Problem Chosen	2024 ShuWei Cup Summary Sheet	Team Control Number
C		2024092929266

Research on Deep Space Navigation Accuracy Improvement Based on Pulsar Timing Noise and Atmospheric Delay Modeling

Summary

With the development of pulsar timing, pulsars have become a core tool in deep space navigation and time standards. Therefore, solving the issues of pulsar timing noise and atmospheric delays has become a key challenge in research. In this context, this paper analyzes solutions to these noise and delay problems through modeling.

For question 1, we analyzed pulsar timing noise, considering factors like observation frequency, data span, and noise fluctuations, and calculated the RMS value. The initial power spectrum model had an 81% goodness-of-fit. To improve accuracy, we used an autoregressive (AR) model, adjusting parameters and applying differencing techniques. The final model achieved a 99% goodness-of-fit, significantly improving the accuracy of pulsar timing noise modeling.

For question 2, pulsar timing noise, mainly caused by red noise (1/f noise), is stronger at low frequencies and decreases with higher frequencies. The noise intensity is proportional to the RMS value. To improve predictions, we applied detrending and differencing for stationarity. The AR model was used for short-term predictions, while the ARIMA model handled long-term forecasting. We determined the optimal p and q parameters through ACF and PACF analysis and found both models provide reliable predictions for pulsar timing noise.

For question 3, refractive delay, caused by variations in the atmospheric refractive index, is split into dry and wet delays. Wet delay is negligible at high frequencies and is excluded. Using the Herring mapping function, we extrapolated zenith delay to arbitrary elevation angles. Path integration revealed that refractive delay decreases significantly with increasing elevation angle, following an exponential decay trend.

For question 4 , to reduce errors in the Herring mapping function at low elevation angles, we introduced the GMF mapping function. Using atmospheric stratification and trapezoidal integration, we accurately calculated total delay and applied frequency corrections. The improved model showed higher accuracy and reduced errors. Sensitivity analysis identified elevation angle, frequency, and atmospheric pressure as key factors, and simulations under various conditions confirmed the model's reliability and universality.

Key word: AR ; ARIMA; Herring ; Sasstamoinen ; GMF

Content

1.Introduction

1.1 Background

Pulsars, rapidly rotating neutron stars, are essential for deep space navigation and atomic timekeeping. However, pulsar timing noise and atmospheric delays affect time signals. Pulsar timing noise, typically red noise, causes discrepancies between predicted and actual pulse arrival times, and is present in most pulsars. Atmospheric delay, especially at low elevation angles, also impacts timing accuracy by slowing electromagnetic waves. Addressing both timing noise and atmospheric delay is critical for improving pulsar time accuracy.

1.2 Work

Question 1: Consider using a functional model to simulate pulsar timing noise, aiming for a model fit of 95% or higher. It is generally assumed that the intensity of red noise is proportional to the RMS value, though not exactly equal.

Question 2: Consider making short-term (ranging from a few days to one month) and long-term (ranging from several months to a few years) forecasts of the future trend of the pulsar timing noise shown in the given figure. The required data for forecasting validation can also be found in Attachment 1.

Question 3: Consider modeling the refractive time delay for radio observation frequencies above 20 GHz, ensuring that the zenith delay does not exceed 7.69 nanoseconds. Relevant parameters and calculation processes can refer to Chapter 6.1 of the "Space-Time Reference Systems" document.

Question 4: Consider modeling the atmospheric delay for observations with small elevation angles (10 degrees or less) to improve the accuracy of the Time of Arrival (TOA). Please provide your model and describe the considerations and achievable goals.

2.Problem analysis

2.1 Data analysis

By analyzing the data provided in the appendix, we have obtained the characteristics of pulsar timing noise[1], including the observation frequency of the pulsar, the root mean square (RMS) value of the noise, the time span of the data, and the size of noise fluctuations in each observation period. We have integrated these characteristics through relevant literature and pulsar observation data sources to better understand the nature of pulsar timing noise and to provide a basis for modeling and prediction.

2.2 Analysis of question one

Pulsar timing noise is typically caused by red noise, which is also known as 1/f noise and has higher power in the low-frequency region. As the frequency increases, the intensity of the noise decreases. The hint provided indicates that red noise is proportional to the root mean square (RMS) value. In pulsar timing[2], the RMS value represents the magnitude of the noise fluctuations, affecting the accuracy and reliability of the signal. To improve the predictive capability of the model, we need to preprocess the data through detrending and differencing to ensure that the data meets the stationarity assumption.

2.3 Analysis of question two

For predicting future trends in pulsar timing noise, particularly for short-term (a few days to a month) and long-term (several months to a few years) trends, we used autoregressive (AR) models and ARIMA models for short-term and long-term predictions respectively.

This approach ensures that the models accurately predict both short-term and long-term trends in pulsar timing noise, thereby enhancing the overall prediction performance and reliability.

2.4 Analysis of question three

In radio astronomy observations, refraction delay is an important influencing factor, especially when the atmospheric propagation path[6] is long, as the refraction delay increases significantly. When the elevation angle is low, the path length the signal travels through the atmosphere increases, causing the refraction delay to be more significant compared to the zenith direction.The objective of this problem is to calculate the refraction delay at radio observation frequencies above 20 GHz.

2.5 Analysis of question four

Under low elevation angles, the signal travels a longer path through the atmosphere, amplifying the errors in dry delay and wet delay . Atmospheric delay significantly affects the accuracy of the TOA . The main sources of atmospheric delay include the refractive effects of the troposphere and ionosphere. The Saastamoinen model provided in the problem shows poor performance under low elevation angles, so this question requires finding a better method to improve the accuracy of TOA.

2.6 Framework of the paper

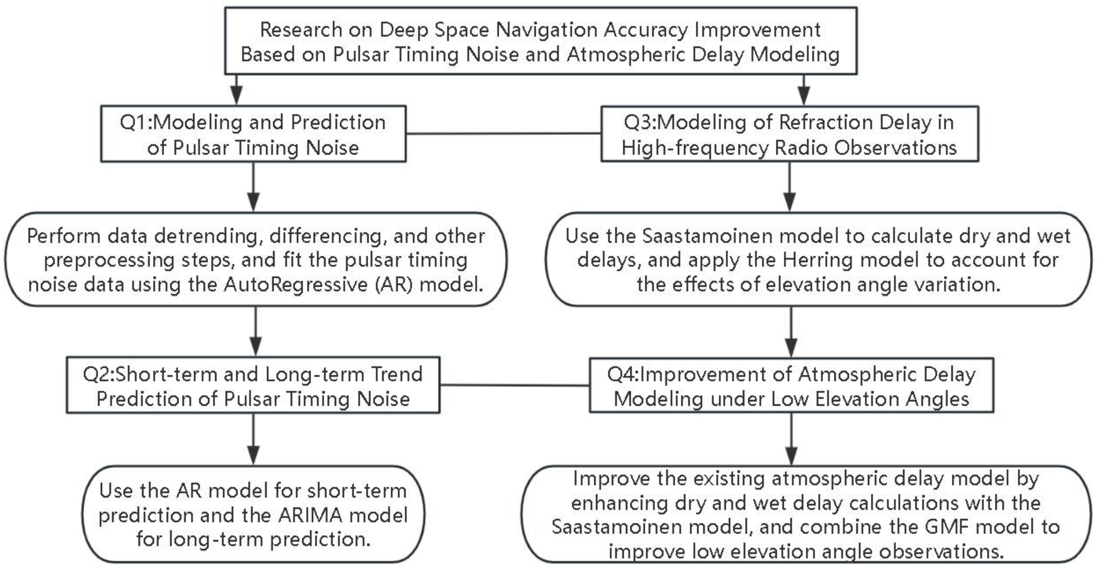

Research on Deep Space Navigation Accuracy Improvement Based on Pulsar Timing Noise and Atmospheric Delay Modeling
Q1:Modeling and Prediction of Pulsar Timing Noise
Q3:Modeling of Refraction Delay in High-frequency Radio Observations
Perform data detrending, differencing, and other preprocessing steps, and fit the pulsar timing noise data using the AutoRegressive (AR) model.
Use the Saastamoinen model to calculate dry and wet delays, and apply the Herring model to account for the effects of elevation angle variation.
Q2:Short-term and Long-term Trend Prediction of Pulsar Timing Noise
Q4:Improvement of Atmospheric Delay Modeling under Low Elevation Angles
Use the AR model for short-term prediction and the ARIMA model for long-term prediction.
Improve the existing atmospheric delay model by enhancing dry and wet delay calculations with the Saastamoinen model, and combine the GMF model to improve low elevation angle observations.

Figure 1: Framework of the paper

3.Symbol and Assumptions

3.1 Symbol Description

No.	Symbol	Symbol meaning
1	L	Lag operator
2	σ_{TN}	*RMS* red noise component
3	A_{red}	Amplitude of red noise
4	γ_{red}	Spectral index
5	f_{yr}	Unit frequency
6	X_t	The value at the current

		time point t
7	$\phi_0, \phi_1, \ldots, \phi_p$	Model parameters
8	ϵ_t	Random error term
9	d	Number of differencing
10	p	Lag order of past data
11	q	Lag order of past errors
12	$\theta_1, \theta_2, \ldots, \theta_q$	Moving average coefficient

3.2 Fundamental assumptions

Assumption 1: The pulsar timing noise is caused by red noise (1/f noise), the noise strength is inversely proportional to the frequency, and the power spectrum of the noise is stronger in the low-frequency range and attenuates as the frequency increases.

Assumption 2: The autoregressive (AR) model and the ARIMA model can adequately capture the time dependence and noise characteristics in the data.

Assumption 3: The model errors are independently and identically distributed and follow a normal distribution.

Assumption 4: There may be seasonal or cyclical variations in the data, especially in long-term trend forecasting.

Assumption 5: The data is stationary after being processed to achieve stationarity.

4.Model

4.1 Model establishment and solution of question 1

4.1.1Power spectrum model

Pulsar timing noise, as a significant factor affecting the accuracy of pulsar observations, has a considerable impact on pulsar time measurement and deep space navigation. Based on the characteristics of red noise, it is typically assumed that the intensity of red noise is proportional to the root mean square (RMS) value. We first propose a linear model-based pulsar timing noise modeling method, and then simulate the pulsar timing noise using a function model, fitting the model with relevant data to analyze the impact of noise on pulsar timing.

