticians such as Francesco Sciacchi introduced mathematical ballistics theory, enabling the theoretical calculation of projectile trajectories. This breakthrough paved the way for more precise and universally applicable firing tables.

The fundamental principle of artillery firing tables lies in decoupling complex ballistic calculations from real-time firing operations. Its core advantage is transforming the cumbersome process of solving nonlinear differential equations into a battlefield-ready workflow of table lookups and simple arithmetic operations. This is achieved by pre-calculating ballistic data under standard conditions and attaching correction factors for varying environmental conditions. This approach demonstrated exceptional rationality and practicality during historical periods of scarce computational resources:

- **Enhance combat effectiveness:** Artillery officers no longer need to perform complex calculations on-site, enabling them to quickly obtain initial firing parameters and significantly reduce reaction time.
- **Ensure shooting accuracy:** Benchmark data pre-calculated based on rigorous physical models and validated correction factors provide a reliable theoretical foundation for shooting, eliminating errors inherent in purely empirical estimation.
- **Enhance environmental adaptability:** By incorporating corrections for wind, elevation, temperature, and other factors, the artillery maintains a high hit rate under varying environmental conditions.

- **Lower the operating threshold**: Trained officers can execute relatively precise artillery missions, reducing the mathematical skills required of operators.

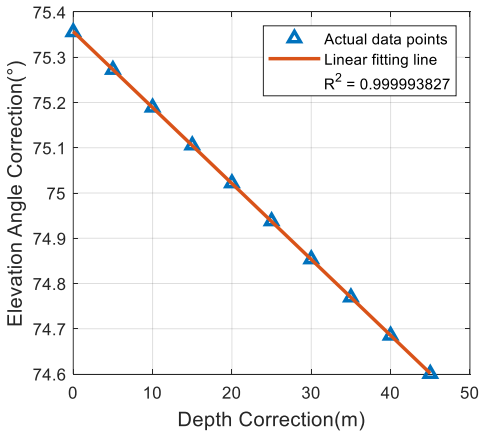
Therefore, the shooting table method adopted in this paper was not only a historically inevitable choice but has also demonstrated full rationality and effectiveness in model validation and practical application.

### 4.3.2 Principles of Target Chart Compilation

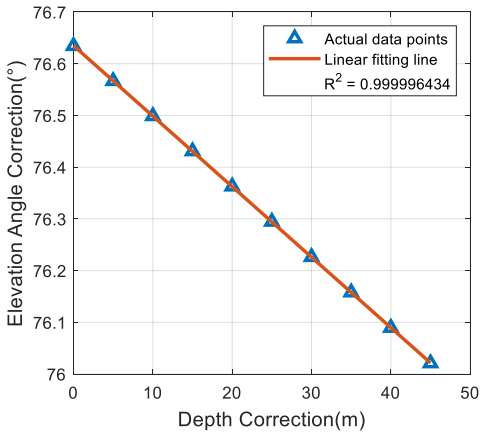
Based on the ballistic model established in Sections 4.1–4.2, the reference elevation angles corresponding to reference ranges from 1000 meters to 1500 meters, spaced at 50-meter intervals, were calculated separately for each propellant configuration.

Based on this, we use a program to determine the correction values under different conditions. Taking wind correction as an example, while keeping all other conditions constant at the same reference range, we only alter the wind speed. We calculate the range deviation of the shell for each wind speed, plot the scatter diagram, and then use linear fitting to describe the relationship between wind speed and deviation. The slope of the fitted line is extracted as the correction rate, yielding the correction value. During linear fitting of each variable, we employed $R^2$ tests, finding $R^2$ values consistently exceeding 0.95. This indicates strong linear fitting, confirming the high precision of the correction values. To facilitate adjustments by officers on the battlefield, all corrections in our artillery firing tables are rounded to whole numbers.

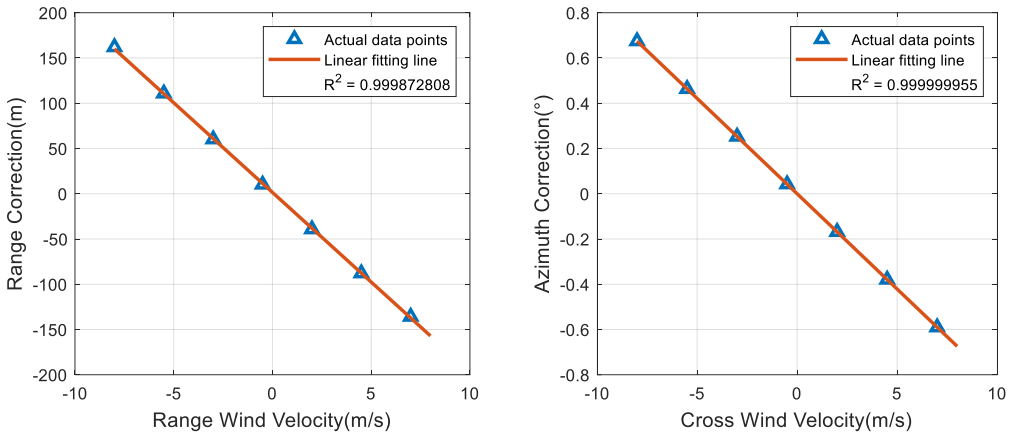
The same applies to other variables, which are ultimately compiled into a projection table, Below are some fitting curves. The graphical results indicate a strong linear correlation between the correction for range, crosswind velocity, range wind velocity, and the firing parameters, with R-squared values closely approaching 1.0.




(a)




(b)




(c) (d)

**Figure 9**: Fitting Curves Diagram

### 4.3.3 How to Use the Projection Chart

First, we provide abbreviations for the parameter names in the artillery firing table for concise representation, as follows:

**Table 2**: Abbreviations and Units for Artillery Fire Tables

| Parameters | Abbreviation | Unit |
|---|---|---|
| Reference Range | *BRm* | *m* |
| Reference elevation angle | *BEd* | ° |
| Reference elevation angle | *BEmi* | *mil* |
| Correction for per 10m depth | *DCmi* | *mil* |
| Correction for per 1m/s range wind | *LWCm* | *m* |
| Correction for per 1m/s crosswind | *CWCmi* | *mil* |
| Correction for per 10m elevation difference | *ECm* | *m* |
| Correction for per 100 meters Altitude | *ACm* | *m* |

As shown in the table, *mil* is a commonly used angular unit in artillery, its conversion relationship is $1\ mil = 0.06° = 0°3'36"$. Below, we provide the physical quantity codes and their positive/negative specifications:

Rule1: As indicated above, we have three propellant loading configurations: loading one propellant charge (Type I), loading two propellant charges (Type II), and loading three propellant charges (Type III), numbered 1 through 3.

Rule2: Similar to the calculation format in Section 4.2, for each corresponding target, we have two firing modes: direct fire (designated as a) and indirect fire (designated as b).

Rule3: For range wind, downwind is defined as positive and upwind as negative. For

crosswind, a right-hand rule is defined where the wind direction forms a right-handed curve with the horizontal forward direction and the vertical upward direction; positive is defined as the direction of the curve, and negative as the opposite direction.

For artillery firing tables applicable to all scenarios, we provide them in sections A1_a to A3_b of Appendix A. Example scenarios are presented in section A4. Below is the operational procedure flowchart and detailed instructions:

