Problem Chosen	2024 ShuWei Cup Summary Sheet	Team Control Number
A		2024110430327

Frequency Estimation Algorithms for Airspeed Measurement: From Statistical to Sparse Methods

Summary

Using lasers to determine a vehicle's operational status is an effective method for measuring velocity. Analyzing the frequency variation of the feedback signal enables real-time calculation of the Doppler shift, which is used to deduce airspeed. It is an effective approach to estimate the frequency of the feedback signal through the utilisation of **statistical signal processing** and **compressed sensing** methodologies.

In response to the initial query, the noise signal is readily obtained and subjected to an in-depth examination in **both the time and frequency domains**. This analysis leads to the conclusion that the nature of the noise is akin to **Gaussian white noise**. Furthermore, the signal's stationarity is corroborated through the utilisation of the **Short-Time Fourier Transform**.

In the second question, we discuss the noise by dividing it into Gaussian white noise and non-Gaussian white noise. We then derive a **maximum likelihood estimation** of the frequency, as well as a **method of moments** based on the implementation of the autocorrelation function, which implements the estimation of the frequency. Concurrently, an exact value of **40967375.553611** Hz for the frequency is sought through the application of a **particle swarm intelligent optimisation algorithm** to a preliminary estimate derived from the aforementioned method.

The third question builds upon the second question, wherein we jointly **develop an iterative estimation algorithm based on the gradient descent method**. This is founded upon the concept of alternating optimisation, utilising both **MLE method and the method of moments**. This algorithm accurately estimates the magnitude and phase of the signal, as well as the frequency. Upon applying this algorithm, we obtain a result of **32099623.112227** Hz.

The signals in the fourth question are **undersampled** signals, and we employ **compressed sensing techniques** to reconstruct the undersampled signals to a limited extent in each time interval. This allows us to apply the algorithms of the previous two questions for accurate estimation. Furthermore, we analyse and compare the results without using the compressed sensing technique and the estimation results from the three CS-based signal reconstruction algorithms. Ultimately, we determine that the frequency of the signal is **54890219.560878** Hz.

It is essential to obtain an accurate estimation of the frequency of the signal. In this paper, we have conducted preliminary research on frequency estimation for airspeed measurement and have made significant progress towards fulfilling the task of the topic.

Key word: Frequency Estimation, Statistical Signal Processing, Compressed Sensing, Airspeed measurement

Content

Content **2**
1. Introduction **3**
1.1 Background 3
1.2 Our work 3
2. Problem Analysis **3**
2.1 Analysis of Question One 3
2.2 Analysis of Question Two 4
2.3 Analysis of Question Three 4
2.4 Analysis of Question Four 4
3. Symbol Description and Assumptions **4**
4. Model **5**
4.1 Model of Question One 5
4.1.1 *Model* 5
4.1.2 *Experiment* 6
4.2 Model of Question Two 7
4.2.1 *Model* 7
4.2.2 *Experiment* 13
4.3 Model of Question Three 16
4.3.1 *Model* 16
4.3.2 *Experiment* 18
4.4 Model of Question Four 20
4.4.1 *Model* 20
4.4.2 *Experiment* 21
5. Strengths and Weakness **25**
5.1 Strengths 25
5.2 Weakness 25
References **26**
Appendix **27**

1. Introduction

1.1 Background

Airspeed is defined as the speed of an aircraft in relation to the surrounding air. It is a crucial parameter to be monitored in real time during flight. The most prevalent method for measuring airspeed is laser speed measurement. This technique relies on the emission of laser signals by a device on the aircraft into the surrounding air. Upon encountering aerosol particles in the air, the laser generates a scattering effect, resulting in a Doppler shift due to the movement of particles with the airflow. The signal-receiving and processing devices on board the aircraft are analysed in order to capture the frequency change of the scattered laser signals, which are then used to calculate the Doppler shift. Based on this calculation, the airspeed is deduced in real time.

1.2 Our work

Task 1: Once the amplitude, frequency and phase of the non-noise portion of the received signal have been determined, an analysis of the noise characteristics can be conducted based on the data provided in the question.

Task 2: Given the amplitude and phase of the non-noise portion of the received signal, the objective is to devise a method for estimating the frequency of the signal, based on the available data.

Task 3: In practice, the amplitude, frequency and phase of the received signal are typically unknown. Therefore, it is necessary to estimate the frequency of the actual signal using the data provided in the question.

Task 4: In order to circumvent the potential for signal interference, intermittent reception is typically employed, which may result in the loss of data and a reduction in the overall usability of the information. Consequently, it is necessary to estimate the frequency based on the data provided by the undersampled signal, as outlined in the title.

2. Problem Analysis

2.1 Analysis of Question One

Given the known amplitude, frequency, and phase of the signal, the noise signal can be readily extracted from the received signal. The noise signal will be analysed from both a time and frequency perspective, with a particular focus on the amplitude of the noise and its power spectral density profile. To determine the stationarity of the noise, the short-time Fourier transform will

be employed.

2.2 Analysis of Question Two

In order to address this issue, a variety of algorithms have been considered with the aim of estimating the frequency at two levels. These levels correspond to the presence of Gaussian white noise and non-Gaussian white noise. Maximum likelihood estimation has been identified as a potential approach for providing an initial estimation result. This can be complemented by the use of particle swarm intelligent optimisation algorithms, which are capable of conducting a more detailed search based on this estimation. Additionally, the method of moments offers the possibility of obtaining an estimator that is not dependent on the characteristics of the noise.

2.3 Analysis of Question Three

This problem can be regarded as a generalised form of the previous problem, given that the amplitude and phase of the signal remain unknown. In order to incorporate the concept of alternating optimisation into the estimation of the magnitude, phase and frequency, we have combined the method of moments from the previous problem with the maximum likelihood estimation method. This has enabled us to develop an estimation algorithm based on the gradient descent method, which will facilitate the achievement of an accurate estimation of the frequency. It should be acknowledged that the algorithm must take into account the constraint that the mean of the noise is zero.

2.4 Analysis of Question Four

In light of the challenge posed by the deterioration in estimator performance resulting from the undersampling of signals, we place our trust in the potential of compression-aware techniques to transform the undersampled signals into more comprehensive ones. This approach aims to enhance the amount of data available, thereby improving the performance of the estimators in the second and third questions. Conversely, the data set from the initial question can be generated using our proposed algorithm to demonstrate the viability of this approach.

3. Symbol Description and Assumptions

Please refer to Appendix for further details, as the information is limited to the length of the page.

4. Model

4.1 Model of Question One

4.1.1 Model

The signal model under consideration is that of a sinusoidal signal, with the noise being of an additive nature. In order to gain insight into the properties of the interfering noise, it is first necessary to separate the noise signal from the receiver signal $x(t)$:

$$x(t) = A\sin(2\pi f_0 t + \phi) + z(t) \tag{1}$$

Therefore, we have

$$z(t) = x(t) - A\sin(2\pi f_0 t + \phi) \tag{2}$$

If we take the sampling frequency to be $f_s = \frac{1}{T_s}$ and discretise the aforementioned signal, we obtain:

$$z[n] = x[n] - A\sin(2\pi f_0 n + \phi), n = 0, 1, \cdots, N-1 \tag{3}$$

The properties of the noise affecting the sequence $z[n]$ can be determined by analysing the data in both the time and frequency domains. Firstly, the mean and variance of the noise signal $z[n]$ can be estimated as follows:

$$\begin{cases} \mu_z = \dfrac{1}{N}\displaystyle\sum_{n=0}^{N-1} z[n] \\ \sigma_z^2 = \dfrac{1}{N-1}\displaystyle\sum_{n=0}^{N-1}(z[n] - \mu_z)^2 \end{cases} \tag{4}$$