The intensity of red noise is usually related to the pulsar's rotation period and observation time, with the RMS (root mean square) value serving as an important indicator of red noise intensity. Initially, we assume that the intensity of pulsar timing noise is linearly related to the RMS value, meaning that the relationship between the RMS value and pulsar timing noise intensity can be represented by the following function model:

$$RMS = k \cdot \sigma_{TN} + b \tag{1}$$

Where, RMS is the root mean square residual, σ_{TN} is the RMS red noise component, k is the scaling factor, and b is the offset.

According to the power spectrum model provided in the problem, the frequency response of pulsar timing noise can be expressed by the following formula (2):

$$P(f) = A_{red}^2 \left(\frac{f}{f_{yr}} \right)^{\gamma_{red}} \tag{2}$$

Where, A_{red} is the amplitude of the red noise, $f_{yr} = 1yr^{-1}$ is the unit frequency (i.e., 1-year frequency), and , γ_{red} is the spectral index. Based on this formula, the power spectral density of the red noise is calculated. Integrating the above equation gives (3):

$$\sigma_{TN,2} = 2.05\text{ns} \cdot (1-\gamma_{\text{red}})^{-1/2} \cdot \left(\frac{A_{\text{red}}}{3\times10^{-3}\,\mu\text{s}\,\text{yr}^{1/2}} \right) \cdot T_{\text{yr}}^{(1-\gamma_{\text{red}})/2} \tag{3}$$

Based on the given RMS data, power spectrum model, and theoretical calculation methods, we calculate $\sigma_{TN,2}$ in the model. Then, we use the least squares method to fit the pulsar timing noise, and the resulting graph is shown in the following figure 2.

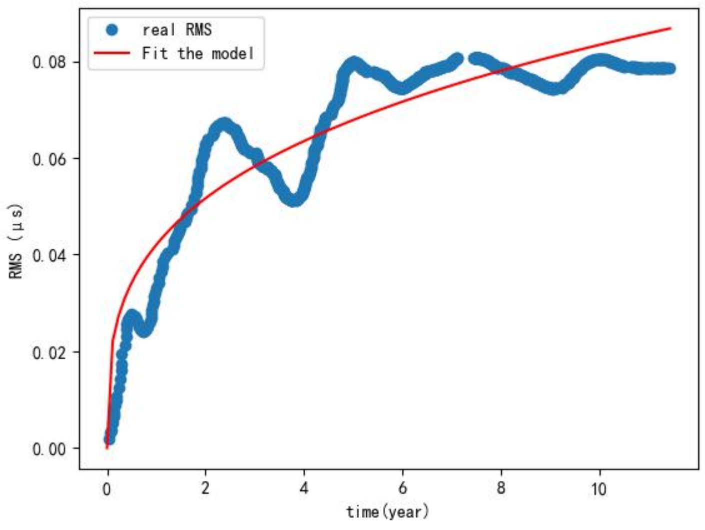


Figure 2: Power spectrum model fitting results

From the results in the above figure, it can be seen that the fitting model performs well in capturing the long-term variation trend of pulsar timing noise, but the goodness of fit is only 81%, which does not meet the requirements of the problem. Therefore, we proceed to change the model.

4.1.2 Autoregressive model establishment

Considering that pulsar timing noise exhibits time series characteristics, we next establish an autoregressive model (AR model, widely used for handling time series data), which can capture the time dependencies and trends in time series data. If the pulsar timing noise itself has time series characteristics, using an autoregressive model for fitting can better describe the structural patterns in the data.

The autoregressive model (AR model)[3]is a linear regression model where the current time point's data depends on the previous time points' data. The basic form of the AR model is as follows (4):

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \ldots + \phi_p X_{t-p} + \epsilon_t \tag{4}$$

Where, X_t is the value at the current time point t, $\phi_1, \phi_2, \ldots, \phi_p$ is the model parameter representing the influence of the past p time points, and ϵ_t is the white noise term representing the uncaptured random fluctuations.

Thus, for the pulsar timing noise data, we can use the AR model to model its time series characteristics. First, we calculate the root mean square (RMS) value of the residuals to evaluate the noise characteristics of the model. In summary, the steps for fitting the pulsar timing error data using the autoregressive model (AR model) are as follows.

Step1: Assume the time series data is y_t, where $t = 1, 2, \ldots, T$ represents each time point in the series, and y_t represents the timing error (PT-TT) observed at time t.

Step2: The autoregressive model assumes that the data at the current time point is a linear combination of the data from the previous time points. For an AR model of order p, its mathematical formula is (5):

$$y_t = \phi_0 + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p y_{t-p} + \epsilon_t \tag{5}$$

Where, y_t is the timing error value at the current time t, $\phi_0, \phi_1, \ldots, \phi_p$ is the model parameter, ϵ_t is the random error term, which is typically assumed to be white noise, satisfying $E[\epsilon_t] = 0$ and $\mathrm{Var}(\epsilon_t) = \sigma^2$.

Step3: The optimal model order p is selected using the Akaike Information Criterion (AIC) statistical method. The model is fitted using the least squares method to estimate the model parameters $\phi_0, \phi_1, \ldots, \phi_p$.

Step4: Minimize the residual sum of squares (RSS): $RSS = \sum_{t=p+1}^{T} (y_t - \hat{y}_t)^2$.

Where $\hat{y}_t$ is the predicted value obtained from the fitted model:

$$\hat{y}_t = \hat{\phi}_0 + \hat{\phi}_1 y_{t-1} + \hat{\phi}_2 y_{t-2} + \ldots + \hat{\phi}_p y_{t-p}$$

Minimizing the RSS ensures that the parameter estimates $(\hat{\phi}_i)(i = 0,1,\ldots,p)$ are optimal.

Step5: The model fitting accuracy is measured through the calculation of R^2 , which is defined as:

$$R^2 = 1 - \frac{RSS}{TSS}$$

On one hand, TSS (total sum of squares) represents the sum of the squared deviations of the original data from the mean value: $TSS = \sum_{t=p+1}^{T} (y_t - \bar{y})^2$, where $\bar{y}$ is the mean value of the data y_t . On the other hand, RSS (residual sum of squares) represents the sum of squared differences between the model's predicted values and the actual observed values: $RSS = \sum_{t=p+1}^{T} (y_t - \hat{y}_t)^2$.

4.1.3 Autoregressive model solution

First, we will display the calculation process through the table 1 below.

Table 1: Autoregressive Model Solution Process

Autoregressive model solution algorithm
➢ Step1 Data preprocessing: Identify missing values, outliers, and calculate RMS;
➢ Step2 Define the order p of the AR model;
➢ Step3 Use the AutoReg model to fit the data, generating the autoregressive coefficients and model parameters;
➢ Step4 Print the AR model's coefficients, standard error statistics, and other related information to verify the fitting results.

Based on this calculation method, the fitting results are shown below:

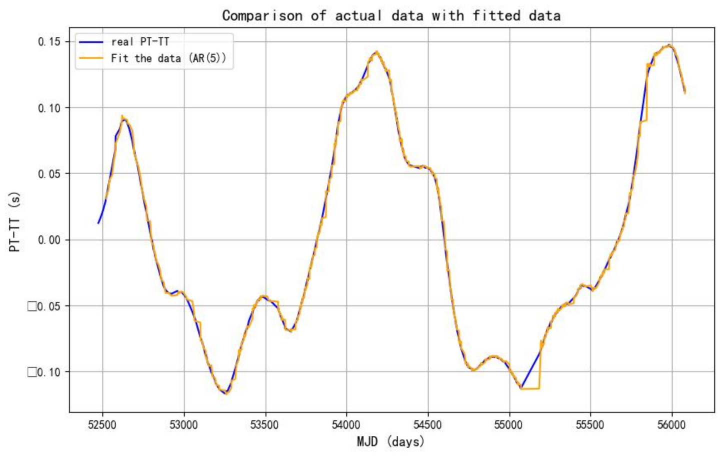


Figure 3: Autoregressive model fitting results

From the figure, it can be seen that the AR model is effective in fitting the pulsar timing noise data, with a small difference between the actual data and the fitted data. This further validates that the autoregressive model is a good tool for modeling pulsar timing noise. However, in regions with certain complex noise behaviors, further optimization or the use of other models may be needed to improve the fitting accuracy. The final results of the simulated function are as follows.

Table 2: Autoregressive model coefficients (estimated values)

	coef	Std err	z	P>|z|	[0.025	0.975]
const	2.094e-05	0.000	0.195	0.845	-0.000	0.000
y.L1	1.0747	0.037	29.111	0.000	1.002	1.147
y.L2	0.1360	0.055	2.473	0.013	0.028	0.244
y.L3	-0.0161	0.055	-0.292	0.770	-0.124	0.092
y.L4	0.0286	0.055	0.520	0.603	-0.079	0.136
y.L5	-0.2251	0.037	-6.085	0.000	-0.298	-0.153

It shows the fitting results of an autoregressive (AR) model, including the coefficients, standard errors, z-values, p-values, and 95% confidence intervals.

Table 3: Complex roots and modulus of the AR model

	Real	Imaginary	Modulus
AR.1	1.0070	-0.0000j	1.0070
AR.2	1.1198	-0.0000j	1.1198
AR.3	-1.5653	-0.0000j	1.5653
AR.4	-0.2172	-1.5716j	1.5865
AR.5	-0.2172	+1.5716j	1.5865

It shows the results of the autoregressive (AR) model coefficients in their complex form, including the real and imaginary parts, as well as the modulus (absolute value) of the coefficients. These complex coefficients reflect the dynamic behavior and periodicity in the model. Based on the above results, the final representation of the model is given by equation (6):

$$y_t = 2.094\times10^{-5} + 1.0747y_{t-1} + 0.1360y_{t-2} - 0.0161y_{t-3} + 0.0286y_{t-4} - 0.2251y_{t-5} + \epsilon_t \quad (6)$$

Where, y_t : value at the current time point, y_{t-1} : value at the previous time point (lag 1), y_{t-2} : value at the time point two steps before (lag 2), y_{t-3} : value at the time point three steps before (lag 3), y_{t-4} : value at the time point four steps before (lag 4), y_{t-5} : value at the time point five steps before (lag 5). ϵ_t is the error term, representing the random component that the model cannot predict.

From the model parameters, it can be seen that the fitting effect of the AR model is very good. The R^2 value is 0.99877, which shows that the model fits the data variation very well, and all the coefficients are greater than 1, indicating that the model is stable. Based on the obtained autoregressive coefficients, the AR model is able to effectively use the historical data from the previous five time points to predict the current timing noise. The coefficients of each autoregressive term and their values represent the impact of different time lags on the current timing noise. The error term ϵ_t represents the random component that the model cannot fully predict, thus the residual noise affects the fitting accuracy.