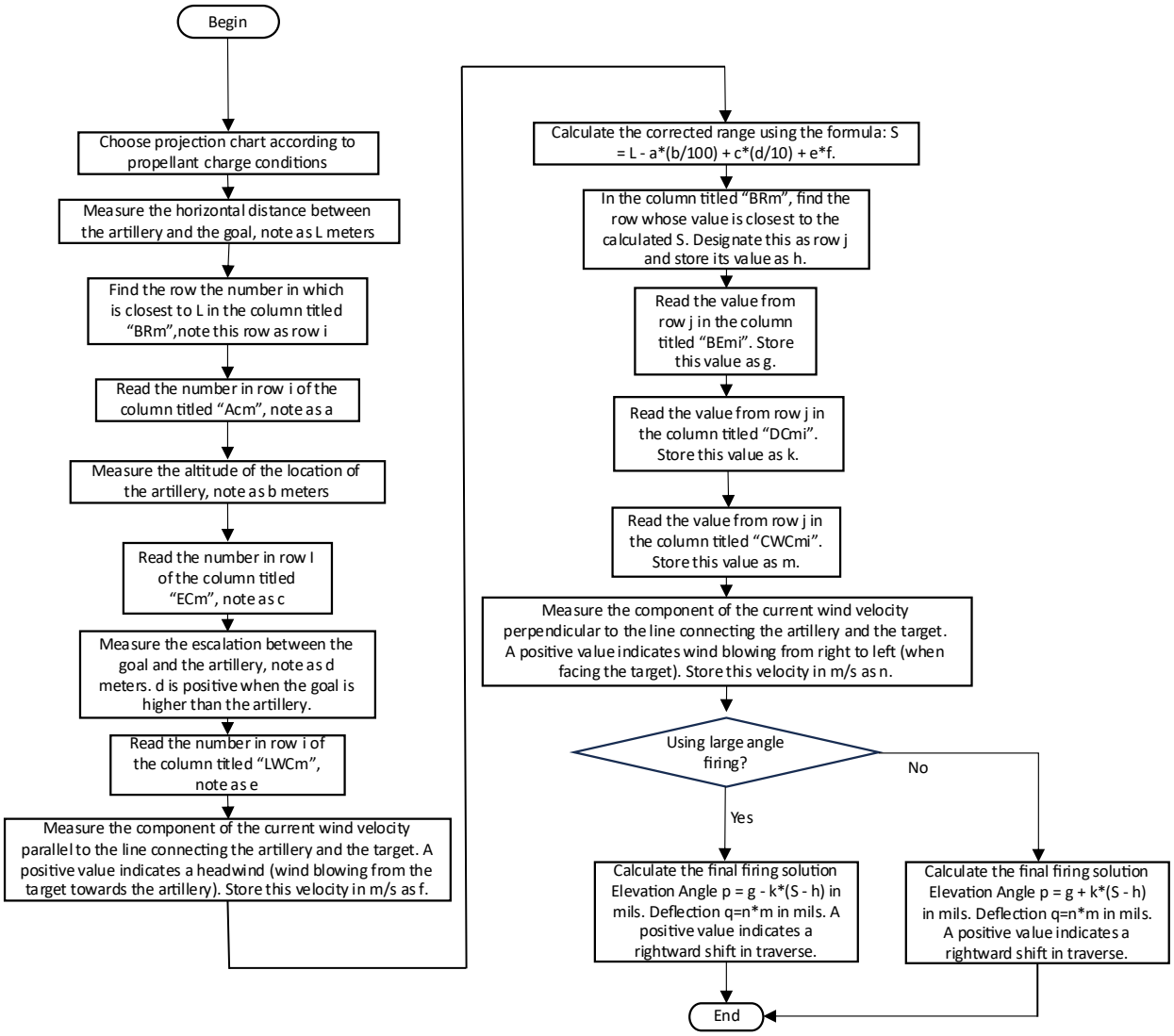



**Figure 10**: Flowchart for Using the Artillery Table

**Step1: Identify fundamental shooting elements**

**1.Acquire target information**:

- **Target distance**: Officers must be obtained through methods such as map surveying and observer reports.

- **Elevation difference**: Target elevation minus artillery position elevation.
- **Altitude of the position**: The elevation of our artillery position.

**2.Retrieve environmental information:**

- **Range wind speed**: The component of wind speed along the horizontal direction of travel.
- **Crosswind speed**: The component of wind velocity perpendicular to the horizontal direction of travel.

**3.Propellant Selection and Firing Mode:**

- **Select based on tactical requirements:** choose "direct fire" or "small angle" for direct engagements, and "indirect fire" or "large angle" for striking targets behind cover. Propellant selection is typically determined by range and ballistic requirements.

**Step2: Query reference data**

In the corresponding artillery firing table, locate the row with the reference range closest to the target distance.

**Step3: Calculate the respective correction values**

Now, we need to apply corrections based on environmental conditions. For each correction value, we round to the nearest integer as specified by IEEE 754.

**1.Depth Correction:**

**Rules:** For direct fire, the correction value is positive; for indirect fire, the correction value is negative. For data not found in the firing table, consider the largest value closest to it and use the difference between the two as the correction data.

**Calculation Method:** $\Delta\theta_{DC} = \left[ \pm \frac{x_{base} - x}{10} \cdot DCmi \right]$

**2.Range wind Correction**:

**Rules:** Range wind correction is opposite to wind direction. When downwind or range wind speed is positive, the correction value is negative; when headwind or range wind speed is negative, the correction value is positive.

**Calculation Method**: $\Delta x_{LW} = \left[ -\frac{u_x}{1} \cdot LWCm \right]$

**3.Crosswind Correction**:

**Rules:** Crosswind correction is opposite to the wind direction. When crosswind speed is positive, the correction value is negative (left to right); when crosswind speed is negative, the correction value is positive (right to left).

**Calculation Method**: $\Delta\varphi_{CW} = \left[ -\frac{u_y}{1} \cdot CWCmi \right]$

**4.Elevation Difference Correction:**

**Rules:** Elevation difference correction is positive. The greater the elevation difference, the more the projection must occur toward the rear, increasing the distance.

**Calculation Method**: $\Delta x_{EC} = \left[ \frac{\Delta z}{10} \cdot ECm \right]$

**5.Altitude correction:**

**Rules:** Altitude correction is negative. The higher the altitude, the thinner the air and the lower the air resistance, requiring a slightly forward trajectory and resulting in reduced distance.

**Calculation Method**: $\Delta x_{AC} = \left[ -\frac{h}{100} \cdot ACm \right]$

**Step4: Calculate the final data comprehensively**

Apply all corrections to the baseline data,

$$x_{final} = x_{base} + \Delta x_{LW} + \Delta x_{EC} + \Delta x_{AC}$$

Then, $x_{final}$ locates the corresponding $\theta_{base}$ value for the closest reference range, and calculates $\theta_{final}$.

$$\theta_{final} = \theta_{base} + \Delta\theta_{DC}$$

$$\varphi_{final} = \Delta\varphi_{CW}$$

# 5 Error Analysis

To verify the accuracy of the artillery firing table, we selected the Type III charge configuration and indirect fire conditions. We systematically analyzed the distance between the shell impact point and the aiming point under varying baseline ranges, elevation differences, and the effects of crosswind and downwind. The left panel below illustrates the impact of baseline range and target elevation difference on range error under these conditions, while the right panel shows a box plot of the range error.

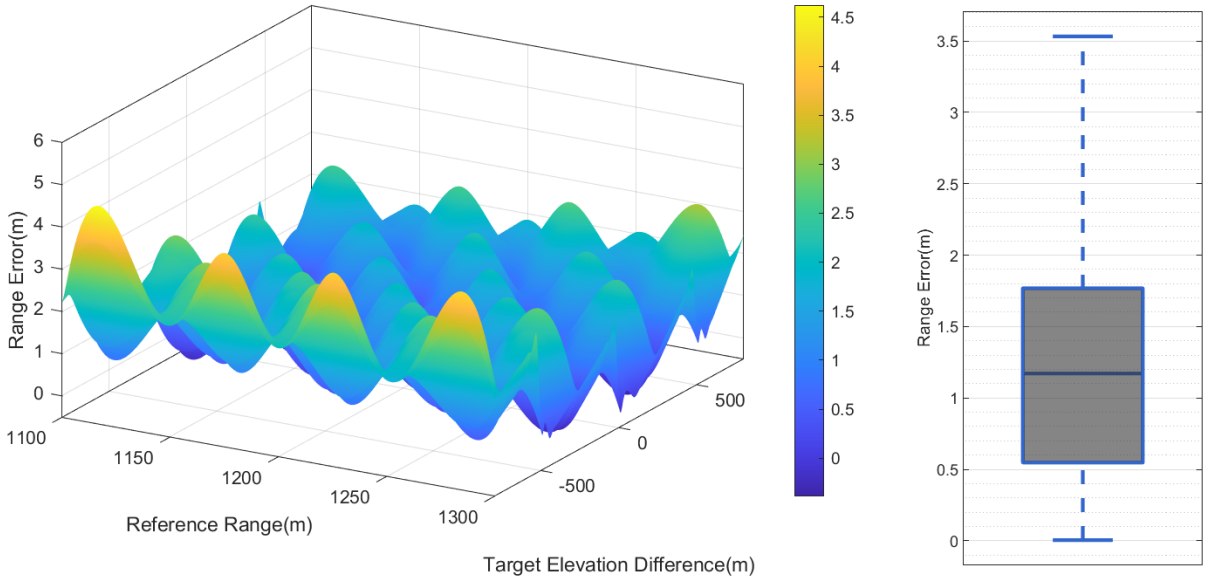



**Figure 11**: Thermal map of the impact of $x_{base}$ and $\Delta z$ on range error and its box plot

As mentioned earlier, we fitted the correction values under different conditions using a linear model and rounded the final results. Consequently, the range error exhibits a distinct periodic relationship with the baseline range and the target elevation difference. Near sampling points close to the firing table's compilation, the error is minimal; it gradually increases as the deviation from these sampling points grows. However, since a separate $ECm$ was defined for each $x_{base}$ during table creation, the overall range error does not increase proportionally with the growing difference between the baseline range and target elevation. Box plots indicate an average range error of 1.20m, a maximum error of 3.53m, and an overall error rate around 0.1%.