Furthermore, the autocorrelation function of the signal can be employed to ascertain the degree of stationary exhibited by the noise. Thus, we have:

$$R_{zz}[l] = E\big[z[n]z[n-l]\big] \tag{5}$$

Secondly, the spectral distribution of the noise can be estimated using the Fourier transform to obtain the power spectral density of the noise signal. In regard to white noise, the frequency components demonstrate a uniform distribution, and the power spectrum is stable and constant. Since our noise signal is processed in the discrete domain, the Fourier transform can be implemented using the Fast Fourier Transform FFT. That means

$$Z[k] = \sum_{n=0}^{N-1} z[n]e^{-i\frac{2\pi}{N}nk}, k = 0, 1, \cdots, N-1 \tag{6}$$

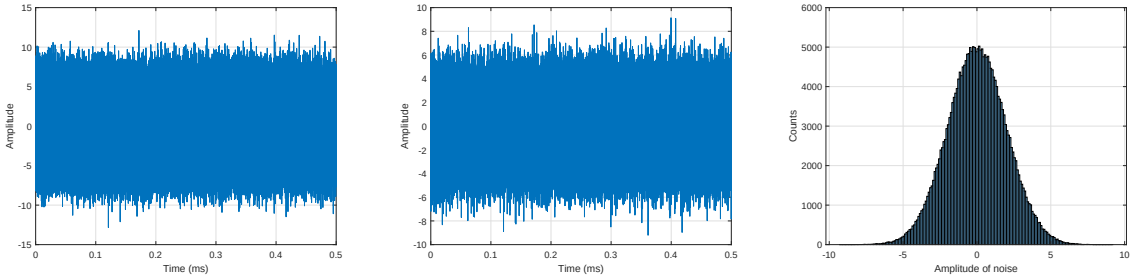
In the case of a stationary noise signal, the power spectral density of a wide-sense stochastic process can be determined by applying the Wiener-Khinchin theorem, which states that the Fourier transform of the autocorrelation function is equal to the power spectral density. Thus we have

$$S_{zz}(f) = \sum_{k=-\infty}^{\infty} R_{zz}[k]e^{-i2\pi f} = \sum_{k=-\infty}^{\infty} E\big[z[n]z[n-k]\big]e^{-i2\pi f} \tag{7}$$

Ultimately, time-frequency analysis of the noise signal can provide further insight into the stationary of the noise signal. Additionally, the amplitude distribution of the statistical noise can elucidate the distributional characteristics of the noise signal.

4.1.2 Experiment

Figure 1 illustrates the time-domain waveforms of the received signal and the noise signal, obtained by applying Eq.(3). Additionally, the figure depicts the amplitude distribution of the noise.



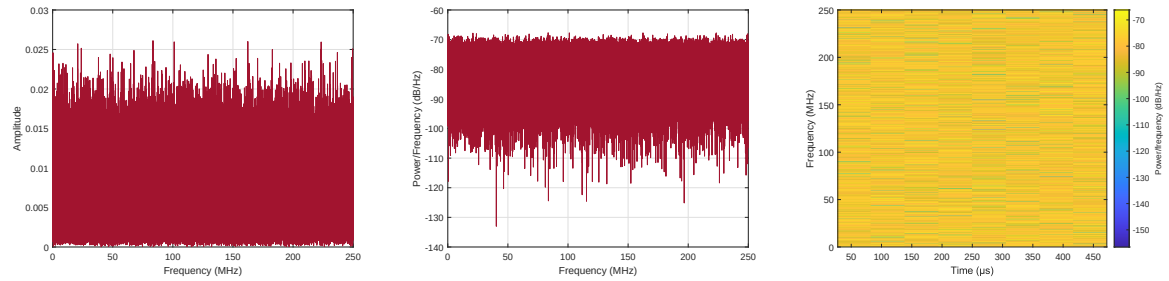
(a) Received signal **(b) Noise signal** **(c) Distribution of noise**

Figure 1 The time domain waveform of the noise signal and its amplitude distribution.

It is evident that sinusoidal signals are essentially obliterated in a noisy setting, which underscores the limitations of relying on the time-domain waveform to directly identify useful signals. Furthermore, the amplitude distribution of the noise exhibits a discernible normal distribution, which suggests that the noise signal under analysis is likely to be Gaussian white noise.

From the distributional properties of the noise in Fig.1, we can easily calculate the mean $\mu_z = 0.005839$ and the variance $\sigma_z^2 = 3.981902$ of the noise sequence $z[n]$. This demonstrates that the mean of the noise in question tends towards zero, which serves to corroborate the assertion that Gaussian white noise is always zero-mean.

Furthermore, given that Gaussian white noise is a stationary signal, it is essential to investigate the spectrum and power spectral density of the noise signal. In Figure 2, the spectrum of the noise signal, the power spectral density curve, and the results of the short-time Fourier transform of the noise signal are presented.



(a) Amplitude spectrum **(b) Power spectral density** **(c) STFT result**

Figure 2 The noise signal is subjected to an analysis in the frequency domain, with the objective of determining the noise characteristics.

This is unequivocal from the results presented in Fig.2, which demonstrate that the noise occupies the entire frequency domain and exhibits a uniform distribution. This is corroborated by the power spectral density curve. In contrast, the time-frequency analysis, specifically the short-time Fourier transform, indicates that the noise signal retains a consistent frequency-domain profile across multiple time slices. This suggests that the noise signal can be considered stationary.

The combination of the results obtained from the aforementioned time and frequency domain analyses allows us to conclude with a high degree of certainty that the noise signal that was separated from the received signal is, in essence, a Gaussian white noise signal. The mean value of this noise tends towards zero, while the power spectrum remains approximately constant.

4.2 Model of Question Two

4.2.1 Model

In order to obtain an accurate estimate of the frequency of the received signal, it is necessary to make reasonable assumptions about the characteristics of the noise in advance. The majority of noise observed in nature is of a Gaussian white noise type, and it has become a principal area of investigation. Furthermore, we consider the remaining noise sources. However, this would require a significant amount of work if we were to incorporate the probability density functions of all potential noise sources into the estimators we construct. Accordingly, we categorise the noise z into two distinct types: Gaussian white noise and non-Gaussian white noise. In the case of non-Gaussian white noise, while the amplitude distribution of the noise may still adhere to a Gaussian distribution, the power spectrum of the noise may no longer be a constant, resulting in what is known as Gaussian colour noise.

The received signal model can be rewritten in vector form from Eq.(1):

$$\mathbf{x} = g(\boldsymbol{\theta}) + \mathbf{z} \tag{8}$$

In this context, $\boldsymbol{\theta}$ represents the parameter to be estimated, while the function $g(\cdot)$ describes the relationship between this parameter vector and the received signal $\mathbf{x}$. In this particular question, the parameter vector $\boldsymbol{\theta}$ will be reduced to a single dimension due to the fact that the amplitude and phase of the useful signal are already known. Consequently, only the frequency of the signal needs to be estimated.

■ If $\mathbf{z}$ is Gaussian white noise

Prior to commencing parameter estimation, it is essential to ascertain the Cramer-Rao lower bounds for the parameters, particularly in instances where the noise is Gaussian white noise. This enables the acquisition of a definitive closed-form solution with respect to the CRLB[1]. Therefore, we have

$$\operatorname{var}(\hat{f}_0) \geq \frac{\sigma_z^2}{\sum_{n=0}^{N-1} \left(\frac{\partial g[n;f_0]}{\partial f_0}\right)^2} \tag{9}$$

where $g[n;f_0] = A\sin(2\pi f_0 n + \phi)$, and

$$\operatorname{var}(\hat{f}_0) \geq \frac{\sigma_z^2}{A^2 \sum_{n=0}^{N-1} \left[2\pi n \cos(2\pi f_0 n + \phi)\right]^2} \tag{10}$$

Given the known amplitude and phase of the signal, we may assume that the desired signal is obtained from a fundamental frequency signal modulated by a carrier. This suggests that the frequency f_0 is likely located in the vicinity of frequencies from 0 to 0.5 Hz. Consequently, we may obtain a CRLB for the estimation of sinusoidal frequencies, as illustrated in Fig.3.