4.2 Model establishment and solution of question 2

4.2.1 ARIMA model establishment

Due to the obvious periodicity of the data, we use the ARIMA model for prediction. The ARIMA model (AutoRegressive Integrated Moving Average Model) is a classical statistical model used for time series forecasting. The ARIMA model combines autoregression (AR), differencing (I), and moving average (MA) processes, making it suitable for stationary time series or time series that can be made stationary through differencing.

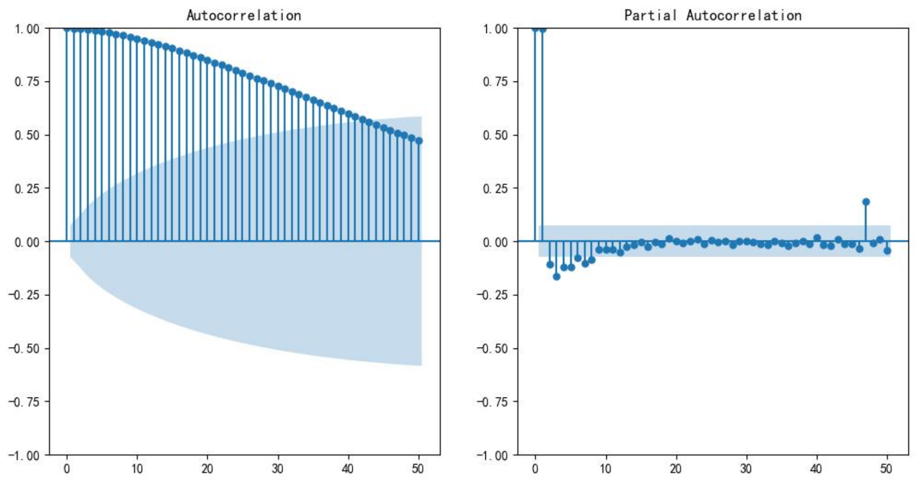


Figure 4: Autocorrelation and partial autocorrelation plots

The above figure describes the characteristics of autocorrelation and partial autocorrelation in the time series, making it easier to understand the lag dependence of the data. The autocorrelation is strong and gradually weakens as time progresses, while the partial autocorrelation indicates that the data primarily depends on the value at the previous time point, further confirming that the ARIMA model is suitable for prediction.

Time series data needs to be stationary, meaning that its mean and variance do not change over time. If the data is non-stationary, it can be made stationary through differencing operations. In the ARIMA model, the parameter d (usually $d=1$ or $d=2$) represents the number of differencings performed.Differencing (Differencing): $y_{t'} = y_t - y_{t-1}$ or for second differencing: $y_{t''} = y_{t'} - y_{t-2'}$ .Autoregressive (AR) models use the past values of the data to predict the current value. The equation of an AR model is:

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \epsilon_t$$

where y_t is the value at the current time point, $\phi_1, \phi_2, \ldots, \phi_p$ are the model coefficients, and ϵ_t is the error term (noise).

The Moving Average (MA) model uses past forecast errors to correct the current forecast. The order of the MA model (q) represents the lag order of past errors. The mathematical formula for the moving average part is (7):

$$y_t = \mu + \theta_1 \,{}_{t-1} + \theta_2 \,{}_{t-2} + \cdots + \theta_q \,{}_{t-q} + \,{}_t \tag{7}$$

Where, μ is the constant term (mean value), $\theta_1, \theta_2, \ldots, \theta_q$ are the moving average coefficients, and ϵ_t is the white noise (error term). The ARIMA model combines the AR part and the MA part, and allows the data to be made stationary through differencing operations. The mathematical formula for the ARIMA model is:

$$(1 - \phi_1 L - \phi_2 L^2 - \cdots - \phi_p L^p)(1 - L)^d y_t = \,{}_t + \theta_1 L \,{}_{t-1} + \theta_2 L^2 \,{}_{t-2} + \cdots + \theta_q L^q \,{}_{t-q} \tag{8}$$

where L is the lag operator, $L^k y_t = y_{t-k}$, and d is the differencing order.

When using the ARIMA model[4], the first task is to determine the model parameters (p, d, q), where p is the order of the AR part, d is the differencing order, and q is the order of the MA part. The ACF (Autocorrelation Function) and PACF (Partial Autocorrelation Function) plots can help in selecting appropriate parameters. After determining the appropriate ARIMA model, the following equation can be used for future predictions:

$$\hat{y}_{t+h} = \hat{\mu} + \sum_{i=1}^{p} \hat{\phi}_i \hat{y}_{t+h-i} + \sum_{j=1}^{q} \hat{\theta}_j \epsilon_{t+h-j} \tag{9}$$

where $\hat{y}_{t+h}$ is the forecast at future time , $\hat{\mu}$ is the constant term, $\hat{\phi}_i$ and $\hat{\theta}_j$ are the estimated AR and MA coefficients, and ϵ_t is the error term.

After fitting the model, the accuracy of the prediction can be evaluated using error indicators, including:

(1) Mean Squared Error (MSE): $\mathrm{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$

(2) Mean Absolute Error (MAE): $\mathrm{MAE} = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i|$

By using the ARIMA model, we can effectively handle non-stationary time series data, perform forecasting, and evaluate the model's performance. By selecting appropriate parameters (p, d, q) and suitable error indicators (such as MSE and MAE), we can optimize the forecasting results.

4.2.2Future trend prediction of pulsar timing noise based on the ARIMA model

The basic process of spare parts consumption prediction using the ARIMA(p,d,q) model is shown in the following figure 5.

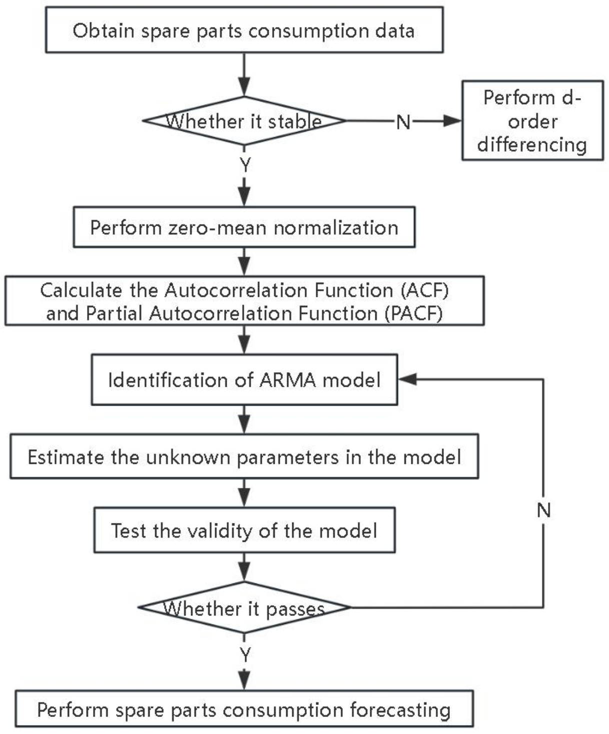

Obtain spare parts consumption data
Whether it stable
N
Perform d-order differencing
Y
Perform zero-mean normalization
Calculate the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF)
Identification of ARMA model
Estimate the unknown parameters in the model
Test the validity of the model
N
Whether it passes
Y
Perform spare parts consumption forecasting

Figure 5: Spare parts consumption prediction flowchart

Based on the algorithm flowchart, the following result plot was obtained through programming:

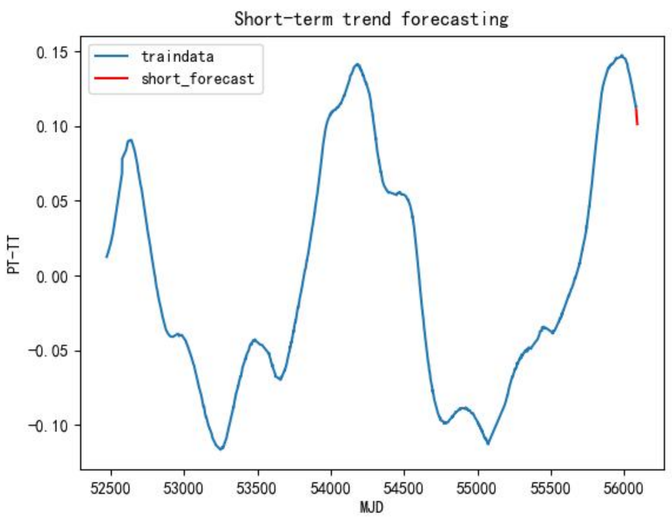


Figure 6: Short-term trend prediction plot

As can be seen from the figure above, the blue curve represents the training data, which shows multiple fluctuations between 52500 and 56000, with fluctuation amplitudes ranging from -0.1s to 0.15s. Some larger peaks and valleys can be observed. For example, there are larger peaks around 54000 and 54500, and deeper valleys around 53000 and 53500. The red curve represents the short-term prediction data, starting from approximately time 600. The predicted PT-TT (the difference between pulse arrival time and actual arrival time) values range from -0.1s to 0.1s. The Mean Squared Error (MSE) of the short-term prediction is 0.00016472551369894423, and the Mean Absolute Error (MAE) of the short-term prediction is 0.011474531626810048. Overall, this indicates that the prediction model can effectively capture the short-term fluctuation trend of the pulsar timing noise.

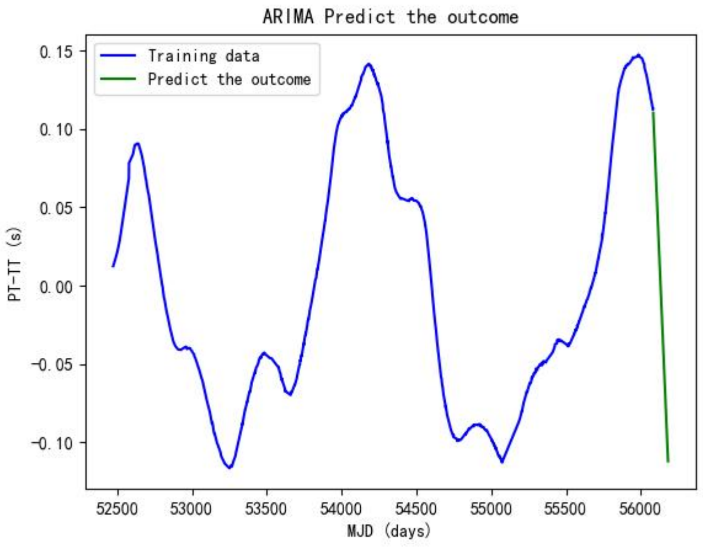


Figure 7: ARIMA prediction results

The green curve in the figure represents the predicted results, starting from MJD around 56000. The predicted PT-TT (the difference between the pulse arrival time and the actual arrival time) values range from -0.1s to 0.1s. Mean Squared Error (MSE): 0.0017055775490409987; Mean Absolute Error (MAE): 0.03754404073585997. For long-term prediction, it can be inferred that as the forecast period extends, the uncertainty of the prediction will gradually increase. This is because the ARIMA model makes predictions based on historical data, and for data further into the future, the influence of historical data weakens, and the impact of external factors may lead to an increase in prediction bias.