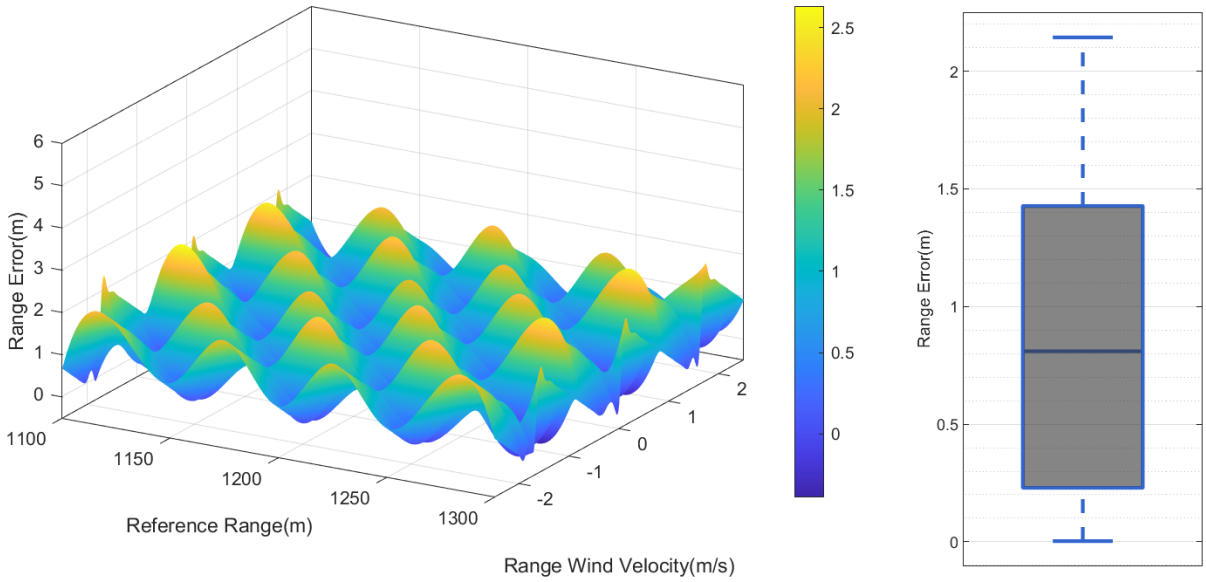



**Figure 12**: Thermal map of the impact of $x_{base}$ and $u_x$ on range error and its box plot

As shown in Figure 12, the effect of altering the reference range and crosswind on range error also exhibits periodic variation. The box plot indicates an average range error of 0.87 m and a maximum error of 2.14 m, with the error remaining below 0.1%.

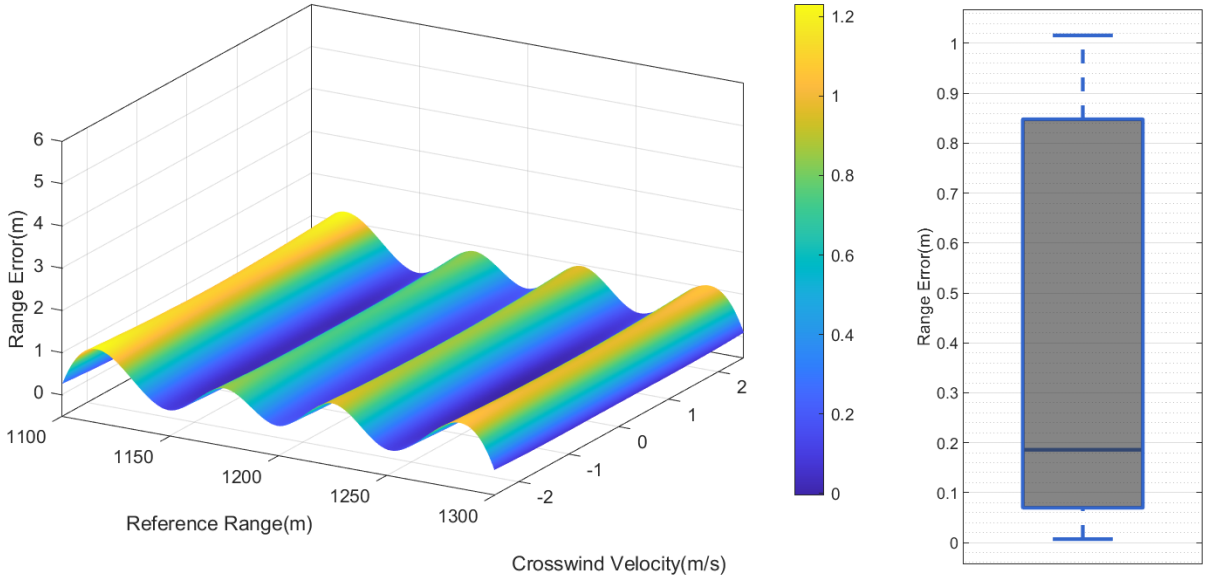



**Figure 13**: Thermal map of the impact of $x_{base}$ and $u_y$ on range error and its box plot

As shown in the figure above, the effects of altering the reference range and crosswind on range error differ significantly from those depicted in Figures 11 and 12. This is primarily because the influence of crosswind on projectile deflection exhibits a highly linear correlation with crosswind speed, enabling linear fitting models to achieve excellent impact point corrections. Meanwhile, the periodic variation in range error with reference range is mainly caused by rounding in the firing table data. Thermal analysis indicates that range error remains virtually unchanged with crosswind speed variations. Box plots reveal an average error of 0.43m and a maximum error of 1.02m, with overall error below 0.05%.

These results indicate that the artillery firing table fitted using a linear model is highly accurate, with range errors not increasing despite expanding influence conditions. In summary, this artillery firing table demonstrates excellent robustness and accuracy while meeting battlefield requirements for rapid fire control parameter binding.

# 6 Discussion

First, we will discuss the strengths and weaknesses of this paper. The primary strengths lie in the accuracy of the mechanistic model and the rationality of the firing table compilation method. When establishing the mechanistic model, the effects of temperature, altitude, and other factors on the atmosphere were comprehensively considered. Furthermore, the Loth model was utilized to calculate the drag coefficient in both supersonic and subsonic regimes, which contributes to the model's high accuracy. Linear fitting and error analysis have demonstrated that the firing table compiled herein possesses extremely high precision, thus validating the soundness of its compilation methodology.

However, this study also has several limitations. First, an analytical model was not established. The system of differential equations, constructed using the more complex parameters from the mechanistic model, is nonlinear and lacks an analytical solution. It can only be solved through numerical computation, which introduces additional error and uncertainty. Second, the model inadequately considers the projectile's transition phase as it decelerates from the supersonic to the subsonic region. Lastly, the model does not account for the effects of vibrations caused by insufficient structural strength of the cannon; in reality, barrel vibration and oscillation are major sources of artillery inaccuracy.

These shortcomings, arising from various data-related and theoretical difficulties, are challenging to resolve at present but are nonetheless objective realities. In the subsequent sections, we will explore other potential directions for this research.

## 6.1 Fire Correction and Trajectory Adjustment

Under realistic combat conditions such as military operations, artillery units find it difficult to obtain precise range, elevation differences, and azimuth angles of targets. At this point, the initial fire based on observation data and firing tables is often inaccurate, with its impact point deviating from the target by dozens to hundreds of meters. In practice, forward observers report the lateral and longitudinal deviations of the impact point to the artillery commander. Based on this feedback, the commander adjusts the elevation and azimuth angles of all artillery pieces by several mils. This method facilitates effective target engagement despite initial measurement inaccuracies.

Furthermore, artillery in military operations frequently needs to make rapid, small-scale adjustments to engagement targets. For instance, after several firing rounds, units might shift their fire to deeper enemy positions or engage retreating or moving targets. Under such circumstances, re-measuring all target parameters is excessively time-consuming. Artillery units

typically employ a range correction method based on the firing table, which involves adjusting only the gun's elevation angle by a few mils to expedite firing preparation. This is also the purpose of the $DCmi$ column in the firing tables developed in this study.

### 6.2 Sonic Boom in Transonic to Supersonic Transition

A sonic boom is the shockwave phenomenon generated when a projectile travels at supersonic speeds. The dissipation of its energy introduces additional drag and affects ballistic stability. This model indirectly accounts for the macro effects of sonic booms through vertical gradients in air density and the speed of sound: for instance, as altitude increases, thinner air reduces the speed of sound, causing the Mach number to rise for the same initial velocity. This intensifies the sonic boom, thereby increasing the drag correction factor.

The implications for firing table corrections are that when projectiles transition from supersonic to subsonic speeds, the nonlinear resistance changes near the sonic boom vanishing point may introduce errors. For high-precision requirements, incorporating a transient aerodynamic model can quantify the transient effects of sonic booms.

### 6.3 Shock Wave

At supersonic conditions, a shock wave forms ahead of the projectile, causing a sudden increase in pressure and a drastic change in the drag coefficient. The Loth drag coefficient model employed in this paper captures the compressibility effects induced by the shock wave through a Mach number correction factor.