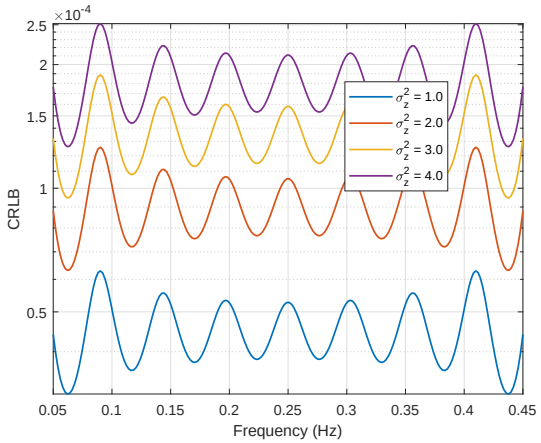


Figure 3 CRBL for sinusoidal frequency estimation.

It can be seen from Eq.(10) that if all the received data is used for estimation, the curve in Fig.3 will decrease abruptly to the point that this CRLB is no longer sensitive to changes in frequency. Conversely, if the modulating signal is estimated directly, the high frequency will likewise invalidate the solution of the CRLB, since the calculated CRLB is quite small.

In order to obtain an accurate estimation of the signal frequency f_0, especially after modulation, a two-stage strategy is employed, comprising coarse estimation followed by fine estimation. Firstly, the conventional frequency estimation method will provide an initial indication of the possible locations of f_0. Subsequently, the intelligent search algorithm will perform a further refinement of the value of f_0.

The method of maximum likelihood estimation was used to obtain a rough estimate of f_0. The objective is to identify an estimate, $\hat{f_0}$, of f_0 such that the following equation attains a minimum value:

$$\hat{f_0} = \arg\min_{f_0} J(f_0) = \arg\min_{f_0} \sum_{n=0}^{N-1} \left(x[n] - A\sin(2\pi f_0 n + \phi)\right)^2 \tag{11}$$

Let $\phi = \phi_0 + \frac{\pi}{2}$, we have

$$\begin{aligned} J(f_0) &= \sum_{n=0}^{N-1} \left(x[n] - A\sin(2\pi f_0 n + \phi_0 + \frac{\pi}{2})\right)^2 \\ &= \sum_{n=0}^{N-1} \left(x[n] - A\cos(2\pi f_0 n + \phi_0)\right)^2 \\ &= \sum_{n=0}^{N-1} \left(x[n] - A\cos\phi_0\cos 2\pi f_0 n + A\sin\phi_0 \sin 2\pi f_0 n\right)^2 \end{aligned} \tag{12}$$

And let $\alpha_1 = A\cos\phi_0, \alpha_2 = -A\sin\phi_0$, then

$$\begin{aligned} J' &= (\mathbf{x} - \alpha_1\mathbf{c} - \alpha_2\mathbf{s})^T(\mathbf{x} - \alpha_1\mathbf{c} - \alpha_2\mathbf{s}) \\ &= (\mathbf{x} - \mathbf{H}\boldsymbol{\alpha})^T(\mathbf{x} - \mathbf{H}\boldsymbol{\alpha}) \end{aligned} \tag{13}$$

Where,

$$\boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 & \alpha_2 \end{bmatrix}^T$$

$$\mathbf{H} = \begin{bmatrix} \mathbf{c} & \mathbf{s} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \cos 2\pi f_0 & \sin 2\pi f_0 \\ \vdots & \vdots \\ \cos 2\pi f_0(N-1) & \sin 2\pi f_0(N-1) \end{bmatrix}$$

This is a quadratic function minimisation problem, and we easily find that solution as:

$$\hat{\boldsymbol{\alpha}} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{x} \tag{14}$$

Then we have

$$J' = \mathbf{x}^T \left(\mathbf{I} - \mathbf{H}(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\right)\mathbf{x} \tag{15}$$

To minimise the above equation is actually equivalent to maximising the following equation:

$$\begin{aligned} \hat{f_0} &= \arg\max_{f_0} \mathbf{x}^T\mathbf{H}(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{x} \\ &= \arg\max_{f_0} \begin{bmatrix} \mathbf{c}^T\mathbf{x} \\ \mathbf{s}^T\mathbf{x} \end{bmatrix}^T \cdot \begin{bmatrix} \mathbf{c}^T\mathbf{c} & \mathbf{c}^T\mathbf{s} \\ \mathbf{s}^T\mathbf{c} & \mathbf{s}^T\mathbf{s} \end{bmatrix}^{-1} \cdot \begin{bmatrix} \mathbf{c}^T\mathbf{x} \\ \mathbf{s}^T\mathbf{x} \end{bmatrix} \end{aligned} \tag{16}$$

Notice that f_0 is not near 0 or 0.5 Hz, we have:

$$\begin{aligned} \frac{1}{N}\mathbf{c}^T\mathbf{s} &= \frac{1}{N}\sum_{n=0}^{N-1} \cos 2\pi f_0 n \sin 2\pi f_0 n \\ &= \frac{1}{2N}\sum_{n=0}^{N-1} \sin 4\pi f_0 n \\ &\approx 0 \end{aligned} \tag{17}$$

Similarly,

$$\frac{\mathbf{c}^T\mathbf{c}}{N} \approx \frac{1}{2}, \frac{\mathbf{s}^T\mathbf{s}}{N} \approx \frac{1}{2}$$

Then we have

$$\begin{aligned} \hat{f_0} &= \arg\max_{f_0} \begin{bmatrix} \mathbf{c}^T\mathbf{x} \\ \mathbf{s}^T\mathbf{x} \end{bmatrix}^T \cdot \begin{bmatrix} \mathbf{c}^T\mathbf{c} & \mathbf{c}^T\mathbf{s} \\ \mathbf{s}^T\mathbf{c} & \mathbf{s}^T\mathbf{s} \end{bmatrix}^{-1} \cdot \begin{bmatrix} \mathbf{c}^T\mathbf{x} \\ \mathbf{s}^T\mathbf{x} \end{bmatrix} \\ &= \arg\max_{f_0} \begin{bmatrix} \mathbf{c}^T\mathbf{x} \\ \mathbf{s}^T\mathbf{x} \end{bmatrix}^T \cdot \begin{bmatrix} \frac{N}{2} & 0 \\ 0 & \frac{N}{2} \end{bmatrix}^{-1} \cdot \begin{bmatrix} \mathbf{c}^T\mathbf{x} \\ \mathbf{s}^T\mathbf{x} \end{bmatrix} \\ &= \arg\max_{f_0} \frac{2}{N}\left[(\mathbf{c}^T\mathbf{x})^2 + (\mathbf{s}^T\mathbf{x})^2\right] \\ &= \arg\max_{f_0} \frac{2}{N}\left|\sum_{n=0}^{N-1} x[n]e^{-i2\pi f_0 n}\right|^2 \end{aligned} \tag{18}$$

This indicates that the result of a maximum likelihood estimate of the frequency f_0 can be obtained by seeking the frequency that corresponds to the peak of the power spectral density of the received signal $\mathbf{x}$.

Particle swarm optimisation algorithms are employed to facilitate the fine search component of the estimation algorithm. The fundamental premise of this optimisation algorithm is to randomly disperse a specified number of solution particles, collectively search the solution space, and leverage the interconnectivity between individual and group optima to mitigate the impact of potential local optima[2]. The algorithmic flowchart is presented in Fig.4.