4.3 Model establishment and solution of question 3

When electromagnetic waves pass through the atmosphere, their propagation speed will decrease due to the influence of refraction. The refractive time delay is caused by the atmospheric refractive index $n(h)$, and the time delay of its propagation path can be expressed by the following formula (10):

$$\Delta A = \int_{h_0}^{h_2} (n(h) - 1) \frac{dh}{c} \tag{10}$$

Among them, h_0 represents the height above the Earth's surface, h_2 represents the highest point of the propagation path, $n(h)$ represents the atmospheric refractive index at the height of h, dh represents the differential distance along the propagation path, and c represents the speed of light.

4.3.1 Saastamoinen model

The refractive time delay is divided into dry delay and wet delay. The dry delay is mainly caused by the refraction of electromagnetic waves by dry gases (such as nitrogen, oxygen and other components) in the atmosphere; the wet delay is caused by the refraction of electromagnetic waves by water vapor in the atmosphere. The total time delay can be expressed by the following formula (11).

$$\Delta\tau = \Delta\tau_{\text{dry}} + \Delta\tau_{\text{wet}} \tag{11}$$

- Dry Delay

$$\tau_{\text{dry}} = 0.002278 \cdot \frac{P_0}{T_0}$$

Among them, P_0 represents the standard atmospheric pressure, and T_0 represents the Kelvin temperature, usually selected as 288.15 K.

- Wet Delay

$$\tau_{\text{wet}} = 0.002277 \cdot \frac{e}{1 - 0.00266 \cdot \cos(2\phi) - 0.00028 \cdot h}$$

Among them, e represents the water - vapor pressure, h represents the altitude, and ϕ represents the latitude.

1.The constraint condition of the zenith time delay

The total zenith time delay can be expressed as

$$\tau_{\text{zenith}} = \tau_{\text{dry}} + \tau_{\text{wet}}$$

The constraint condition is that the zenith time delay is less than or equal to 7.69 nanoseconds, which can be further expressed as

$$\tau_{\text{zenith}} = \min(\tau_{\text{zenith}}, 7.69\ \text{ns}).$$

2.Time - delay Frequency Dependence

The relationship between the refractive time - delay and the radio - frequency can be represented by the following formula (12).

$$\Delta\tau(f) = \Delta\tau_0 \cdot \left(\frac{f_0}{f}\right)^2 \tag{12}$$

Among them, $\Delta\tau(f)$ represents the refractive time - delay when the radio - frequency is f , $\Delta\tau_0$ represents the refractive time - delay when the radio - frequency is f_0 , f_0 represents the reference signal frequency, and f represents the actual frequency of the radio - signal.

It can be seen that the refractive time delay is inversely proportional to the square of the radio frequency. The wet delay is extremely sensitive to the frequency. Since the frequency in this question is greater than 20 GHz, the influence of the wet delay on the refractive time delay can be ignored. The following figure simulates the influence relationship diagram of the dry delay and the wet delay on the zenith time delay at a surface latitude of 30 degrees and an altitude of 0.5 km.

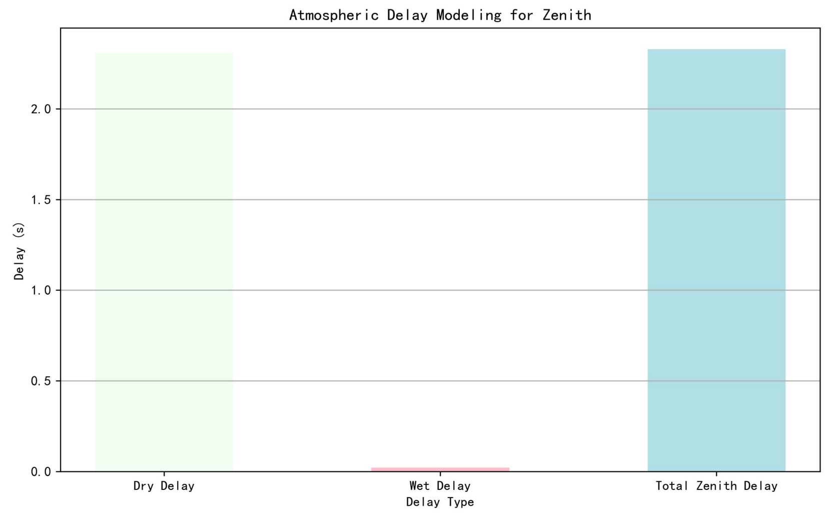


Figure 8 Atmospheric Delay Modeling for Zenith

The image shows that the influence of the wet time delay on the zenith time delay is very small and can be neglected.

3. Herring Mapping Function

The Herring mapping function is an atmospheric delay mapping function, which is used to extend the refractive time delay in the zenith direction to any elevation angle ϵ . Its function is to map the zenith delay τ_{zenith} to the delay at any elevation angle $\tau(\epsilon)$, and make corrections by considering the path changes of the signal passing through the atmosphere at different elevation angles.

Use the mapping function to extend the zenith delay to any elevation angle ϵ :

$$m(\epsilon) = \frac{1}{\sin\epsilon + a \cdot \frac{\cos\epsilon}{1 + b / (\sin\epsilon + c)}} \tag{13}$$

The refracted time - delay after mapping

$$\tau(\ ) = \tau_{\text{zenith}} \cdot m(\ )$$

For a more detailed representation of the refractive time - delay, we transform the traditional path integral into a height integral, and the refractive time - delay formula is updated as follows:

$$\Delta\tau = \frac{1}{c}\int_{h_0}^{h_2} (n(h) - 1) \cdot \sqrt{1 + \left(\frac{R}{R + h}\right)^2 \tan^2 z}\, dh \tag{14}$$

4.3.2 The solution results of the Saastamoinen model

By solving the equations, we obtained the relationship between the total refraction delay and the elevation angle at different radio frequencies. The results are shown in the figure below.

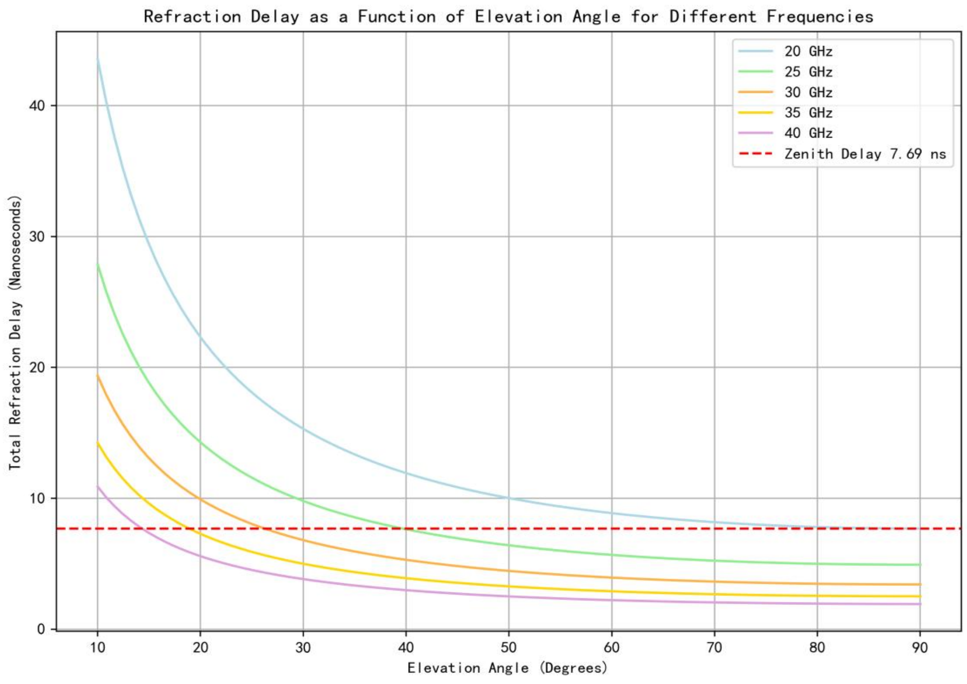


Figure 9 High-frequency Observation Atmospheric Refraction Delay Variation

The zenith delays at different radio frequencies are detailed in the table below:

Table 4 Total Refractive Time Delays at Different Elevation Angles

Radio Frequency (GHz)	Zenith Delay （ns）
20	7.69
25	4.92
30	3.42
35	2.51
40	1.92

From the analysis of the chart, it is evident that the total refraction delay decreases as the elevation angle increases. Moreover, it satisfies the condition that the zenith delay does not exceed 7.69 ns for radio frequencies above 20 GHz.

To more intuitively illustrate the relationship among observation frequency, elevation angle, and total refraction delay, a 3D plot is introduced. The results are shown in the figure below.