The shock wave's influence is particularly pronounced in the ballistic trajectory: during the supersonic phase, the projectile decelerates more rapidly and follows a steeper trajectory; while at transonic speeds, the nonlinear variation in drag coefficient may induce trajectory jitter. Through error analysis, this model demonstrates prediction errors below 0.1% within the supersonic range, validating the Loth model's reasonable representation of shock wave effects. However, in actual combat, shock waves may also induce structural vibrations or stability issues in projectiles.

## 7 Conclusion

This study successfully developed a historically-informed ballistic model and artillery firing table for spherical projectiles of the mid-19th century. By integrating a physics-based trajectory model with a advanced drag coefficient formulation valid across multiple speed regimes, the flight behavior of projectiles under varying conditions was accurately captured. Under the

assumptions of this study, the mechanism model has obtained six sets of ballistic solutions for the horizontal distance $\Delta x = 1200m$ and vertical distance $\Delta z = 0m$.

The artillery firing table provides field officers with a practical tool for rapidly determining firing parameters. Systematic error analysis confirms the model's robustness, with range errors below 0.1% under typical combat conditions. The incorporation of environmental corrections further ensures adaptability and reliability in real-world scenarios.

This work bridges historical artillery practices with modern computational modeling, offering a validated framework for understanding 19th-century ballistics. The methodology and results can support further historical analysis, simulation-based training, or the design of period-accurate wargaming systems.

# References

[1] Mizner, R. A. ,Cham pion, K. S. W. , and Pond, H. L. ,1959 ARDC model atmosphere, AFCRCTR-59-267, Air Force Surveys in Geophysics No. 11, Bedford, MA, 1987.

[2] The US Standard Atmosphere, Washington DC, US Government Printing Office, 1976.

[3] Paine, S. , The Am Atmospheric Model, Smithsonian Astrophysical Observatory, Cambridge, MA, March 2014.

[4] M. H. Sadraey, Aircraft Performance: An Engineering Approach. (2;Second; ed.) 2024;2023;. DOI: 10.1201/9781003279068.

[5] Eric Loth, John Tyler Daspit, and Michael Jeong, Takayuki Nagata, and Taku Nonomura, “Supersonic and Hypersonic Drag Coefficients for a Sphere”, AIAA Journal, Vol. 59, No. 8, 2021, pp. 3261-3263.

# Appendices

## A1_a Type I, Small angle

| *BRm* | *BEd* | *BEmi* | *DCmi* | *LWCm* | *CWCmi* | *ECm* | *ACm* |
|---|---|---|---|---|---|---|---|
| 1000 | 06°40' | 111 | 2 | 3 | 1 | 6 | 3 |
| 1050 | 07°10' | 119 | 2 | 4 | 1 | 7 | 3 |
| 1100 | 07°42' | 128 | 2 | 4 | 2 | 8 | 3 |
| 1150 | 08°16' | 138 | 2 | 4 | 2 | 11 | 4 |
| 1200 | 08°51' | 147 | 2 | 5 | 2 | 12 | 4 |
| 1250 | 09°27' | 158 | 2 | 5 | 2 | 13 | 4 |
| 1300 | 10°06' | 168 | 2 | 6 | 2 | 14 | 5 |
| 1350 | 10°47' | 180 | 2 | 6 | 2 | 15 | 5 |
| 1400 | 11°29' | 191 | 2 | 6 | 2 | 17 | 5 |
| 1450 | 12°14' | 204 | 3 | 7 | 2 | 19 | 6 |
| 1500 | 13°02' | 217 | 3 | 7 | 2 | 20 | 6 |

- **Propellant charges Type I, Small Angle (Direct fire)**
- **The target elevation correction (*DCmi*) is positive when firing directly**
- **Range wind correction (*LWCm*) is opposite to the wind direction**
- **Crosswind correction (*CWCmi*) is opposite to the wind direction**
- **Elevation difference correction value (*ECm*) is positive**
- **Altitude correction value (*ACm*) is negative**

## A1_b Type I, Large angle

| *BRm* | *BEd* | *BEmi* | *DCmi* | *LWCm* | *CWCmi* | *ECm* | *ACm* |
|---|---|---|---|---|---|---|---|
| 1000 | 75°21' | 1256 | 3 | 20 | 17 | 1 | 7 |
| 1050 | 74°31' | 1242 | 3 | 20 | 16 | 1 | 7 |
| 1100 | 73°40' | 1228 | 3 | 20 | 15 | 2 | 8 |
| 1150 | 72°47' | 1213 | 3 | 20 | 15 | 2 | 8 |
| 1200 | 71°53' | 1198 | 3 | 20 | 14 | 2 | 8 |
| 1250 | 70°58' | 1183 | 3 | 20 | 13 | 2 | 9 |
| 1300 | 70°02' | 1167 | 3 | 20 | 13 | 2 | 9 |
| 1350 | 69°03' | 1151 | 3 | 20 | 12 | 2 | 9 |
| 1400 | 68°04' | 1134 | 3 | 20 | 12 | 2 | 10 |
| 1450 | 67°02' | 1117 | 4 | 20 | 11 | 2 | 10 |
| 1500 | 65°58' | 1099 | 4 | 20 | 11 | 2 | 10 |

- **Propellant charges Type I, Large Angle (Indirect fire)**
- **The target elevation correction (*DCmi*) is negative when firing indirectly**
- **Range wind correction (*LWCm*) is opposite to the wind direction**
- **Crosswind correction (*CWCmi*) is opposite to the wind direction**
- **Elevation difference correction value (*ECm*) is positive**
- **Altitude correction value (*ACm*) is negative**

## A2_a Type II, Small angle

| *BRm* | *BEd* | *BEmi* | *DCmi* | *LWCm* | *CWCmi* | *ECm* | *ACm* |
|---|---|---|---|---|---|---|---|
| 1000 | 03°55' | 65 | 1 | 3 | 1 | 5 | 3 |
| 1050 | 04°14' | 71 | 1 | 4 | 1 | 5 | 4 |
| 1100 | 04°34' | 76 | 1 | 4 | 1 | 6 | 4 |
| 1150 | 04°54' | 82 | 1 | 4 | 2 | 6 | 4 |
| 1200 | 05°16' | 88 | 1 | 5 | 2 | 7 | 4 |
| 1250 | 05°38' | 94 | 1 | 5 | 2 | 8 | 5 |
| 1300 | 06°02' | 101 | 1 | 5 | 2 | 9 | 5 |
| 1350 | 06°27' | 107 | 1 | 6 | 2 | 13 | 5 |
| 1400 | 06°53' | 115 | 1 | 6 | 2 | 14 | 6 |
| 1450 | 07°20' | 122 | 2 | 6 | 2 | 15 | 6 |
| 1500 | 07°48' | 130 | 2 | 7 | 2 | 16 | 6 |

- **Propellant charges Type II, Small Angle (Direct fire)**
- **The target elevation correction (*DCmi*) is positive when firing directly**
- **Range wind correction (*LWCm*) is opposite to the wind direction**
- **Crosswind correction (*CWCmi*) is opposite to the wind direction**
- **Elevation difference correction value (*ECm*) is positive**
- **Altitude correction value (*ACm*) is negative**

## A2_b Type II, Large angle

| *BRm* | *BEd* | *BEmi* | *DCmi* | *LWCm* | *CWCmi* | *ECm* | *ACm* |
|---|---|---|---|---|---|---|---|
| 1000 | 77°58' | 1299 | 2 | 26 | 23 | 1 | 8 |
| 1050 | 77°18' | 1288 | 2 | 26 | 22 | 1 | 8 |
| 1100 | 76°38' | 1277 | 2 | 26 | 21 | 1 | 9 |
| 1150 | 75°57' | 1266 | 2 | 26 | 20 | 1 | 9 |
| 1200 | 75°16' | 1254 | 2 | 26 | 19 | 1 | 9 |
| 1250 | 74°33' | 1243 | 2 | 26 | 19 | 1 | 10 |
| 1300 | 73°50' | 1231 | 2 | 26 | 18 | 1 | 10 |
| 1350 | 73°07' | 1219 | 2 | 26 | 17 | 1 | 10 |
| 1400 | 72°22' | 1206 | 3 | 26 | 16 | 1 | 11 |
| 1450 | 71°37' | 1194 | 3 | 26 | 16 | 1 | 11 |
| 1500 | 70°50' | 1181 | 3 | 27 | 15 | 1 | 12 |

- **Propellant charges Type II, Large Angle (Indirect fire)**
- **The target elevation correction (*DCmi*) is negative when firing indirectly**
- **Range wind correction (*LWCm*) is opposite to the wind direction**
- **Crosswind correction (*CWCmi*) is opposite to the wind direction**
- **Elevation difference correction value (*ECm*) is positive**
- **Altitude correction value (*ACm*) is negative**