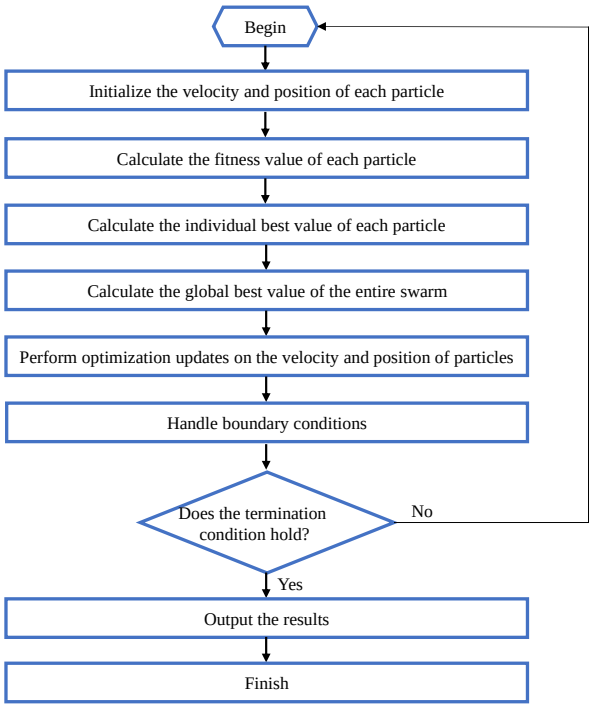


Figure 4 Flowchart of particle swarm intelligent optimization algorithm.

In order to address the issue at hand, it is necessary to conduct a meticulous examination of the approximated value of f_0, as derived from MLE. Given that the noise in question is presumed to be of a Gaussian white noise nature, with a mean value of zero, the RMSE is employed as the objective function in the optimisation algorithm. Thus we have

$$\hat{f_0} = \arg\min_{f_0} \sqrt{\frac{1}{N}\sum_{n=0}^{N-1}\left(x[n] - A\sin(2\pi f_0 n + \phi)\right)^2} \tag{19}$$

■ If **z** is non-Gaussian white noise

As previously stated, the noise present in the received signal may not be of a Gaussian white noise nature. However, it is highly probable that it still exhibits Gaussian characteristics and retains the zero-mean property. In fact, our received signal model should be rewritten from Eq.(1) as:

$$x[n] = A\sin(2\pi f_0 n + \phi) + E(z[n]) + z'[n], n = 0, 1, \cdots, N-1 \tag{20}$$

where $z'[n]$ is the 'zero mean' noise. This is due to the fact that the noise in question retains the property of zero-mean $E(z[n]) = 0$, as evidenced by both Eq.(11) and Eq.(19). This remains true even when considering non-Gaussian white noise.

In order to streamline the calculations, the same phase transformation as that described in Eq.12 is employed. Let $\phi = \phi_0 + \frac{\pi}{2}$, we have

$$\begin{aligned} x[n] &= A\sin(2\pi f_0 n + \phi) + z[n] \\ &= A\sin(2\pi f_0 n + \phi_0 + \frac{\pi}{2}) + z[n] \\ &= A\cos(2\pi f_0 n + \phi_0) + z[n] \end{aligned} \tag{21}$$

The autocorrelation function of the received signal is:

$$r_{xx}[k] = E\big[x[n]x[n+k]\big] = r_{ss}[k] + r_{zz}[k] \tag{22}$$

Where,

$$\begin{aligned} r_{ss}[k] &= E\big[\big(A\cos(2\pi f_0 n + \phi_0)\big)\cdot\big(A\cos(2\pi f_0[n+k] + \phi_0)\big)\big] \\ &= E\big[A^2\cos(2\pi f_0 n + \phi_0)\cos(2\pi f_0 n + 2\pi f_0 k + \phi_0)\big] \\ &= E\big[\frac{A^2}{2}\cos(4\pi f_0 n + 2\pi f_0 k + 2\phi_0) + \frac{A^2}{2}\cos 2\pi f_0 k\big] \end{aligned} \tag{23}$$

Let $k = 1$, we have

$$\begin{aligned} r_{ss}[1] &= E\big[\frac{A^2}{2}\cos(4\pi f_0 n + 2\pi f_0 + 2\phi_0) + \frac{A^2}{2}\cos 2\pi f_0\big] \\ &= E\big[\frac{A^2}{2}\cos(4\pi f_0 n + 2\pi f_0 + 2\phi_0)\big] + \frac{A^2}{2}\cos 2\pi f_0 \end{aligned} \tag{24}$$

For $k = 1$, a suitable estimation of the ACF is:

$$\begin{aligned} \hat{r}_{ss}[1] &= \frac{1}{N-1}\sum_{n=0}^{N-2}\big(A\cos(2\pi f_0 n + \phi_0)\big)\cdot\big(A\cos(2\pi f_0[n+k] + \phi_0)\big) \\ &= \frac{1}{N-1}\sum_{n=0}^{N-2}\frac{A^2}{2}\cos(4\pi f_0 n + 2\pi f_0 + 2\phi_0) + \frac{1}{N-1}\sum_{n=0}^{N-2}\frac{A^2}{2}\cos 2\pi f_0 \end{aligned} \tag{25}$$

Given that the function $\cos(4\pi f_0 n + 2\pi f_0 + 2\phi_0)$ is periodic, it follows that

$$\lim_{N\to\infty}\frac{1}{N-1}\sum_{n=0}^{N-2}\frac{A^2}{2}\cos(4\pi f_0 n + 2\pi f_0 + 2\phi_0) = 0 \tag{26}$$

And,

$$\begin{aligned} \hat{r}_{ss}[1] &= \frac{1}{N-1}\sum_{n=0}^{N-2}\frac{A^2}{2}\cos 2\pi f_0 = \frac{A^2}{2}\cos 2\pi f_0 \\ &= r_{ss}[1] \\ &\Rightarrow E\big[\frac{A^2}{2}\cos(4\pi f_0 n + 2\pi f_0 + 2\phi_0)\big] = 0 \end{aligned} \tag{27}$$

Therefore,

$$r_{xx}[1] = \frac{A^2}{2}\cos 2\pi f_0 + r_{zz}[1] = \frac{1}{N-1}\sum_{n=0}^{N-2} x[n]x[n+1] \tag{28}$$

From this, an estimate of the frequency f_0 can be derived:

$$\hat{f_0} = \frac{1}{2\pi}\arccos\left(\frac{2\sum_{n=0}^{N-2} x[n]x[n+1] - 2(N-1)r_{zz}[1]}{A^2(N-1)}\right) \tag{29}$$

It should be noted that the frequency obtained in this instance has been normalised and thus requires multiplication by the sampling frequency in order to obtain the original signal frequency. Equation (29) is derived using the method of moments. In the absence of certainty regarding the properties of the noise, it is necessary to make an estimate of the autocorrelation function of the noise, that is to say, to estimate the value of $r_{zz}[1]$.

4.2.2 Experiment

Despite the use of disparate methodologies to ascertain the frequency f_0, contingent on the noise characteristics, the processing of the actual data invariably commences with an analysis of the received signal in terms of Gaussian white noise. As illustrated in Fig.5, the power spectral density of the **x** was obtained.