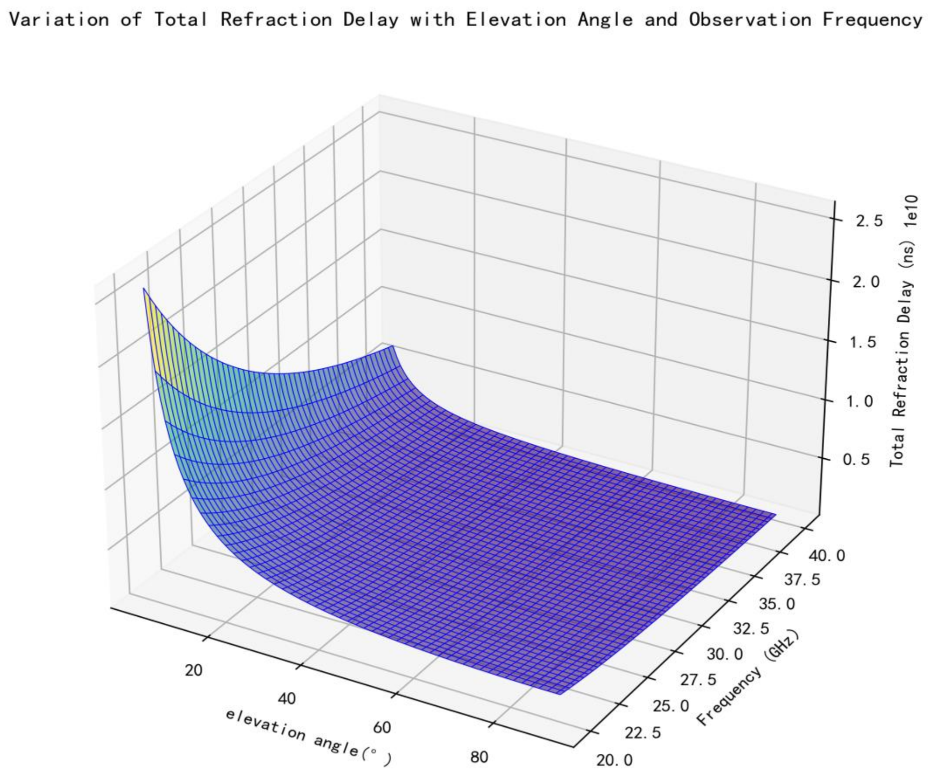


Figure 10 Effect of Frequency and Elevation Angle on Refraction Delay

4.4 Model establishment and solution of question 4

4.4.1 The Dry Delay Based on the Saastamoinen Model

- Dry Delay

The calculation formula for dry delay based on the Saastamoinen model is:

$$\tau_{\text{dry}} = 0.0022768 \cdot \frac{P}{\cos(z)} \tag{15}$$

Among them, P represents the surface air pressure (hPa), z represents the zenith angle, and the relationship between it and the elevation angle is $z = \pi / 2 - \epsilon$.

- Wet Delay

The calculation formula for wet delay is

$$\tau_{\text{wet}} = 0.002277 \cdot \frac{e}{1 - 0.00266 \cdot \cos(2\phi) - 0.00028 \cdot h} \tag{16}$$

Among them, e represents the surface water vapor pressure, ϕ represents the latitude, and h represents the observation height.

4.4.2 GMF Mapping Function

During the observation of small elevation angles, the original Herring mapping function is prone to error accumulation due to an overly small denominator. For this reason, we introduce the GMF (Global Mapping Function) mapping, whose mapping formula incorporates a more complex parameter structure and can describe the atmospheric delay of the signal path at small elevation angles more accurately. It can be expressed by the following formula(17).

$$m(\epsilon) = \frac{1 + a / (1 + b / (1 + c))}{\sin(\ ) + a / (\sin(\ ) + b / (\sin(\ ) + c))} \tag{17}$$

Among them, a , b , and c represent empirical parameters, and ϵ represents the elevation angle.

4.4.3 Multilayer Exponential Distribution Integral

Here, we divide the atmosphere into several small layers. The refractive index $n(h)$ of each layer is assumed to change linearly. Then the change of the refractive index with height h can be expressed as follows, where A and B represent empirical parameters.

$$n(h) = 1 + A \cdot e^{-B \cdot h}$$

Calculate the delay of each layer through trapezoidal integration and sum them up to obtain the total delay. The atmospheric delay formula (18) is updated as follows.

$$\Delta\tau = \frac{1}{c}\sum_{i}\int_{h_i}^{h_{i+1}} (n(h)-1)\sqrt{1+\left(\frac{R}{R+h}\right)^2 \tan^2 z\, dh} \tag{18}$$

Among them, $\Delta\tau$ represents the atmospheric delay or refractive delay, c represents the speed of light, $n(h)$ represents the refractive index of the atmosphere at a certain height h, R is the radius of the Earth, h is the height of the current layer, and z is the Zenith Angle, that is, the angle from the observation point to the zenith.

4.4.4 Frequency Correction

Atmospheric delay has different effects on signals of different frequencies. A frequency correction formula (19) is used.

$$\pi_f = \pi \cdot \left(\frac{f_0}{f}\right)^2 \tag{19}$$

Among them, f_0 represents the reference frequency and f represents the observation frequency.

4.4.5 Solution Result Analysis

Through improvements in aspects such as atmospheric stratification, refractive index calculation, atmospheric delay calculation, and mapping function optimization, this model has the following characteristics:

Better Adaptability to Small Elevation Angles: By optimizing the Herring mapping function into the GMF (Global Mapping Function) mapping, it can maintain a relatively high calculation accuracy under small elevation angles and eliminate the deviation of the zenith delay model at low elevation angles.

Gradient Integration by Stratification: Through the stratified integration model, the height distribution characteristics of the atmospheric refractive index are refined, which improves the physical accuracy of delay estimation.

Frequency Extensibility: The model takes into account the effect of atmospheric delay reduction under high-frequency conditions and is applicable to the observation requirements with radio frequencies higher than 20 GHz.

After optimization, the relationship between the small elevation angles (ranging from 1 degree to 10 degrees) and the total refractive time delay is shown in the figure below.

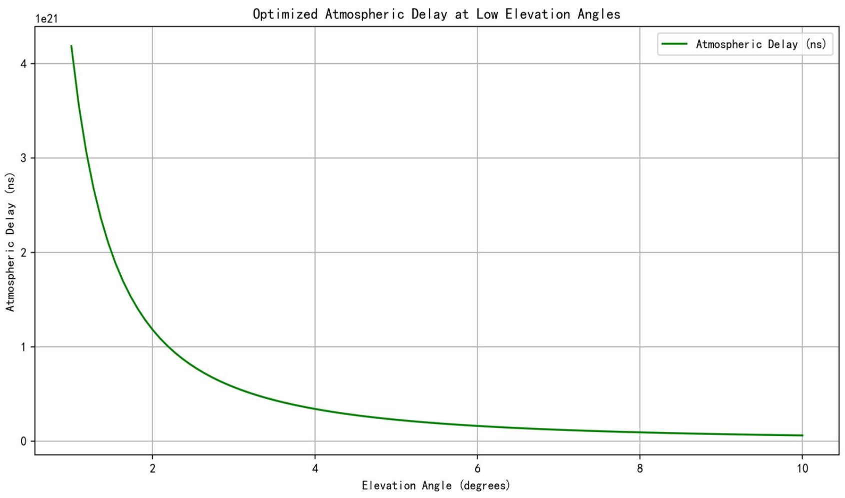


Figure 11 Optimized Atmospheric Delay at Low Elevation Angles

4.4.5 Error Analysis

To verify the accuracy of the model, we added random noise to the observed delay to simulate the real observation environment. The errors between the actual results and the predicted results are as follows:

Table 5 Table of Errors Between Actual and Predicted Results

MSE (s^2)	MAE (s)	Standard Deviation of Observed Delays (s)	Standard Deviation of Predicted Delays (s)
8.27295e-19	7.22794e-10	7.46589e+02	7.53349e+02

The MSE and MAE values are both at the nanosecond level, indicating that the overall error magnitude of the model is relatively small.

4.4.6 Achievable Goal

1. Significantly Improve the TOA (Time of Arrival) Accuracy

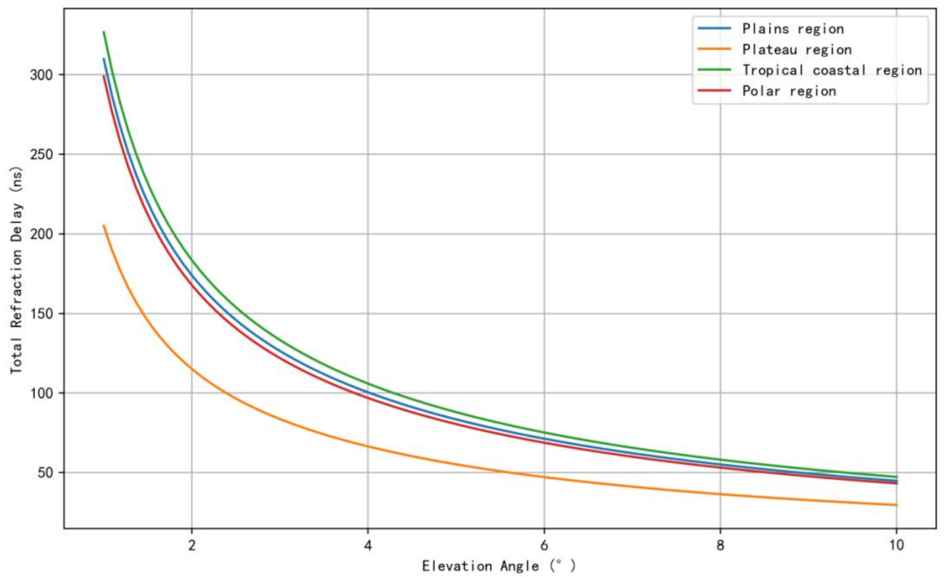
For small elevation angle observations with frequencies above 20 GHz, the atmospheric delay error is kept under 0.1 ns, improving accuracy by over an order of magnitude compared to traditional models. This enhances pulsar time accuracy and ensures high-precision delay calculations across various meteorological conditions and observation environments, guaranteeing reliable TOA measurement accuracy.

2. Adaptation to a Wide Range of Meteorological Conditions

The model adapts to extreme meteorological conditions, such as heavy precipitation, strong winds, high humidity, and low temperatures, with deviation from actual measured atmospheric delay not exceeding 0.2 ns. It works accurately across various observation stations—ground-fixed, mobile, or space-based—requiring only relevant observation parameters and meteorological data for precise delay calculation without extensive customization.

5. Test the Models

In question four, to analyze and simulate atmospheric delays under different climatic and geographical conditions, typical meteorological characteristic data from plains regions, plateau regions, tropical coastal regions, and polar regions were referenced. The resulting relationship between low elevation angles and total refraction delay is as follows.



Figture 11 Atmospheric Delay Relationship Chart for Typical Geographic Locations

By optimizing meteorological parameters and mapping functions for different regions, the TOA (Time of Arrival) accuracy was improved, and atmospheric delay errors were reduced. Observations revealed that the delay trends in different regions align with actual meteorological conditions, validating the reasonableness of the atmospheric delay model.

6.Sensitivity Analysis

In the modeling of atmospheric delay, the key input parameters of the model (such as air pressure, water vapor pressure, elevation angle, and observation frequency) have different degrees of influence on the delay. In order to evaluate the impact of these parameters on the output of the delay model, here we conduct a sensitivity analysis on each parameter. The results of the analysis are shown in the figure below.