## A3_a Type III, Small angle

| *BRm* | *BEd* | *BEmi* | *DCmi* | *LWCm* | *CWCmi* | *ECm* | *ACm* |
|---|---|---|---|---|---|---|---|
| 1000 | 02°55' | 49 | 1 | 3 | 1 | 4 | 4 |
| 1050 | 03°10' | 53 | 1 | 3 | 1 | 4 | 4 |
| 1100 | 03°26' | 57 | 1 | 4 | 1 | 5 | 4 |
| 1150 | 03°42' | 62 | 1 | 4 | 2 | 6 | 5 |
| 1200 | 03°59' | 66 | 1 | 4 | 2 | 6 | 5 |
| 1250 | 04°17' | 71 | 1 | 4 | 2 | 7 | 5 |
| 1300 | 04°36' | 77 | 1 | 5 | 2 | 8 | 6 |
| 1350 | 04°55' | 82 | 1 | 5 | 2 | 9 | 6 |
| 1400 | 05°16' | 88 | 1 | 5 | 2 | 10 | 6 |
| 1450 | 05°37' | 94 | 1 | 6 | 2 | 10 | 7 |
| 1500 | 05°60' | 100 | 1 | 6 | 2 | 15 | 7 |

- **Propellant charges Type III, Small Angle (Direct fire)**
- **The target elevation correction (*DCmi*) is positive when firing directly**
- **Range wind correction (*LWCm*) is opposite to the wind direction**
- **Crosswind correction (*CWCmi*) is opposite to the wind direction**
- **Elevation difference correction value (*ECm*) is positive**
- **Altitude correction value (*ACm*) is negative**

## A3_b Type III, Large angle

| *BRm* | *BEd* | *BEmi* | *DCmi* | *LWCm* | *CWCmi* | *ECm* | *ACm* |
|---|---|---|---|---|---|---|---|
| 1000 | 78°54' | 1315 | 2 | 29 | 27 | 1 | 8 |
| 1050 | 78°17' | 1305 | 2 | 29 | 25 | 1 | 9 |
| 1100 | 77°41' | 1295 | 2 | 29 | 24 | 1 | 9 |
| 1150 | 77°03' | 1284 | 2 | 29 | 23 | 1 | 9 |
| 1200 | 76°26' | 1274 | 2 | 29 | 22 | 1 | 10 |
| 1250 | 75°47' | 1263 | 2 | 29 | 21 | 1 | 10 |
| 1300 | 75°09' | 1252 | 2 | 30 | 20 | 1 | 11 |
| 1350 | 74°29' | 1242 | 2 | 30 | 20 | 1 | 11 |
| 1400 | 73°49' | 1230 | 2 | 30 | 19 | 1 | 12 |
| 1450 | 73°09' | 1219 | 2 | 30 | 18 | 1 | 12 |
| 1500 | 72°28' | 1208 | 2 | 30 | 18 | 1 | 12 |

- **Propellant charges Type III, Large Angle (Indirect fire)**
- **The target elevation correction (*DCmi*) is negative when firing indirectly**
- **Range wind correction (*LWCm*) is opposite to the wind direction**
- **Crosswind correction (*CWCmi*) is opposite to the wind direction**
- **Elevation difference correction value (*ECm*) is positive**
- **Altitude correction value (*ACm*) is negative**

## A4 Sample

Simulating a real-world scenario: An officer positioned at an elevation of 500 meters aims to hit a target located 1300 meters horizontally away at an elevation of 550 meters using the maximum velocity indirect fire method. At this time, the officer has measured a downwind speed of +3 m/s and a crosswind speed of -5 m/s.

**Step1:** Reference range $x_{base}$ is 1300m,our elevation $h$ is 500m,elevation difference $\Delta z$ is 50m,,range wind speed $u_x$ is 3 m/s, crosswind speed $u_y$ is −5 m/s.

**Step2:** The officer selected Type III and indirect fire, corresponding to the A3_b artillery table. The base range was 1300m. We located the corresponding row and determined the other data.

**Step3:** Substituting the data , yields $\Delta\theta_{DC} = 0mi$ , $\Delta x_{LW} = -90m$ , $\Delta\varphi_{CW} = 100mi$ , $\Delta x_{EC} = 5m$ , $\Delta x_{AC} = -55m$

**Step4:** Summarize the elevation angle, azimuth angle, and actual range.

$$x_{final} = 1160m$$

$$\theta_{final} = 1274mi$$

$$\varphi_{final} = 100mi$$

# B Code

## Problem_1.m

```
function FIG=Problem_1(fig_i,fig_j,i_total,j_total)
if nargin < 1
    fig_i=1;fig_j=2;i_total=3;j_total=3;
end
v_0=450*sqrt(fig_i/i_total);
x_target=1200;
params.g=9.80665;
params.rou=1.225;
params.T=288.15;
params.m=5;
params.S=pi*0.055^2;
params.C_D=0.285;
params.z_0=500;
deg2rad=pi/180;
neg_range_fun = @(theta) -calculate_range(theta, v_0, params);
theta_search_interval_rad = [0, 90*deg2rad];
[theta_max_rad, neg_max_range] = fminbnd(neg_range_fun, ...
                                theta_search_interval_rad(1), ...
                                theta_search_interval_rad(2));
theta_max_deg = theta_max_rad / deg2rad;
max_range = -neg_max_range;
if max_range<x_target
    return
end
if fig_j==1
    fprintf('\n\n炮口初速: %.4fm/s\n', v_0);
    [D, M, ~] = deg2dms(theta_max_deg);
    fprintf('最大射程对应的仰角: %d°%2d"\n', D, M);
    fprintf('最大射程: %.2f 米\n', max_range);

    theta_deg_vec = 1:1:89;
    range_vec = zeros(size(theta_deg_vec));
    for i = 1:length(theta_deg_vec)
        range_vec(i) = calculate_range(theta_deg_vec(i) * deg2rad, v_0, params);
    end
    subplot(i_total,j_total,(fig_i-1)*j_total+fig_j);
    plot(theta_deg_vec, range_vec, '-', 'LineWidth', 2);
    hold on;

    plot(theta_max_deg, max_range, 'x', 'MarkerSize', 10, 'LineWidth', 2, 'MarkerFaceColor', 'r');
    title(['仰角与水平射程的关系(初速=',num2str(v_0),'m/s)'],'FontSize',8);
    label_text = sprintf('仰角(°)\n仰角 %d°%2d" 时对应最大射程 %.2fm', D, M, max_range);
    xlabel(label_text,'FontSize',8);
    ylabel('水平射程 (m)','FontSize',8);
    legend('射程曲线', '最大射程','FontSize',5);
    axis ([-5 95 -50 3500]);
    grid on;
    set(gca, 'LooseInset', [1,1,1,1]);
    hold off;
end
if fig_j==2
    theta_guess_interval_deg=[1,theta_max_deg];
elseif fig_j==3
    theta_guess_interval_deg=[theta_max_deg,89];
else
    return
end
theta_guess_interval_rad=theta_guess_interval_deg*deg2rad;
loss_fun=@(theta)calculate_loss(theta,v_0,x_target,params);
[theta_solution_rad,loss_at_solution]=fzero(loss_fun,theta_guess_interval_rad);
theta_solution_deg=theta_solution_rad/deg2rad;
fprintf('目标距离:%.1fm\n',x_target);
[D, M, ~] = deg2dms(theta_solution_deg);
fprintf('所需仰角:%d°%2d"\n', D, M);
if loss_at_solution>1e-3
    fprintf('最终误差(x_final-x_target):%.4fm\n',loss_at_solution);
end
Z0_optimal=[0;v_0*cos(theta_solution_rad);0;v_0*sin(theta_solution_rad)];
options_optimal=odeset('Events',@landing_event);
ode_fun_optimal=@(t,Z)main_model_ODE(t,Z,params);
[t_sol,Z_sol]=ode45(ode_fun_optimal,[0,1000],Z0_optimal,options_optimal);
turbo_15=turbo(15);
if fig_i==1
    smoke_color = turbo_15(2,:);
elseif fig_i==2
    smoke_color = turbo_15(9,:);
elseif fig_i==3
    smoke_color = turbo_15(10,:);
end
if fig_j==2
    FIG=plot(Z_sol(:,1),Z_sol(:,3),'-','LineWidth',2,'Color',smoke_color);
elseif fig_j==3
    FIG=plot(Z_sol(:,1),Z_sol(:,3),'--','LineWidth',2,'Color',smoke_color);
end
hold on;
xlabel('Horizontal Range x(m)','FontSize',12);
ylabel('Height Above Ground z(m)','FontSize',12);
axis ([-5 1250 -50 2600]);
set(gca, 'LooseInset', [1,1,1,1]);
grid on;
end
```

```
function range = calculate_range(theta, v_0,
        params)
    Z_0 = [0; v_0*cos(theta); 0; v_0*sin(theta)];
    options = odeset('Events', @landing_event);
    ode_fun = @(t, Z) main_model_ODE(t, Z,
        params);

    try
        [~, Z_sol, ~, Z_E] = ode45(ode_fun, [0,
            1000], Z_0, options);
        if isempty(Z_E)
            range = Z_sol(end, 1);
        else
            range = Z_E(1, 1);
        end
    catch
        range = -1;
    end
end
function[value,isterminal,direction]=land-
        ing_event(t,Z)
    value=Z(3);
    isterminal=1;
    direction=-1;
end
function loss=calculate_loss(theta,v_0,x_tar-
        get,params)




    x_0=0;
    z_0=0;
    x_t_0=v_0*cos(theta);
    z_t_0=v_0*sin(theta);
    Z_0=[x_0;x_t_0;z_0;z_t_0];



    options=odeset('Events',@landing_event);



        ode_fun=@(t,Z)main_model_ODE(t,Z,
        params);



    t_span=[0,100];

        [~,Z_sol,T_E,Z_E]=ode45(ode_fun,t_sp
        an,Z_0,options);



    if isempty(T_E)

        x_final=Z_sol(end,1);
        loss=x_final-x_target;
    else

        x_final=Z_E(1);
        loss=x_final-x_target;
    end
```

```
end
function dZ=main_model_ODE(t,Z,params)


    g=params.g;
    T=params.T-0.0065*(Z(3)+params.z_0);

    rou=params.rou*exp((-
            g*(Z(3)+params.z_0))/(287*T));
    miu=(1.485e-
            6*sqrt(0.972*T))/(1+110.4/(0.972*T));
    m=params.m;
    S=params.S;
    v=sqrt(Z(2)^2+Z(4)^2);
    Re=rou*0.055*2*v/miu;
    c_0=sqrt(1.4*8.31446*T/0.02897);
    Ma=v/c_0;

    if Ma<0.8
        G_M=166*Ma^3+3.29*Ma^2-10.9*Ma+20;
    elseif Ma>=0.8
        G_M=5+40*Ma^-3;
    end
    if Ma<1
        H_M=0.0239*Ma^3+0.212*Ma^2-
            0.074*Ma+1;
    elseif Ma>=1
        H_M=0.93+1/(3.5+Ma^5);
    end
    if Ma<1.5
        C_M=1.65+0.65*tanh(4*Ma-3.4);
    elseif Ma>=1.5
        C_M=2.18-0.13*tanh(0.9*Ma-2.7);
    end


        C_D=24*(1+0.15*Re^0.687)*H_M/Re
        +0.42*C_M/(1+(42500/Re^(1.16*C_M)
        )+(G_M/sqrt(Re)));


    dZ=zeros(4,1);

    dZ(1)=Z(2);
    dZ(2)=-v*Z(2)*(rou*C_D*S)/(2*m);
    dZ(3)=Z(4);
    dZ(4)=-v*Z(4)*(rou*C_D*S)/(2*m)-g;
end
function [D, M, S] = deg2dms(deg_decimal)
    if ~isnumeric(deg_decimal) || ~issca-
            lar(deg_decimal)
        error('输入必须是一个标量数值');
    end

    sign_val = sign(deg_decimal);
    deg_abs = abs(deg_decimal);

    D_abs = floor(deg_abs);
    M_decimal = (deg_abs - D_abs) * 60;
    M_abs = floor(M_decimal);
    S = (M_decimal - M_abs) * 60;


    D = sign_val * D_abs;
    M = M_abs;
end
```