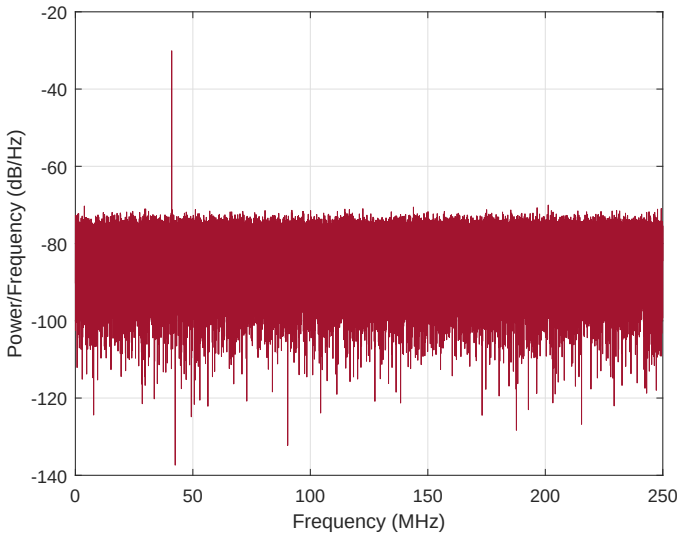
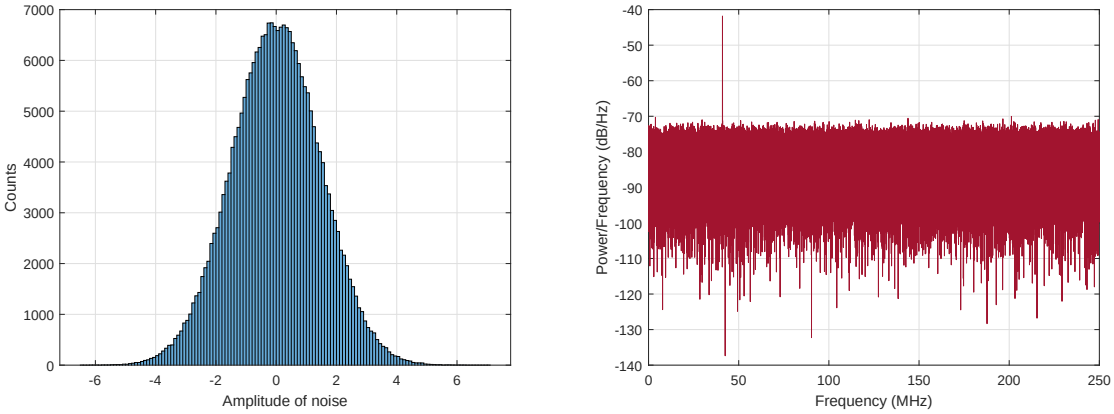


Figure 5 Power spectral density profile of the received signal x.

We seem to be able to determine that the type of noise is Gaussian white noise, as the power spectrum curve shows an impulse-shaped peak. And on the rest of the frequencies, the signal gets a large attenuation. With Eq.(18), we choose to take the frequency at which the peak on the power spectral density curve is located as the result of the maximum likelihood estimation.

Then, again following the processing model of Eq.(3), we evaluate the amplitude distribution of the remaining noise signal and its power spectral density after excluding the estimated monotone signal from the received signal, as shown in Fig.6.



(a) Distribution of amplitude **(b) PSD**

Figure 6 The suspected amplitude distribution of the noise signal and the suspected power spectral density of the noise signal.

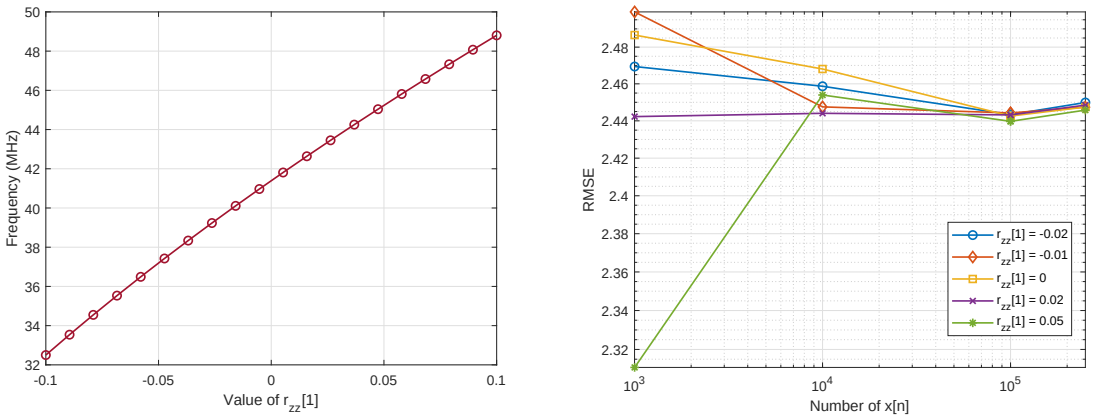
It can be stated with certainty that the noise in question still obeys a Gaussian distribution; however, it is no longer akin to white noise. This is due to the fact that its power spectral density profile is no longer constant, but rather aligns with that of the received signal, exhibiting a decline from -30 dB/Hz to -41 dB/Hz at the peak.

Surprisingly, the frequencies corresponding to the peaks of the power spectral density curves of the noise signals are the very frequencies that we have previously obtained from the MLE estimates. This shows that the noisy signal follows a Gaussian distribution, but has the same frequency as the useful signal and is also superimposed with white noise.

On the other hand, the frequency of the signal can be estimated using Eq.(29), independent of the power spectral density. However, at the same time, the parameter $r_{zz}[1]$ in Eq.(29) controls the accuracy of the frequency estimation. Figure 7 illustrates the trend of the transformation of the frequency estimation results with $r_{zz}[1]$. Furthermore, the effect of the dependence of Eq.(29) on the amount of data is evaluated by taking different $r_{zz}[1]$, and the results are also shown in Fig.7. These results are expressed in terms of the RMSE defined in Eq.(19).

The rationale behind setting the parameter $r_{zz}[1]$ to a relatively low value is that the results presented in Fig.6 demonstrate that the noisy signal still exhibits the majority of the characteristics associated with Gaussian white noise. This is because the autocorrelation function of Gaussian white noise can be expressed as:

$$r_{zz}[k] = \sigma_{zz}^2 \delta[k] = \begin{cases} \sigma_{zz}^2, & k = 0 \\ 0, & k \neq 0 \end{cases} \tag{30}$$



(a) **(b)**

Figure 7 Exploring the estimation performance of Eq.(29). (a) Variation of the parameter $r_{zz}[1]$ induces an increase in the estimated frequency. (b) The effect of N on the estimation accuracy for different parameter $r_{zz}[1]$.

As illustrated in Fig.7, an increase in $r_{zz}[1]$ is associated with a corresponding rise in the estimated frequency. Conversely, a larger $r_{zz}[1]$ may yield an estimation result that is less reliant on extensive data. Nevertheless, the MLE approximation for the estimated frequency is most closely aligned with the result when $r_{zz}[1]$ is equal to -0.02.

In order to obtain a more accurate estimate, the particle swarm intelligent optimisation algorithm previously referenced is employed to identify the desired frequency. The range of the desired frequency is between 40 MHz and 41 MHz, and the iteration curve of the algorithm is presented in Figure 8.

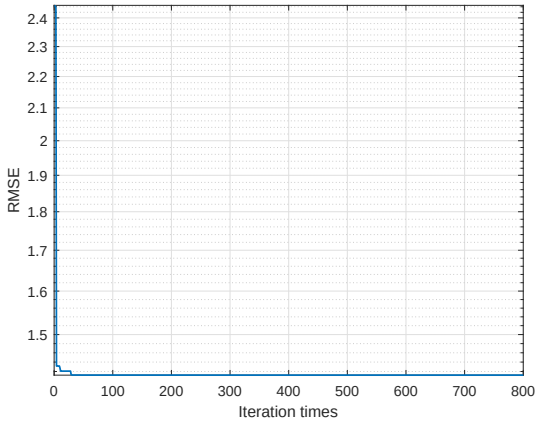


Figure 8 Iteration curves for particle swarm intelligent optimisation algorithms.