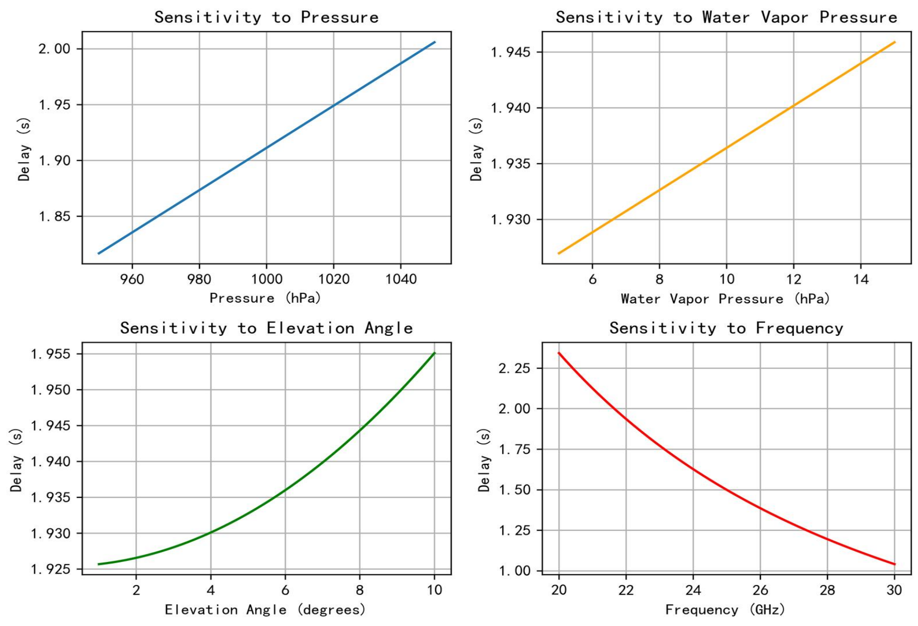


Figure 12 Factors Affecting Atmospheric Delay

The sensitivity analysis shows that the delay is most sensitive to the elevation angle. The delay decreases exponentially with the frequency, and high-frequency observations can significantly reduce the atmospheric delay. The impact of air pressure on the delay is also relatively significant, and the correction of dry delay requires high-precision air pressure measurements. In contrast, the water vapor pressure has a relatively low sensitivity to the delay.

Overall, low elevation angle correction, high-frequency selection, and air pressure measurement accuracy are the main directions for optimizing the model, while wet delay serves as a supplementary optimization measure to further improve the accuracy and reliability of the model.

7.Strengths and Weakness

7.1 Advantages of the model

(1) The ARIMA model can handle non-stationary data because it makes the data stationary through differencing (the I component). Therefore, it is applicable not only to stationary data but also to time series with trends, seasonality, or other long-term changes.
(2) The ARIMA model can perform both short-term and long-term predictions. By adjusting the model parameters, ARIMA can efficiently predict across various time scales, especially for pulsar timing noise data, which tends to show strong trends.
(3) The model uses multilayer atmospheric distribution and trapezoidal integration to improve precision in atmospheric refractive index variations. It is ideal for low-Earth orbit satellite constellations like Starlink, optimizing signal delay corrections in low-elevation areas and enhancing signal quality in edge coverage regions.

7.2 Disadvantages of the model

(1) The selection of parameters (p, d, q) for the ARIMA model is relatively complex and needs to be based on the data's autocorrelation (ACF, PACF) and stationarity (the order of differencing, d). Incorrect parameter selection may lead to poor model performance or overfitting.
(2) In the high-frequency radio model, the impact of ground-level atmospheric parameters, such as pressure, temperature, and humidity, on dry delay wasn't fully considered in terms of their spatial variability.

References

[1] Li Bian, Qu Lili, Chen Yongqi. Research on Millisecond Pulsar Timing Noise Processing Methods [J]. Journal of Time and Frequency, 2020, 43(04): 310-317. DOI: 10.13875/j.issn.1674-0637.2020-04-0310-08.

[2] Gao Feng. Analysis and Application of Millisecond Pulsar Timing Noise [D].

University of Chinese Academy of Sciences (National Time Service Center, Chinese Academy of Sciences), 2019.

[3] Xie Ronghua, Zhang Chenghong. Application of Curve Fitting and Autoregressive Models in Deformation Prediction[J]. Surveying and Mapping Technology and Equipment, 2020, 22(04): 47-49.

[4] Li Changsheng, Liu Sujun, Liu Zongcheng, et al. Research on the Analysis of Yellow River Water-Sediment Monitoring Data Based on Time Series Models[J]. Modern Information Technology, 2024, 8(20): 159-163+168. DOI: 10.19850/j.cnki.2096-4706.2024.20.032.

[5] Zhang Hanwei, Mao Wei, Tie Qiongxian. Numerical Simulation of Astronomical Atmospheric Refraction and Mapping Function[J]. Journal of Liaoning University of Engineering and Technology (Natural Science Edition), 2008, (01): 32-34.

[6] Chen Shaojie, Zheng Yong, Zhan Yinha, et al. Comparative Analysis of stronomical Atmospheric Refraction Correction Models[J]. Global Navigation Satellite Systems, 2017, 42(06): 61-65. DOI: 10.13442/j.gnss.1008-9268.2017.06.010.

Appendix

Appendix 1

Introduce: 基于自回归模型的脉冲星计时噪声拟合 Python 代码

```
import warnings
warnings.filterwarnings("ignore", category=UserWarning, module="matplotlib")

import numpy as np
import pandas as pd
from statsmodels.tsa.ar_model import AutoReg
import matplotlib.pyplot as plt
from matplotlib import font_manager
import matplotlib
matplotlib.rcParams['font.family'] = 'SimHei'   # 或 'Microsoft YaHei'

# 设置字体，使用系统中支持中文的字体
font_path = "C:/Windows/Fonts/simhei.ttf"   # Windows 系统下的 SimHei 字体路径
prop = font_manager.FontProperties(fname=font_path)

# 读取数据 (假设你已经将数据加载为 DataFrame)
df = pd.read_excel('C:\\Users\\12070\\Downloads\\2024“ShuWei Cup”_Problem\\2024“ShuWei Cup”_Problem\\2024_“ShuWei Cup”C_Problem\\Attachment 1.xlsx',header=None)

mjd = df.iloc[0:700, 0].to_numpy()   # MJD (时间)
pt_tt = df.iloc[0:700, 1].to_numpy()   # PT-TT 数据

# 计算 RMS 值（如果需要）
# 可以在这里对数据进行处理或选择使用 RMS 值
# 这里假设我们直接使用 pt_tt 数据进行拟合
data = pt_tt
```

```
# 使用 AutoReg 来选择 AR 模型的阶数
# AutoReg 是用于拟合自回归模型的函数，lag (p) 是自回归的阶数
model = AutoReg(data, lags=5)  # lags 参数定义了 AR 模型的阶数 p
model_fitted = model.fit()

# 打印模型的拟合结果
print(model_fitted.summary())

# 获取拟合后的模型参数
params = model_fitted.params
print("AR 模型的参数:", params)

# 使用拟合模型对未来进行预测
# 假设你需要预测未来的若干数据点
forecast_steps = 10
forecast = model_fitted.predict(start=len(data), end=len(data) + forecast_steps - 1)

# 绘制图表时使用该字体
plt.plot(mjd, data, label="实际 PT-TT 数据")
plt.plot(mjd[-1] + np.arange(1, forecast_steps + 1), forecast, label="预测数据",
color="red")
plt.xlabel("MJD (days)", fontproperties=prop)
plt.ylabel("PT-TT (s)", fontproperties=prop)
plt.legend(prop=prop)
plt.show()

# 计算总平方和 (TSS) 仅基于拟合部分的原始数据
tss = np.sum((data[5:] - np.mean(data[5:])) ** 2)

# 计算残差平方和 (RSS)
rss = np.sum((data[5:] - model_fitted.fittedvalues) ** 2)

# 计算 R-squared
r_squared = 1 - (rss / tss)
```

```
print("R-squared:", r_squared)

########################################
# 模型的拟合数据
fitted_values = model_fitted.fittedvalues   # AR 模型对训练数据的拟合结果

# 实际数据
actual_data = data

# 绘制实际数据和拟合数据的图表
plt.figure(figsize=(10, 6))
plt.plot(mjd, actual_data, label="real PT-TT", color='blue')
plt.plot(mjd[5:], fitted_values, label="Fit the data (AR(5))", color='orange')   # 从第
6 个点开始绘制拟合数据
plt.xlabel("MJD (days)", fontsize=12)
plt.ylabel("PT-TT (s)", fontsize=12)
plt.title("Comparison of actual data with fitted data", fontsize=14)
plt.legend()
plt.grid(True)
plt.show()
```

Appendix 2

Introduce: 基于 AR 模型的短期预测

```
import numpy as np
import pandas as pd
import statsmodels.api as sm
from statsmodels.tsa.ar_model import AutoReg
import matplotlib.pyplot as plt
from matplotlib import rcParams
from sklearn.metrics import mean_squared_error, mean_absolute_error

# 设置字体为 SimHei (黑体)，如果你使用的是 Windows 操作系统，或者安装
了该字体
rcParams['font.sans-serif'] = ['SimHei']   # 支持中文
```

```
rcParams['axes.unicode_minus'] = False   # 正常显示负号

# 读取验证数据（例如，MJD 和 PT-TT）
excel_file = "C:\\Users\\12070\\Downloads\\2024“ShuWei Cup”_Problem\\2024“ShuWei Cup”_Problem\\2024_“ShuWei Cup”C_Problem\\validation_data.xlsx"

# 从 Excel 读取验证数据（假设数据在第一个工作表，且 MJD 和 PT-TT 在前两列）
validation_df = pd.read_excel(excel_file, sheet_name='Sheet1', header=None)

# 假设 MJD 和 PT-TT 在前两列
validation_data = validation_df.iloc[:, [0, 1]].values
# 将其转换为 DataFrame 格式
validation_df = pd.DataFrame(validation_data, columns=['MJD', 'PT-TT'])

# 提取 PT-TT 数据
validation_pt_tt = validation_df['PT-TT'].to_numpy()

# 读取数据
df = pd.read_excel('C:\\Users\\12070\\Downloads\\2024“ShuWei Cup”_Problem\\2024“ShuWei Cup”_Problem\\2024_“ShuWei Cup”C_Problem\\Attachment 1.xlsx', header=None)

# 取前 700 个数据点作为训练数据
mjd = df.iloc[0:700, 0].to_numpy()   # MJD (时间)
pt_tt = df.iloc[0:700, 1].to_numpy()   # PT-TT 数据

# 训练 AR 模型
model = AutoReg(pt_tt, lags=5)   # 只使用 PT-TT 数据来训练
model_fitted = model.fit()

# 打印模型的参数以确保模型正确加载
print(model_fitted.summary())
```

```
# 预测未来 8 天
forecast_steps_short = 8
forecast_short = model_fitted.predict(start=len(pt_tt), end=len(pt_tt) + forecast_steps_short - 1)

# 绘制短期预测结果
plt.plot(np.arange(len(pt_tt)), pt_tt, label='训练数据')
plt.plot(np.arange(len(pt_tt), len(pt_tt) + forecast_steps_short), forecast_short, label='短期预测', color='red')
plt.xlabel('时间')
plt.ylabel('PT-TT')
plt.legend()
plt.title('短期趋势预测')
plt.show()

# 假设你已经有实际的验证数据（validation_data）

# 将验证数据转换为 DataFrame
validation_df = pd.DataFrame(validation_data, columns=['MJD', 'PT-TT'])

# 取实际的 8 个验证数据（假设验证数据里有 8 个数据点）
y_true_short = validation_df['PT-TT'].to_numpy()[:8]
y_pred_short = forecast_short

# 计算 MSE 和 MAE
mse_short = mean_squared_error(y_true_short, y_pred_short)
mae_short = mean_absolute_error(y_true_short, y_pred_short)

print(f"短期预测的均方误差 (MSE): {mse_short}")
print(f"短期预测的平均绝对误差 (MAE): {mae_short}")
```