## run_Problem_1.m

```
clc;clear;close all;
i_total=3;
j_total=3;



    subfolder = 'Figures_问题一';



    if ~isfolder(subfolder)
        mkdir(subfolder);
    end
text_L={ 'Small Angle Solution for propellant
        charges(Type I)', 'Large Angle Solution
        for propellant charges(Type I)';...
    'Small Angle Solution for propellant
        charges(Type II)', 'Large Angle Solution
        for propellant charges(Type II)';...
    'Small Angle Solution for propellant
        charges(Type III)', 'Large Angle
        Solution for propellant charges(Type
        III)'};
for i=1:i_total
    figure("Position",[100 100 400 350]);
    FIG=zeros(1,2);
    for j=2:j_total
        FIG(j-1)=Problem_1(i,j,i_total,j_total);
    end
    hold off;
    legend([FIG(1), FIG(2)], ...
      {text_L{i,1},text_L{i,2}}, ...
     'Location','northeast', 'FontSize', 8);
    set(gca, 'LooseInset', [1,1,1,1]);
    name=sprintf('%d号装药下的数据', i);
    full_path_top = fullfile(subfolder, name);
    print(gcf, full_path_top, '-djpeg', '-r600');
    print(gcf, full_path_top, '-dmeta');
    close all;
end
```