In the final analysis, the estimation results of the MLE are compared with those of the MLE-assisted particle swarm intelligent optimisation algorithm and the direct estimation results from Eq.(29), as illustrated in Table 1.

Table 1 The results of the estimation of the signal frequency by the three methods.

Methods	$\hat{f}_0$ (Hz)
MLE	40999836.000656
MLE + PSO	40999999.908274
By Eq.(29)	40967375.553611

4.3 Model of Question Three

4.3.1 Model

The estimator presented in Eq.(29) demonstrates satisfactory estimation performance and does not necessitate significant concern regarding the characterisation of the noise. Accordingly, in addressing this issue, we proceed with the construction of a new estimator based on the methodology outlined in Eq.(29).

Firstly, the amplitude of the signal is unknown, so we need to estimate the amplitude A first. Although the noise characteristics are no longer distinguished in detail, the results of the maximum likelihood estimation (MLE) of frequency can be employed to obtain an estimate of the signal amplitude.

In the event that an estimate of the frequency f_0 has already been obtained, it is possible to derive an estimate of the magnitude in accordance with the principles set out in equation (14). That means

$$\hat{\boldsymbol{\alpha}} \approx \frac{2}{N}\begin{bmatrix}\hat{\mathbf{c}}^T\mathbf{x}\\ \hat{\mathbf{s}}^T\mathbf{x}\end{bmatrix} = \begin{bmatrix}\frac{2}{N}\sum_{n=0}^{N-1} x[n]\cos 2\pi\hat{f}_0 n\\ \frac{2}{N}\sum_{n=0}^{N-1} x[n]\sin 2\pi\hat{f}_0 n\end{bmatrix} \tag{31}$$

Thereby, the magnitude A is estimated as:

$$\hat{A} = \sqrt{\hat{\alpha}_1^2 + \hat{\alpha}_2^2} = \frac{2}{N}\left|\sum_{n=0}^{N-1} x[n]e^{-i2\pi\hat{f}_0 n}\right| \tag{32}$$

Also, the phase ϕ_0 can be estimated:

$$\hat{\phi}_0 = \arctan\left(\frac{-\sum_{n=0}^{N-1} x[n]\sin 2\pi\hat{f}_0 n}{\sum_{n=0}^{N-1} x[n]\cos 2\pi\hat{f}_0 n}\right) \tag{33}$$

And we have:

$$\hat{\phi} = \hat{\phi}_0 + \frac{\pi}{2} = \arctan\left(\frac{-\sum_{n=0}^{N-1} x[n]\sin 2\pi\hat{f}_0 n}{\sum_{n=0}^{N-1} x[n]\cos 2\pi\hat{f}_0 n}\right) + \frac{\pi}{2} \tag{34}$$

By combining the results of equations (19), (29), (32) and (34), an estimation algorithm based on the alternating optimisation concept can be constructed:

$$\begin{cases} \mathcal{J}(A,\phi,f_0) = \sqrt{\dfrac{1}{N}\displaystyle\sum_{n=0}^{N-1}\left(x[n]-A\sin(2\pi f_0 n+\phi)\right)^2} \\ A^{k+1} = \arg\min\limits_{A} \mathcal{J}(A, f_0^k, \phi^k) \\ \phi^{k+1} = \arg\min\limits_{\phi} \mathcal{J}(A^k, f_0^k, \phi) \\ f_0^{k+1} = \arg\min\limits_{f_0} \mathcal{J}(A^k, f_0, \phi^k) \end{cases} \tag{35}$$

It is reasonable to conclude that gradient descent represents an effective approach to solving the alternating optimisation problem. The following algorithm represents a potential solution:

Algorithm 1: An alternative optimisation method based on the gradient descent technique was employed in order to obtain the most accurate frequency estimation.

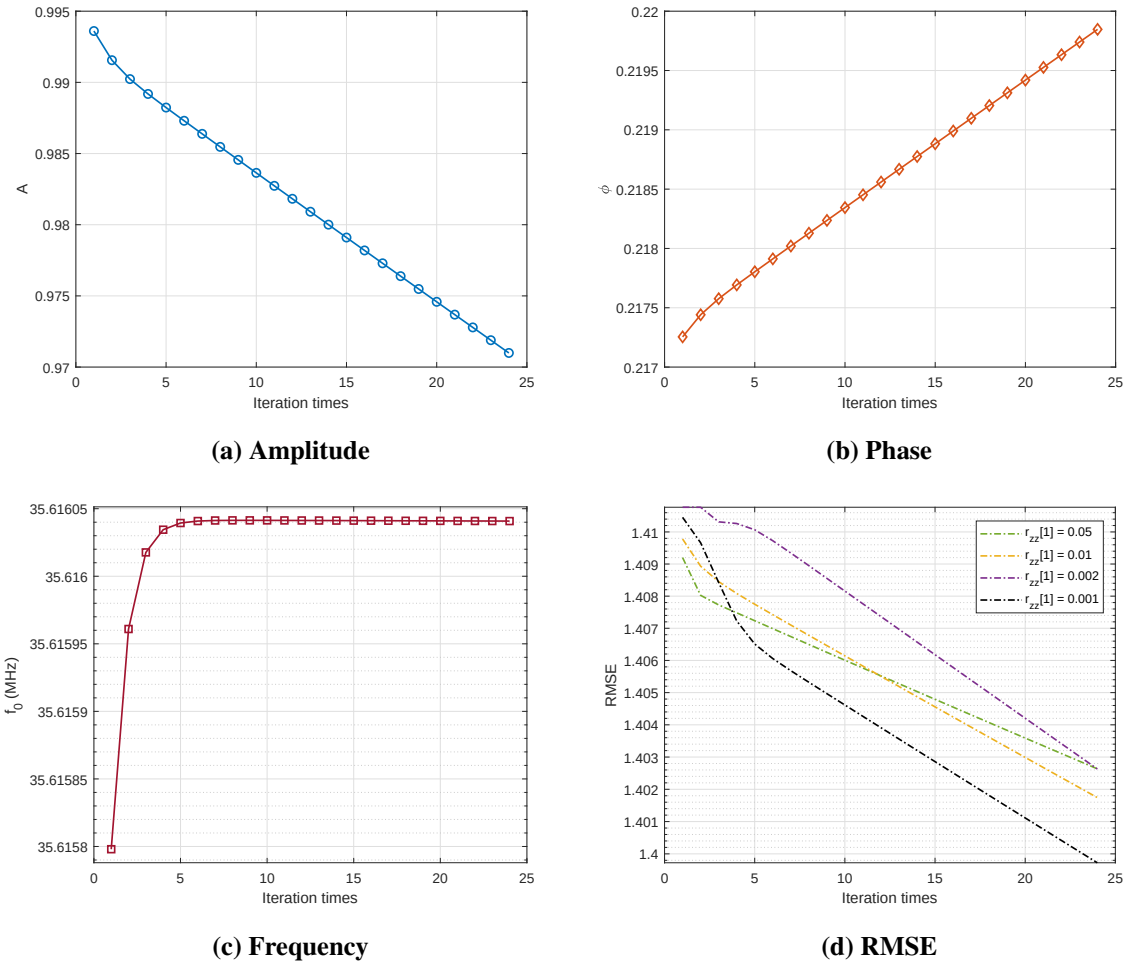
Input: Convergence tolerance ϵ, Maximum iterations K, Learning rate $\eta_A, \eta_\phi, \eta_{f_0}$