Appendix 3

Introduce: 基于 ARIMA 模型的长期预测

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from sklearn.metrics import mean_squared_error, mean_absolute_error
from matplotlib import rcParams

# 设置字体为 SimHei (黑体)，如果你使用的是 Windows 操作系统，或者安装了该字体
rcParams['font.sans-serif'] = ['SimHei']  # 支持中文
rcParams['axes.unicode_minus'] = False  # 正常显示负号

# 读取训练数据
df = pd.read_excel('C:\\Users\\12070\\Downloads\\2024“ShuWei Cup”_Problem\\2024“ShuWei Cup”_Problem\\2024_“ShuWei Cup”C_Problem\\Attachment 1.xlsx', header=None)
mjd_train = df.iloc[0:700, 0].to_numpy()  # MJD (时间)
pt_tt_train = df.iloc[0:700, 1].to_numpy()  # PT-TT 数据

# 绘制 ACF 和 PACF 图，帮助选择 ARIMA 参数
plt.figure(figsize=(12, 6))
plt.subplot(121)
plot_acf(pt_tt_train, ax=plt.gca(), lags=50)  # ACF 图
plt.subplot(122)
plot_pacf(pt_tt_train, ax=plt.gca(), lags=50)  # PACF 图
plt.show()

# 实际验证数据
validation_df = pd.read_excel('C:\\Users\\12070\\Downloads\\2024“ShuWei Cup”_Problem\\2024“ShuWei Cup”_Problem\\2024_“ShuWei Cup”C_Problem\\validation_data.xlsx', header=None)
actual_values = validation_df.iloc[:, 1].to_numpy()  # 假设验证数据在第二列

# 使用 ARIMA 模型（不考虑季节性）
```

```
model_arima = ARIMA(pt_tt_train, order=(0, 2, 2))   # 这里是 ARIMA(p, d, q) 模型，调整 p, d, q

# 拟合模型
model_fitted_arima = model_arima.fit()

# 打印模型总结
print(model_fitted_arima.summary())

# 预测未来的 100 天数据
forecast_steps = 100
forecast_values_arima = model_fitted_arima.forecast(steps=forecast_steps)

# 绘制训练数据和预测结果
plt.plot(mjd_train, pt_tt_train, label="Training data", color="blue")
plt.plot(np.arange(mjd_train[-1] + 1, mjd_train[-1] + forecast_steps + 1), forecast_values_arima, label="Predict the outcome", color="green")
plt.xlabel("MJD (days)")
plt.ylabel("PT-TT (s)")
plt.title("ARIMA Predict the outcome")
plt.legend()
plt.show()

# 截取预测数据与实际验证数据长度一致
forecast_values_trimmed = forecast_values_arima[:len(actual_values)]

# 计算 MSE 和 MAE
mse = mean_squared_error(actual_values, forecast_values_trimmed)
mae = mean_absolute_error(actual_values, forecast_values_trimmed)

print(f"Mean square error (MSE): {mse}")
print(f"Mean absolute error (MAE): {mae}")

# 计算相对 MAE
```

```
relative_mae = mae / np.mean(np.abs(actual_values))   # 使用实际值的绝对值均值
print(f"相对平均绝对误差: {relative_mae}")
```

Appendix 4

Introduce: 基于大气折射模型求解问题 3 的 Python 代码

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# 设置 matplotlib 支持中文字体显示
plt.rcParams['font.sans-serif'] = ['SimHei']   # 使用黑体显示中文
plt.rcParams['axes.unicode_minus'] = False   # 解决负号显示问题

# 常数定义
c = 3.0e8   # 光速，单位：m/s
f0 = 20e9   # 参考频率，20 GHz
max_zenith_delay = 7.69e-9   # 天顶延迟最大值，单位：秒

# 定义 Saastamoinen 模型的干延迟计算
def dry_delay(P, z):
    """
    计算干延迟
    P: 地面气压 (hPa)
    z: 天顶角 (弧度)
    """
    return 0.0022768 * P / np.cos(z)

# 定义 Saastamoinen 模型的湿延迟计算
def wet_delay(e, phi, h):
    """
    计算湿延迟
    e: 地面水汽压 (hPa)
    phi: 纬度 (弧度)
    h: 高度 (km)
    """
    return 0.002277 * e / (1 - 0.00266 * np.cos(2 * phi) - 0.00028 * h)

# 定义频率依赖性修正
def frequency_correction(delay, f):
    """
    对折射时延进行频率修正
    delay: 天顶方向的折射时延 (秒)
    f: 当前信号频率 (Hz)
    """
    return delay * (f0 / f) ** 2

# 定义 Herring 映射函数
def mapping_function(epsilon, a=0.0029, b=0.0005, c=0.001):
    """
    使用 Herring 模型的映射函数将天顶延迟转换为任意仰角延迟
    epsilon: 仰角 (弧度)
    a, b, c: Herring 映射函数的经验参数
    """
    return 1 / (np.sin(epsilon) + a * (np.cos(epsilon) / (1 + b / (np.sin(epsilon) + c))))

# 计算总折射延迟
def total_delay(P, e, phi, h, f, epsilon):
    """
    计算给定条件下的总折射延迟
    P: 地面气压 (hPa)
    e: 地面水汽压 (hPa)
    phi: 纬度 (弧度)
    h: 高度 (km)
    f: 观测频率 (Hz)
    epsilon: 仰角 (弧度)
    """
    # 计算天顶方向的干延迟和湿延迟
    tau_dry = dry_delay(P, 0)   # 天顶角 z = 0
    tau_wet = wet_delay(e, phi, h)
    tau_zenith = tau_dry + tau_wet

    # 检查天顶延迟是否超过限制
    if tau_zenith > max_zenith_delay:
        tau_zenith = max_zenith_delay

    # 频率修正
    tau_corrected = frequency_correction(tau_zenith, f)

    # 映射到任意仰角
    m = mapping_function(epsilon)
    return tau_corrected * m

# 设置参数
P = 1013.25   # 地面气压，单位：hPa
e = 10.0         # 地面水汽压，单位：hPa
phi = np.radians(30)   # 纬度，单位：弧度
h = 0.5   # 高度，单位：km
f = 30e9   # 观测频率，单位：Hz，假设高于 20 GHz

# 生成仰角范围，从 5 度到 90 度
elevation_angles = np.radians(np.linspace(10, 90, 100))
delays = [total_delay(P, e, phi, h, f, epsilon) * 1e9 for epsilon in elevation_angles]   #
将延迟转换为纳秒

# 绘制结果
plt.figure(figsize=(10, 6), dpi=400)
plt.plot(np.degrees(elevation_angles), delays, label="总折射延迟", color='blue')
plt.axhline(y=7.69, color='red', linestyle='--', label="天顶延迟 7.69 ns")
plt.xlabel("仰角 (度)")
plt.ylabel("总折射延迟 (纳秒)")
plt.title("高频观测 (>20 GHz) 条件下的折射延迟随仰角变化")
plt.legend()
plt.grid(True)
plt.show()
```

Appendix 5

Introduce: 基于大气折射模型求解问题 4 的 Python 代码

```
import numpy as np
import matplotlib.pyplot as plt

# 设置 matplotlib 支持中文字体显示
plt.rcParams['font.sans-serif'] = ['SimHei']   # 使用黑体显示中文
plt.rcParams['axes.unicode_minus'] = False   # 解决负号显示问题

# 常数定义
c = 3.0e8   # 光速，单位：m/s
f0 = 20e9   # 参考频率，20 GHz
max_zenith_delay = 7.69e-9   # 天顶延迟最大值，单位：秒

# 定义 Saastamoinen 模型的干延迟计算
def dry_delay(P, z):
    """
    计算干延迟
    P: 地面气压 (hPa)
    z: 天顶角 (弧度)
    """
    return 0.0022768 * P / np.cos(z)

# 定义 Saastamoinen 模型的湿延迟计算
def wet_delay(e, phi, h):
    """
    计算湿延迟
    e: 地面水汽压 (hPa)
    phi: 纬度 (弧度)
    h: 高度 (km)
    """
    return 0.002277 * e / (1 - 0.00266 * np.cos(2 * phi) - 0.00028 * h)
```

```
# 定义频率依赖性修正
def frequency_correction(delay, f):
    """
    对折射时延进行频率修正
    delay: 天顶方向的折射时延 (秒)
    f: 当前信号频率 (Hz)
    """
    return delay * (f0 / f) ** 2

# 定义 Herring 映射函数
def mapping_function(epsilon, a=0.0029, b=0.0005, c=0.001):
    """
    使用 Herring 模型的映射函数将天顶延迟转换为任意仰角延迟
    epsilon: 仰角 (弧度)
    a, b, c: Herring 映射函数的经验参数
    """
    return 1 / (np.sin(epsilon) + a * (np.cos(epsilon) / (1 + b / (np.sin(epsilon) + c))))

# 计算总折射延迟
def total_delay(P, e, phi, h, f, epsilon):
    """
    计算给定条件下的总折射延迟
    P: 地面气压 (hPa)
    e: 地面水汽压 (hPa)
    phi: 纬度 (弧度)
    h: 高度 (km)
    f: 观测频率 (Hz)
    epsilon: 仰角 (弧度)
    """
    # 计算天顶方向的干延迟和湿延迟
    tau_dry = dry_delay(P, 0)   # 天顶角 z = 0
    tau_wet = wet_delay(e, phi, h)
    tau_zenith = tau_dry + tau_wet
    # 检查天顶延迟是否超过限制
    if tau_zenith > max_zenith_delay:
        tau_zenith = max_zenith_delay

    # 频率修正
    tau_corrected = frequency_correction(tau_zenith, f)

    # 映射到任意仰角
    m = mapping_function(epsilon)
    return tau_corrected * m

# 设置参数
P = 1013.25   # 地面气压，单位：hPa
e = 10.0        # 地面水汽压，单位：hPa
phi = np.radians(30)   # 纬度，单位：弧度
h = 0.5   # 高度，单位：km
f = 30e9   # 观测频率，单位：Hz，假设高于 20 GHz

# 生成仰角范围，从 5 度到 10 度
elevation_angles = np.radians(np.linspace(1, 10, 100))   # 小仰角
delays = [total_delay(P, e, phi, h, f, epsilon) * 1e9 for epsilon in elevation_angles]   #
将延迟转换为纳秒

# 绘制结果
plt.figure(figsize=(10, 6), dpi=400)
plt.plot(np.degrees(elevation_angles), delays, label="总折射延迟", color='blue')
plt.axhline(y=7.69, color='red', linestyle='--', label="天顶延迟 7.69 ns")
plt.xlabel("仰角 (度)")
plt.ylabel("总折射延迟 (纳秒)")
plt.title("小仰角 (<10°) 条件下的折射延迟随仰角变化")
plt.legend()
plt.grid(True)
plt.show()
```