## Problem_2.m

```
function [theta_solution_nowind_deg,phi_solu-
        tion_deg,deta_X]=Problem_2(to-
        tal_params,fig_i,fig_j)
    theta_solution_nowind_deg = NaN;
    phi_solution_deg = NaN;
    deta_X = NaN;
if nargin < 1
    total_params = struct();
    total_params.i_total = 3;
    total_params.z_0 = 3500;
    total_params.v_wind_x = 0;
    total_params.v_wind_y = 0;
    total_params.x_target = 1200;
    total_params.z_target = 0;
    fig_i=3;fig_j=2;
end
i_total=total_params.i_total;
params.z_0=total_params.z_0;
params.v_wind_x=total_params.v_wind_x;
params.v_wind_y=total_params.v_wind_y;
v_0=450*sqrt(fig_i/i_total);
x_target=total_params.x_target;
z_target=total_params.z_target;
targets = [x_target, 0, z_target];
params.g=9.80665;
params.rou=1.225;
params.T=288.15;
params.m=5;
params.S=pi*0.055^2;
params.C_D=0.285;
params_zerotest=params;
params_zerotest.v_wind_x=0;
params_zerotest.v_wind_y=0;
params_zerotest.z_0=0;
deg2rad=pi/180;
neg_range_fun = @(theta) -calcu-
        late_range(theta, v_0, params_zerotest);
theta_search_interval_rad = [0, 90*deg2rad];
[theta_max_rad, neg_max_range] =
        fminbnd(neg_range_fun, ...
                theta_search_inter-
        val_rad(1), ...
                theta_search_inter-
        val_rad(2));
theta_max_deg = theta_max_rad / deg2rad;
max_range=-neg_max_range;
if max_range<x_target
    return
end
if fig_j==2
    theta_guess_interval_deg=[1,theta_max_deg];
elseif fig_j==3
    theta_guess_inter-
        val_deg=[theta_max_deg,89];
else
    return
end
theta_guess_interval_rad=theta_guess_inter-
        val_deg*deg2rad;
loss_fun=@(theta)calculate_loss(theta,v_0,x_tar-
        get,params_zerotest);
[theta_solution_rad,loss_at_solu-
        tion]=fzero(loss_fun,theta_guess_inter-
        val_rad);
theta_solution_nowind_deg=theta_solu-
        tion_rad/deg2rad;
```

```
if loss_at_solution>1e-3

end
theta_guess_rad=theta_solution_rad;
phi_guess_rad = -atan(params.v_wind_y / (v_0 +
        params.v_wind_x));

    initial_guess = [theta_guess_rad,
        phi_guess_rad];
    loss_fun_handle = @(angles) calcu-
        late_wind_loss(angles, v_0, targets,
        params);
    options = optimoptions('fsolve', 'Display', 'off',
        'Algorithm', 'levenberg-marquardt');


    [solution, fval] = fsolve(loss_fun_handle, ini-
        tial_guess, options);
    if sum(fval.^2) > 1e-3
        return
    end
    theta_solution_rad = solution(1);
    phi_solution_rad = solution(2);




x_real=calculate_range(theta_solution_rad, v_0,
        params_zerotest);
deta_X=x_real-x_target;
[D, M, ~] = deg2dms(theta_solu-
        tion_nowind_deg);
str_theta_solu-
        tion_nowind_deg=sprintf('%d°%d''', D,
        M);
phi_solution_deg=phi_solution_rad * 180 / pi;
[D, M, ~] = deg2dms(phi_solution_deg);
str_phi_solution_deg=sprintf('%d°%d''', D, M);
end
function range = calculate_range(theta, v_0,
        params)
    x_0=0;
    y_0=0;
    z_0=0;
    x_t_0=v_0*cos(theta)*1;
    y_t_0=v_0*cos(theta)*0;
    z_t_0=v_0*sin(theta);
    Z_0=[x_0;x_t_0;y_0;y_t_0;z_0;z_t_0];
    options = odeset('Events', @landing_event);
    ode_fun = @(t, Z) main_model_ODE(t, Z,
        params);

    try
        [~, Z_sol, ~, Z_E] = ode45(ode_fun, [0,
            1000], Z_0, options);
        if isempty(Z_E)
            range = Z_sol(end, 1);
        else
            range = Z_E(1, 1);
        end
    catch
        range = -1;
    end
end
function[value,isterminal,direction]=land-
        ing_event(t,Z)
    value=Z(5);
    isterminal=1;
    direction=-1;
end
function[value,isterminal,direction]=tar-
        get_height_event(t,Z,z_target)
    value=Z(5)-z_target;
    isterminal=1;
    direction=-1;
end
function loss=calculate_loss(theta,v_0,x_tar-
        get,params)



    x_0=0;
    y_0=0;
    z_0=0;
    x_t_0=v_0*cos(theta)*1;
    y_t_0=v_0*cos(theta)*0;
    z_t_0=v_0*sin(theta);
    Z_0=[x_0;x_t_0;y_0;y_t_0;z_0;z_t_0];



    options=odeset('Events',@landing_event);



        ode_fun=@(t,Z)main_model_ODE(t,Z,
        params);



    t_span=[0,100];

        [~,Z_sol,T_E,Z_E]=ode45(ode_fun,t_sp
        an,Z_0,options);



    if isempty(T_E)

        x_final=Z_sol(end,1);
        loss=x_final-x_target;
    else

        x_final=Z_E(1);
        loss=x_final-x_target;
    end
end
function loss=calculate_wind_loss(an-
        gles,v_0,targets,params)



    theta=angles(1);
    phi=angles(2);

    x_0=0;
    y_0=0;
    z_0=0;
    x_t_0=v_0*cos(theta)*cos(phi);
    y_t_0=v_0*cos(theta)*sin(phi);
    z_t_0=v_0*sin(theta);
```

```
    Z_0=[x_0;x_t_0;y_0;y_t_0;z_0;z_t_0];

    x_target = targets(1);
    y_target = targets(2);
    z_target = targets(3);
    event_fun = @(t, Z) target_height_event(t, Z,
        z_target);
    options = odeset('Events', event_fun, 'RelTol',
        1e-6);



        ode_fun=@(t,Z)main_model_ODE(t,Z,
        params);

    try
      t_span=[0,100];

        [~,Z_sol,T_E,Z_E]=ode45(ode_fun,t_sp
        an,Z_0,options);

      if isempty(Z_E)
        loss = [1e6; 1e6];
      else
        x_final = Z_E(1);
        y_final = Z_E(3);


        loss = [x_final - x_target; y_final - y_tar-
          get];
      end
    catch
      loss = [1e6; 1e6];
    end
end
function dZ=main_model_ODE(t,Z,params)

    g=params.g;
    T=params.T-0.0065*(Z(5)+params.z_0);

    rou=params.rou*exp((-
        g*(Z(5)+params.z_0))/(287*T));
    miu=(1.485e-
        6*sqrt(0.972*T))/(1+110.4/(0.972*T));
    m=params.m;
    S=params.S;
    v_wind_x=params.v_wind_x;
    v_wind_y=params.v_wind_y;
    v=sqrt((Z(2)-v_wind_x)^2+(Z(4)-
        v_wind_y)^2+Z(6)^2);
    Re=rou*0.055*2*v/miu;
    c_0=sqrt(1.4*8.31446*T/0.02897);
    Ma=v/c_0;
```

```
    if Ma<0.8
      G_M=166*Ma^3+3.29*Ma^2-10.9*Ma+20;
    elseif Ma>=0.8
      G_M=5+40*Ma^-3;
    end
    if Ma<1
      H_M=0.0239*Ma^3+0.212*Ma^2-
          0.074*Ma+1;
    elseif Ma>=1
      H_M=0.93+1/(3.5+Ma^5);
    end
    if Ma<1.5
      C_M=1.65+0.65*tanh(4*Ma-3.4);
    elseif Ma>=1.5
      C_M=2.18-0.13*tanh(0.9*Ma-2.7);
    end



        C_D=24*(1+0.15*Re^0.687)*H_M/Re
        +0.42*C_M/(1+(42500/Re^(1.16*C_M)
        )+(G_M/sqrt(Re)));


    dZ=zeros(4,1);

    dZ(1)=Z(2);
    dZ(2)=-v*(Z(2)-
        v_wind_x)*(rou*C_D*S)/(2*m);
    dZ(3)=Z(4);
    dZ(4)=-v*(Z(4)-
        v_wind_y)*(rou*C_D*S)/(2*m);
    dZ(5)=Z(6);
    dZ(6)=-v*Z(6)*(rou*C_D*S)/(2*m)-g;
end
function [D, M, S] = deg2dms(deg_decimal)
    if ~isnumeric(deg_decimal) || ~issca-
        lar(deg_decimal)
      error('输入必须是一个标量数值');
    end

    sign_val = sign(deg_decimal);
    deg_abs = abs(deg_decimal);

    D_abs = floor(deg_abs);
    M_decimal = (deg_abs - D_abs) * 60;
    M_abs = floor(M_decimal);
    S = (M_decimal - M_abs) * 60;


    D = sign_val * D_abs;
    M = M_abs;
end
```