Output: A, ϕ, f_0

Initialize variables: $f_0^0 \leftarrow$ Eq.(18), $A^0 \leftarrow$ Eq.(32), $\phi^0 \leftarrow$ Eq.(34);
for $k = 0, 1, 2, \cdots, K-1$ **do**
  $A^{k+1} = A^k - \eta_A \cdot \frac{\partial \mathcal{J}}{\partial A}$;
  $\phi^{k+1} = \phi^k - \eta_\phi \cdot \frac{\partial \mathcal{J}}{\partial \phi}$;
  $f_0^{k+1} = f_0^k - \eta_{f_0} \cdot \frac{\partial \mathcal{J}}{\partial f_0}$;
  where,
  $\frac{\partial \mathcal{J}}{\partial A} = -\frac{1}{N\mathcal{J}} \sum_{n=0}^{N-1} \left(x[n] - A\sin(2\pi f_0^k n + \phi^k)\right)^2 \cdot \sin(2\pi f_0^k n + \phi^k)$;
  $\frac{\partial \mathcal{J}}{\partial \phi} = -\frac{1}{N\mathcal{J}} \sum_{n=0}^{N-1} \left(x[n] - A^{k+1}\sin(2\pi f_0^k n + \phi)\right) \cdot A^{k+1}\cos(2\pi f_0^k n + \phi)$;
  $\frac{\partial \mathcal{J}}{\partial f_0} = -\frac{1}{N\mathcal{J}} \sum_{n=0}^{N-1} \left(x[n] - A^{k+1}\sin(2\pi f_0 n + \phi^{k+1})\right) \cdot A^{k+1} \cdot n\cos(2\pi f_0 n + \phi^{k+1})$;
  if $\mathcal{J}(A^{k+1}, \phi^{k+1}, f_0^{k+1}) \le \epsilon$ **then**
    break;
  else
    if $\mathcal{J}(A^{k+1}, \phi^{k+1}, f_0^{k+1}) > \mathcal{J}(A^k, \phi^k, f_0^k)$ **then**
      $A^{k+1} \leftarrow$ Eq.(32) where $\hat{f}_0 = f_0^k$;
      $\phi^{k+1} \leftarrow$ Eq.(34) where $\hat{f}_0 = f_0^k$;
      $f_0^{k+1} \leftarrow$ Eq.(29) where $A = A^k$;
    end
  end
end
return $A = A^k, \phi = \phi^k, f_0 = f_0^k$

4.3.2 Experiment

In order to apply the proposed algorithm, the initial estimate of the frequency, f_0^0, was calculated as 34999860.000560 Hz by MLE. This value was then substituted into equations (32) and (34), resulting in an initial estimate of 0.993606 for the magnitude and an initial estimate of 0.217254 for the phase. By employing our alternating optimisation algorithm, we were able to ascertain the precise value of the frequency, which is presented in Fig.9. Furthermore, the parameters within the iterative algorithm were configured in accordance with the values outlined in Table 2.



(a) Amplitude **(b) Phase**

(c) Frequency **(d) RMSE**

Figure 9 Use Algorithm 1 to combine magnitude and phase to estimate the exact frequency.

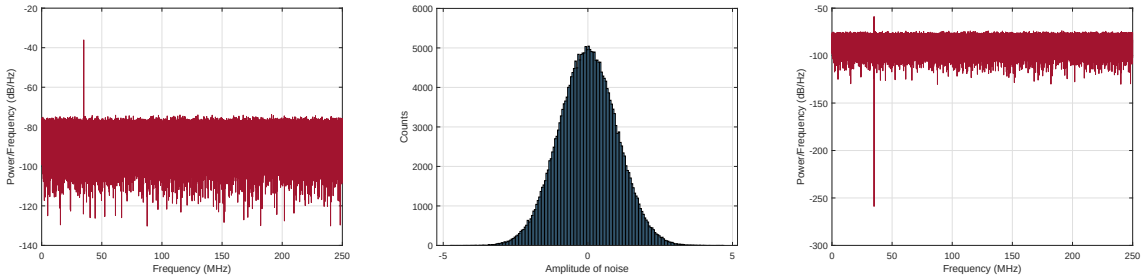
Table 2 The parameters used by the iterative algorithm.

Parameters	K	ϵ	η_A	η_ϕ	η_{f_0}
Value	25	1E+0	0.8	0.2	1.8

As illustrated in Fig.9, the amplitude demonstrates a declining trend, whereas the phase exhibits an ascending trend as the iteration progresses. However, the changes in amplitude and phase have a minimal impact on the estimation of frequency, given that the frequency estimation results require only a few iterations to be rapidly rectified to a stabilizing state. Finally, the RMSE curves generated by Algorithm 1 were also evaluated when the parameter $r_{zz}[1]$ in Eq.(29) was set to different values. It can be observed that as the parameter $r_{zz}[1]$ approaches zero, the estimation performance of the algorithm improves, as evidenced by the enhanced rate of decrease of the RMSE curve.

Based on our experience with the second question, it seems highly probable that the noise of a signal is a constant that obeys a Gaussian distribution, but whose power spectral density profile is not constant. Consequently, the value of the parameter $r_{zz}[1]$ may only tend to zero, but will not be equal to zero.

Similarly, we give the amplitude distribution of the noise signal and its power spectral density profile after applying Eq.(3), as shown in Fig.10.



(a) PSD of received signal **(b) Distribution of noise** **(c) PSD of noise signal**

Figure 10 The power spectrum of the received signal, as well as the amplitude distribution of our suspected noise signal and its power spectral density profile.

This is evident due to the fact that the useful signal has been extracted from the received signal, which has resulted in a significant reduction in noise power within the received signal, thereby corroborating the aforementioned analysis.

Table 3 Estimation results of the four methods for unknown magnitude, phase, and frequency.

Methods	A	ϕ (rad)	$\hat{f}_0$ (Hz)
MLE	0.993606	0.217254	34999860.000560
Robust MLE	0.500759	-0.002764	35000000.000000
MUSIC	-	-	348229645.9292
Ours	**0.965322**	**0.218798**	**32099623.112227**

Ultimately, a comparison was conducted between the estimation outcomes of Algorithm 1 and the remaining methodologies, as illustrated in Table 3.

It is evident that our estimation results are more closely aligned with those derived from MLE approach. This is attributable to the fact that our algorithmic framework is based on the MLE implementation, with subsequent enhancements in optimisation.

4.4 Model of Question Four

4.4.1 Model

The implementation of intermittent sampling not only circumvents the potential for signal interference but also results in a reduction in energy consumption within the system. Nevertheless, intermittent sampling is deleterious to the estimation of a signal frequency, whereby the exact frequency can only be approximated with a limited amount of sampled data. The objective of addressing this type of undersampled signal $\mathbf{y}$ is to proactively attempt to reconstruct its original signal $\mathbf{x}$, and subsequently utilise the accumulated insights from the second and third questions to ascertain the frequency of the original signal $\mathbf{x}$.

Compressed sensing techniques represent a well-established methodology for addressing such issues[3]. In the context of our signal model, the undersampled signal $\mathbf{y}$ can be conceptualised as the output of a sampling process applied to the original signal $\mathbf{x}$, with the measurement matrix $\mathbf{\Psi}$ employed as the sampling matrix. Thus we have:

$$\mathbf{y} = \mathbf{\Psi x} \tag{36}$$

where $\mathbf{y}$ represents an $M \times 1$ vector, the variable $\mathbf{x}$ is an $N \times 1$ vector, and $M \ll N$. If $\mathbf{x}$ can be rendered sparse in a given transformation domain by passing through the matrix $\mathbf{\Phi}$, then the aforementioned equation can be rewritten as follows:

$$\mathbf{y} = \mathbf{\Psi x} = \mathbf{\Psi \Phi w} = \mathbf{Hw} \tag{37}$$

where $\mathbf{w}$ is a highly sparse vector. Indeed, we are currently in possession of $\mathbf{y}$ and $\mathbf{H}$, and our objective is to recover $\mathbf{x}$, that is to say, to identify $\mathbf{w}$ within the sparse domain. This therefore necessitates the resolution of the optimisation problem described below:

$$\begin{cases} \min\limits_{\mathbf{w}} & \|\mathbf{y} - \mathbf{Hw}\|_2^2 \\ \text{s.t.} & \|\mathbf{x}\|_0 \end{cases} \tag{38}$$

The issue can be addressed through the utilisation of established CS signal reconstruction-based methodologies[4, 5, 6]. As the signals in question are obtained by intermittent sampling, and this intermittency is periodic, it is not feasible to reconstruct the original complete signal directly from these undersampled data.