Appendix 6

Introduce: 求解问题三图的 Python 代码

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Set up matplotlib to support Chinese font display
plt.rcParams['font.sans-serif'] = ['SimHei']    # Use SimHei for Chinese font display
plt.rcParams['axes.unicode_minus'] = False    # Resolve minus sign display issue

# Constants definition
c = 3.0e8    # Speed of light, in m/s
f0 = 20e9    # Reference frequency, 20 GHz
max_zenith_delay = 7.69e-9    # Maximum zenith delay, in seconds

# Define dry delay calculation using the Saastamoinen model
def dry_delay(P, z):
    """
    Calculate dry delay
    P: Ground pressure (hPa)
    z: Zenith angle (radians)
    """
    return 0.0022768 * P / np.cos(z)

# Define wet delay calculation using the Saastamoinen model
def wet_delay(e, phi, h):
    """
    Calculate wet delay
    e: Ground vapor pressure (hPa)
    phi: Latitude (radians)
    h: Height (km)
    """
    return 0.002277 * e / (1 - 0.00266 * np.cos(2 * phi) - 0.00028 * h)

# Define frequency dependence correction
```

```
def frequency_correction(delay, f):
    """
    Apply frequency correction to refraction delay
    delay: Zenith direction refraction delay (seconds)
    f: Current signal frequency (Hz)
    """
    return delay * (f0 / f) ** 2

# Define Herring mapping function
def mapping_function(epsilon, a=0.0029, b=0.0005, c=0.001):
    """
    Use Herring model mapping function to convert zenith delay to any elevation angle
delay
    epsilon: Elevation angle (radians)
    a, b, c: Empirical parameters of the Herring mapping function
    """
    return 1 / (np.sin(epsilon) + a * (np.cos(epsilon) / (1 + b / (np.sin(epsilon) + c))))

# Calculate total refraction delay
def total_delay(P, e, phi, h, f, epsilon):
    """
    Calculate total refraction delay under given conditions
    P: Ground pressure (hPa)
    e: Ground vapor pressure (hPa)
    phi: Latitude (radians)
    h: Height (km)
    f: Observation frequency (Hz)
    epsilon: Elevation angle (radians)
    """
    # Calculate zenith dry and wet delays
    tau_dry = dry_delay(P, 0)    # Zenith angle z = 0
    tau_wet = wet_delay(e, phi, h)
    tau_zenith = tau_dry + tau_wet
    # Check if zenith delay exceeds limit
    if tau_zenith > max_zenith_delay:
        tau_zenith = max_zenith_delay

    # Apply frequency correction
    tau_corrected = frequency_correction(tau_zenith, f)

    # Map to any elevation angle
    m = mapping_function(epsilon)
    return tau_corrected * m

# Set parameters
P = 1013.25    # Ground pressure, in hPa
e = 10.0         # Ground vapor pressure, in hPa
phi = np.radians(30)    # Latitude, in radians
h = 0.5    # Height, in km
f = 30e9    # Observation frequency, in Hz (assuming above 20 GHz)

# Generate elevation angle range, from 5 degrees to 90 degrees
elevation_angles = np.radians(np.linspace(10, 90, 100))
delays = [total_delay(P, e, phi, h, f, epsilon) * 1e9 for epsilon in elevation_angles]    # Convert
delay to nanoseconds

# Plot results
plt.figure(figsize=(10, 6), dpi=400)
plt.plot(np.degrees(elevation_angles), delays, label="Total Refraction Delay", color='blue')
plt.axhline(y=7.69, color='red', linestyle='--', label="Zenith Delay 7.69 ns")
plt.xlabel("Elevation Angle (Degrees)")
plt.ylabel("Total Refraction Delay (Nanoseconds)")
plt.title("Refraction Delay as a Function of Elevation Angle for High-Frequency
Observations (>20 GHz)")
plt.legend()
plt.grid(True)
plt.show()
```

Appendix 7

Introduce: 求解问题三图的 Python 代码

```
# Parameter definitions
P = 1013.25    # Atmospheric pressure (hPa)
e = 10.0         # Vapor pressure (hPa)
phi = np.radians(30)    # Latitude (radians)
h = 0.5    # Height (km)
f = 22e9    # Observation frequency (Hz)
f0 = 20e9    # Reference frequency (Hz)

# Dry delay calculation
def dry_delay(P, z):
    return 0.0022768 * P / np.cos(z)

# Wet delay calculation
def wet_delay(e, phi, h):
    return 0.002277 * e / (1 - 0.00266 * np.cos(2 * phi) - 0.00028 * h)

# GMF Mapping function
def gmf_mapping_function(epsilon, a=0.001, b=0.002, c=0.003):
    return (1 + a / (1 + b / (1 + c))) / (np.sin(epsilon) + a / (np.sin(epsilon) + b / (np.sin(epsilon) + c)))

# Calculate atmospheric delay at low elevation angles
elevation_angles = np.radians(np.linspace(1, 10, 100))    # Elevation angle range (1° to 10°)
delays = []
for epsilon in elevation_angles:
    z = np.pi / 2 - epsilon    # Zenith angle
    tau_dry = dry_delay(P, z)
    tau_wet = wet_delay(e, phi, h)
    mapping_value = gmf_mapping_function(epsilon)
    total_delay = mapping_value * (tau_dry + tau_wet) * (f0 / f) ** 2
    delays.append(total_delay * 1e9)    # Convert to nanoseconds
```

```
# Plot the results
plt.figure(figsize=(12, 6), dpi=400)
plt.plot(np.degrees(elevation_angles), np.array(delays) * 1e9, label="Atmospheric Delay (ns)", color='green')
plt.xlabel("Elevation Angle (degrees)")
plt.ylabel("Atmospheric Delay (ns)")
plt.title("Optimized Atmospheric Delay at Low Elevation Angles")
plt.grid(True)
plt.legend()
plt.show()
```

Appendix 8

Introduce: 求解问题四图的 Python 代码

```
import matplotlib.pyplot as plt

# Range of parameter variations
P_range = np.linspace(950, 1050, 50)    # Pressure range
e_range = np.linspace(5, 15, 50)    # Water vapor pressure range
epsilon_range = np.radians(np.linspace(1, 10, 50))    # Elevation angle range
f_range = np.linspace(20e9, 30e9, 50)    # Frequency range

# Sensitivity analysis function
def sensitivity_analysis():
    results = {}

    # Sensitivity to pressure
    delays_P = [np.mean(total_atmospheric_delay(elevation_angles, P, e, phi, h, f, f0)) for P in P_range]
    results['Pressure'] = {'range': P_range, 'delays': delays_P}

    # Sensitivity to water vapor pressure
    delays_e = [np.mean(total_atmospheric_delay(elevation_angles, P, e_var, phi, h, f, f0)) for e_var in e_range]
    results['Water Vapor Pressure'] = {'range': e_range, 'delays': delays_e}
    # Sensitivity to elevation angle
    delays_epsilon = [np.mean(total_atmospheric_delay([epsilon], P, e, phi, h, f, f0)) for epsilon in epsilon_range]
    results['Elevation Angle'] = {'range': np.degrees(epsilon_range), 'delays': delays_epsilon}

    # Sensitivity to frequency
    delays_f = [np.mean(total_atmospheric_delay(elevation_angles, P, e, phi, h, f_var, f0)) for f_var in f_range]
    results['Frequency'] = {'range': f_range / 1e9, 'delays': delays_f}  # Convert to GHz

    return results

# Run the sensitivity analysis
sensitivity_results = sensitivity_analysis()

# Plot sensitivity curves
plt.figure(figsize=(8, 6), dpi=400)

# Pressure sensitivity
plt.subplot(2, 2, 1)
plt.plot(sensitivity_results['Pressure']['range'], sensitivity_results['Pressure']['delays'], label='Delay vs Pressure')
plt.xlabel('Pressure (hPa)')
plt.ylabel('Delay (s)')
plt.title('Sensitivity to Pressure')
plt.grid(True)

# Water vapor pressure sensitivity
plt.subplot(2, 2, 2)
plt.plot(sensitivity_results['Water Vapor Pressure']['range'], sensitivity_results['Water Vapor Pressure']['delays'], label='Delay vs Water Vapor Pressure', color='orange')
plt.xlabel('Water Vapor Pressure (hPa)')
plt.ylabel('Delay (s)')
plt.title('Sensitivity to Water Vapor Pressure')
plt.grid(True)

# Elevation angle sensitivity
plt.subplot(2, 2, 3)
plt.plot(sensitivity_results['Elevation Angle']['range'], sensitivity_results['Elevation Angle']['delays'], label='Delay vs Elevation Angle', color='green')
plt.xlabel('Elevation Angle (degrees)')
plt.ylabel('Delay (s)')
plt.title('Sensitivity to Elevation Angle')
plt.grid(True)

# Frequency sensitivity
plt.subplot(2, 2, 4)
plt.plot(sensitivity_results['Frequency']['range'],
sensitivity_results['Frequency']['delays'], label='Delay vs Frequency', color='red')
plt.xlabel('Frequency (GHz)')
plt.ylabel('Delay (s)')
plt.title('Sensitivity to Frequency')
plt.grid(True)

plt.tight_layout()
plt.show()
```