## run_Problem_2.m

```
clc;clear;close all;
if isempty(gcp('nocreate'))
    parpool('Processes');
    pool = gcp('nocreate');
    fprintf('已启动并行池，工作线程数: %d\n',
        pool.NumWorkers);
end
    total_params = struct();
    total_params.i_total = 3;
    total_params.z_0 = 0;
    total_params.v_wind_x = 0;
    total_params.v_wind_y = 0;
    total_params.x_target = 1000;
    total_params.z_target = 0;
    fig_i=1;fig_j=2;

    Basic_Table=zeros(total_params.i_total,3,11);
    TargetX_Table=zeros(total_params.i_to-
        tal,3,11);
    WindX_Table=zeros(total_params.i_to-
        tal,3,11);
    WindY_Table=zeros(total_params.i_to-
        tal,3,11);
    Height_Table=zeros(total_params.i_to-
        tal,3,11);
    Altitude_Table=zeros(total_params.i_to-
        tal,3,11);
    parfor fig_i=1:total_params.i_total
        temp_slice_Basic = zeros(3, 11);
        temp_slice_TargetX = zeros(3, 11);
        temp_slice_WindX = zeros(3, 11);
        temp_slice_WindY = zeros(3, 11);
        temp_slice_Height = zeros(3, 11);
        temp_slice_Altitude = zeros(3, 11);
        local_params = total_params;
        for fig_j=2:3
            for x_target=1000:50:1500
                local_params.x_target=x_target;
                [theta_deg,~,~]=Problem_2(lo-
            cal_params,fig_i,fig_j);
                temp_slice_Basic(fig_j,(x_target-
            950)/50)=theta_deg;
                Y_fix_target_x=zeros(1,10);
                Y_fix_wind_x=zeros(1,33);
                Y_fix_wind_y=zeros(1,33);
                Y_fix_height=zeros(1,35);
                Y_fix_altitude=zeros(1,71);
                for target_x=0:5:45
                    count=(target_x+5)/5;
                    local_params.x_target=x_target+tar-
            get_x;
                    [theta_deg,~,~]=Problem_2(lo-
            cal_params,fig_i,fig_j);
                    Y_fix_target_x(count)=theta_deg;
                    local_params.x_target=x_target;
                end
                Result_fix_tar-
            get_x=fitlm(0:5:45,Y_fix_target_x);
                if Result_fix_target_x.Rsquared.Ordi-
            nary>0.95
                    fix_target_x=Result_fix_tar-
            get_x.Coefficients.Estimate(2);
                else
                    fix_target_x=0;
                end
                temp_slice_TargetX(fig_j,(x_target-
            950)/50)=fix_target_x;
                if Result_fix_tar-
            get_x.Rsquared.Ordinary > 0.99998
                    x_str=sprintf('Depth Correc-
            tion(m)');
                    y_str=sprintf('Elevation Angle
            Correction(°)');
                    filename_str = sprintf('Fit_Tar-
            getX_Charge%d_Mode%d_Range%d',
            fig_i, fig_j, x_target);
                    plot_fit_results(0:5:45,
            Y_fix_target_x, Result_fix_target_x,
            x_str,y_str, filename_str);
                end
            for wind=-8:0.5:8
                count=(wind+8.5)*2;
                local_params.v_wind_x=wind;
                [~,~,deta_X]=Problem_2(lo-
            cal_params,fig_i,fig_j);
                Y_fix_wind_x(count)=deta_X;
                local_params.v_wind_x=0;
                local_params.v_wind_y=wind;
                [~,phi_deg,~]=Problem_2(lo-
            cal_params,fig_i,fig_j);
                Y_fix_wind_y(count)=phi_deg;
                local_params.v_wind_y=0;
            end
            Result_fix_wind_x=fitlm(-
            8:0.5:8,Y_fix_wind_x);
            Result_fix_wind_y=fitlm(-
            8:0.5:8,Y_fix_wind_y);
            if Result_fix_wind_x.Rsquared.Ordi-
            nary>0.95
                fix_wind_x=Result_fix_wind_x.Co-
            efficients.Estimate(2);
            else
                fix_wind_x=0;
            end
            if Result_fix_wind_y.Rsquared.Ordi-
            nary>0.95
                fix_wind_y=Result_fix_wind_y.Co-
            efficients.Estimate(2);
            else
                fix_wind_y=0;
            end
            temp_slice_WindX(fig_j,(x_target-
            950)/50)=fix_wind_x;
            temp_slice_WindY(fig_j,(x_target-
            950)/50)=fix_wind_y;
                if Re-
            sult_fix_wind_x.Rsquared.Ordi-
            nary>0.9996
                    x_str=sprintf('Range Wind Ve-
            locity(m/s)');
                    y_str=sprintf('Range Correc-
            tion(m)');
                    filename_str =
            sprintf('Fit_WindX_Charge%d_Mode%
            d_Range%d', fig_i, fig_j, x_target);
                    plot_fit_results(-8:0.5:8,
```

```
Y_fix_wind_x, Result_fix_wind_x,
x_str,y_str, filename_str);
        end

        if Re-
sult_fix_wind_y.Rsquared.Ordi-
nary>0.99999988
            x_str=sprintf('Cross Wind Ve-
locity(m/s)');
            y_str=sprintf('Azimuth Correc-
tion(°)');
            filename_str =
sprintf('Fit_WindY_Charge%d_Mode%
d_Range%d', fig_i, fig_j, x_target);
            plot_fit_results(-8:0.5:8,
Y_fix_wind_y, Result_fix_wind_y,
x_str,y_str, filename_str);
        end
  for height=-850:50:850
     count=(height+900)/50;
     local_params.z_target=height;
     [~,~,deta_X]=Problem_2(lo-
cal_params,fig_i,fig_j);
     Y_fix_height(count)=deta_X;
     local_params.z_target=0;
  end
  Result_fix_height=fitlm(-
850:50:850,Y_fix_height);
  if Result_fix_height.Rsquared.Ordi-
nary
     fix_height=Result_fix_height.Coef-
ficients.Estimate(2);
  else
     fix_height=0;
  end
  temp_slice_Height(fig_j,(x_target-
950)/50)=fix_height;

        if Result_fix_height.Rsquared.Or-
dinary>0.95
            x_str=sprintf('Target Elevation
Difference(m)');
            y_str=sprintf('Range Correc-
tion(m)');
            filename_str =
sprintf('Fit_Height_Charge%d_Mode%
d_Range%d', fig_i, fig_j, x_target);

        end
  for altitude=0:50:3500
     count=(altitude+50)/50;
     local_params.z_0=altitude;
     [~,~,deta_X]=Problem_2(lo-
cal_params,fig_i,fig_j);
     Y_fix_altitude(count)=deta_X;
     local_params.z_0=0;
  end
  Result_fix_alti-
tude=fitlm(0:50:3500,Y_fix_altitude);
  if Result_fix_altitude.Rsquared.Ordi-
nary>0.985
     fix_altitude=Result_fix_altitude.Co-
efficients.Estimate(2);
  else
     fix_altitude=0;
  end
  temp_slice_Altitude(fig_j,(x_target-
950)/50)=fix_altitude;

        if Result_fix_alti-
tude.Rsquared.Ordinary>0.95
            x_str=sprintf('Current Alti-
tude(m)');
            y_str=sprintf('Range Correc-
tion(m)');
            filename_str = sprintf('Fit_Alti-
tude_Charge%d_Mode%d_Range%d',
fig_i, fig_j, x_target);

        end
    end
  end
  Basic_Table(fig_i, :, :) = temp_slice_Basic;
  TargetX_Table(fig_i, :, :) = temp_slice_Tar-
    getX;
  WindX_Table(fig_i, :, :) =
    temp_slice_WindX;
  WindY_Table(fig_i, :, :) =
    temp_slice_WindY;
  Height_Table(fig_i, :, :) =
    temp_slice_Height;
  Altitude_Table(fig_i, :, :) = temp_slice_Al-
    titude;
end
excel_file = '火炮射表.xlsx';
for m = 1:total_params.i_total
  for n = 2:3


    if m == 1
      charge_name = 'I号装药';
    elseif m == 2
      charge_name = 'II号装药';
    else
      charge_name = 'III号装药';
    end

    if n == 2
      fire_mode_name = '平射';
      TargetX_name='平射时修正值为正';
      mode_name = '平射(直接瞄准射击)';
    else
      fire_mode_name = '曲射';
      TargetX_name='曲射时修正值为负';
      mode_name = '曲射(超越射击)';
    end

    sheet_name = [charge_name, '_',
      fire_mode_name];
    num_rows = size(Basic_Table, 3);
    output_cell = cell(num_rows + 2, 8);

    output_cell{1, 1} = charge_name;
    output_cell{1, 2} = mode_name;
    output_cell{1, 3} = '';
    output_cell{1, 4} = TargetX_name;
    output_cell{1, 5} = '纵风修正与风向相反';
    output_cell{1, 6} = '横风修正与风向相反';
    output_cell{1, 7} = '高程差修正为正值';
    output_cell{1, 8} = '海拔修正为负值';
    output_cell{2, 1} = '基准射程(m)';
    output_cell{2, 2} = '基准仰角';
```

```
        output_cell{2, 3} = '基准仰角(密位)';
        output_cell{2, 4} = '每10m纵深修正(密
            位)';
        output_cell{2, 5} = '每1m/s纵风修正(m)';
        output_cell{2, 6} = '每1m/s横风修正(密
            位)';
        output_cell{2, 7} = '每10m高程差修正(m)';
        output_cell{2, 8} = '每100m海拔修正(m)';


        for i = 1:num_rows
            output_cell{i + 2, 1} = sprintf('%.0f', 950
              + 50 * i);
            [D, M] = deg2dms(Basic_Table(m, n, i));
            output_cell{i + 2, 2} =
              sprintf('%02d°%02d''', D, M);
            output_cell{i + 2, 3} = sprintf('%.0f',
              Basic_Table(m, n, i) / 0.06);
            output_cell{i + 2, 4} = sprintf('%.0f',
              abs(10 * TargetX_Table(m, n, i) /
              0.06));
            output_cell{i + 2, 5} = sprintf('%.0f',
              abs(WindX_Table(m, n, i)));
            output_cell{i + 2, 6} = sprintf('%.0f',
              abs(WindY_Table(m, n, i) / 0.06));
            output_cell{i + 2, 7} = sprintf('%.0f', 10 *
              Height_Table(m, n, i));
            output_cell{i + 2, 8} = sprintf('%.0f',
              abs(100 * Altitude_Table(m, n, i)));
        end
        writecell(output_cell, excel_file, 'Sheet',
            sheet_name, 'WriteMode', 'over-
            writesheet');

        fprintf('已成功将数据写入工作表: %s\n',
            sheet_name);
    end
end
function [D, M] = deg2dms(deg_decimal)
    if ~isnumeric(deg_decimal) || ~issca-
            lar(deg_decimal)
        error('输入必须是一个标量数值');
    end

    sign_val = sign(deg_decimal);
    deg_abs = abs(deg_decimal);

    D_abs = floor(deg_abs);
    M_decimal = (deg_abs - D_abs) * 60;
    M_abs = floor(M_decimal);
    S = (M_decimal - M_abs) * 60;
    if S>=30
        M_abs=M_abs+1;
    end


    D = sign_val * D_abs;
    M = M_abs;
end
function plot_fit_results(x_data, y_data,
            fit_model, x_str,y_str, filename_str)
    fig = figure('Visible', 'off','Position',[100 100
            400 350]);
    display_threshold = 30;
    display_interval = 5;
    if numel(x_data) > display_threshold
        step = display_interval;
    else
        step = 1;
    end
    plot(x_data(1:step:end), y_data(1:step:end), '^',
            'LineWidth', 2, 'DisplayName', 'Actual
            data points');

    hold on;
    x_range = linspace(min(x_data), max(x_data),
            100);
    y_fit = predict(fit_model, x_range');
    plot(x_range, y_fit, '-', 'LineWidth', 2, 'Dis-
            playName', 'Linear fitting line');
    r_squared = fit_model.Rsquared.Ordinary;
    legend_text = sprintf('R^2 = %.9f', r_squared);
    plot(NaN, NaN,'LineStyle', 'none', 'Display-
            Name',legend_text);

    legend('show','Location','northeast');
    xlabel(x_str, 'FontSize', 12);
    ylabel(y_str, 'FontSize', 12);
    grid on;
    hold off;



    subfolder = 'Figures_拟合曲线';


    if ~isfolder(subfolder)
        mkdir(subfolder);
    end
    full_path_top = fullfile(subfolder, file-
            name_str);
    print(gcf, full_path_top, '-djpeg', '-r600');
    print(gcf, full_path_top, '-dmeta');
    close(fig);
end
```