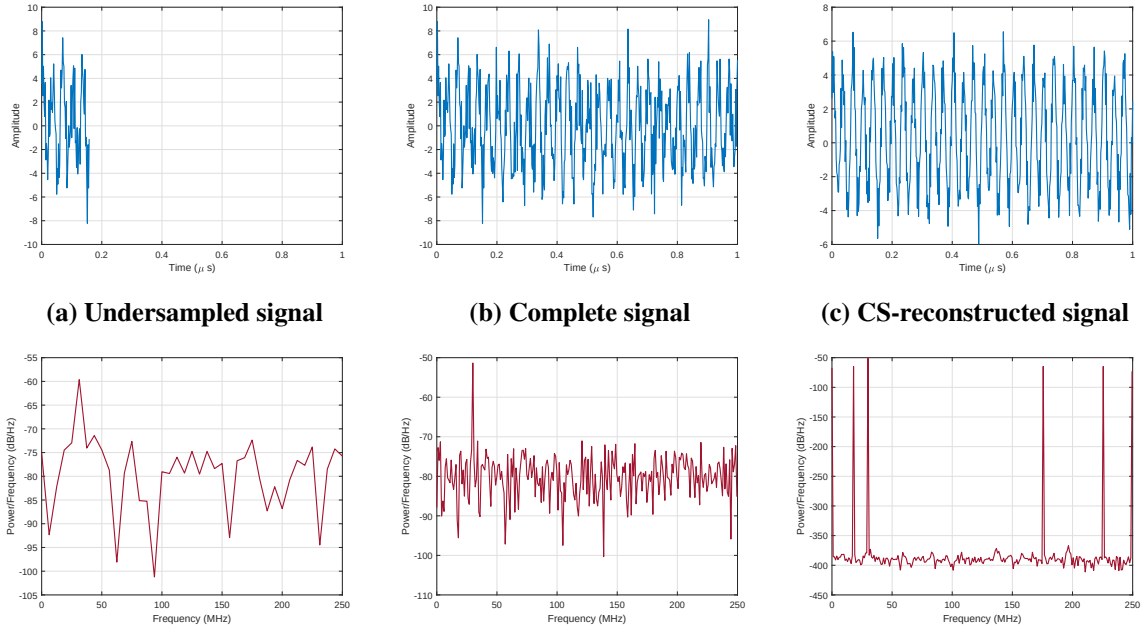
Moreover, a limiting relationship exists between the length M of the undersampled signal and the length N of the recovered signal. Consequently, the maximum possible number of recovery points N can be estimated:

$$M \geq C \cdot k \cdot \log\left(\frac{N}{k}\right) \tag{39}$$

where C is a constant and k is the sparsity. Empirical evidence suggests that typical sparse signals ($k = 10$) can typically be recovered to $N \approx 200 \sim 500$ for $M = 80$. In our process, for an undersampled signal $\mathbf{y}$ in the time interval $[t_0, t_1]$, we obtain an enhanced version of $\mathbf{y}^*$ with the help of the mathematical model of Eq.(38) and by applying the signal reconstruction algorithm.

The signal $\mathbf{y}^*$ is substituted into equation (18) in order to obtain MLE result of the signal frequency f_0. Furthermore, the signal $\mathbf{y}$ in each time interval can be reconstructed as a series of independent $\mathbf{y}^*$, which will all be estimated by MLE in order to obtain the frequency f_0. Finally, the estimation results are averaged in each time interval in order to obtain the final estimation results.

4.4.2 Experiment



(a) Undersampled signal **(b) Complete signal** **(c) CS-reconstructed signal**

(d) PSD of Undersampled signal **(e) PSD of complete signal** **(f) PSD of CS-reconstructed signal**

Figure 11 The time-domain waveforms and power spectra of the undersampled signal, the complete signal, and the CS-reconstructed signal are presented below. These results are based on the data provided in the initial question.

The CS approach is initially employed on the data set pertaining to the first question, which is to ascertain the viability of the algorithm. Figure 11 illustrates the time-domain waveforms

and power spectral density profiles of the undersampled signal, the original signal, and the CS-reconstructed signal, respectively. It should be noted that the results presented in Fig.11 have been obtained using a time slice. Furthermore, the duration of the reconstructed signal is 1 μs. The algorithm employed for signal reconstruction is Orthogonal Matching Pursuit (OMP), with sparsity parameter k set to 10.

It is evident that the true frequency of the signal is $f_0 = 30$ MHz. The reconstructed signal in the initial time interval yields a comparable result, although it exhibits peaks at several remaining frequency positions that are not anticipated.

Table 4 presents the estimated frequencies of the undersampled signals, as well as the reconstructed signals, in each time interval, calculated by MLE. The means of these frequencies are then compared to assess the efficacy of the CS technique.

Table 4 Frequency estimation results (Hz) in each time interval are estimated either directly using the undersampled signal or using the CS reconstructed signal.

Index of time interval	Undersampled Signal	CS-reconstructed Signal
1	31250000.000000	29940119.760479
2	31250000.000000	29940119.760479
3	31250000.000000	29940119.760479
4	31250000.000000	30938123.752495
5	31250000.000000	29940119.760479
6	31250000.000000	29940119.760479
7	31250000.000000	29940119.760479
8	31250000.000000	29940119.760479
9	31250000.000000	29940119.760479
10	31250000.000000	29940119.760479
Mean	31250000.000000	30039920.159681

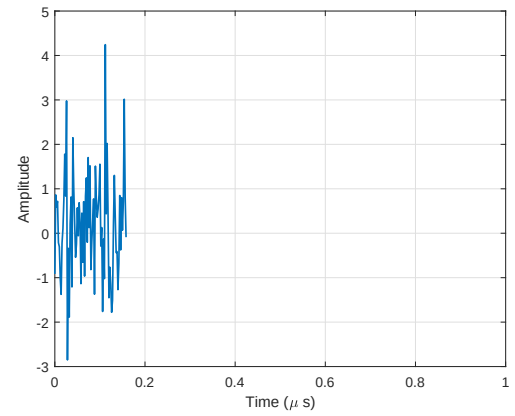
It can be seen that the under-sampled signals in each time interval are reconstructed by CS according to Eq.(38), and then the frequency of the signals is estimated according to Eq.(18), and finally these estimation results are averaged, so that the estimation results so obtained are closer to the real value.

Furthermore, it is essential to ascertain the precision of the ultimate estimation outcomes derived from disparate signal reconstruction algorithms. In Table 5, we present the estimation

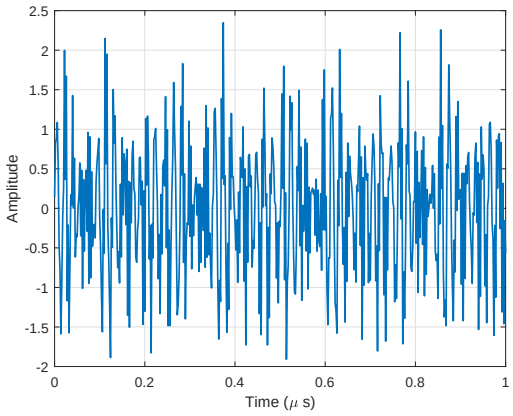
results of the three CS signal reconstruction algorithms.

Table 5 Frequency estimation results (Hz) due to three CS signal reconstruction algorithms.

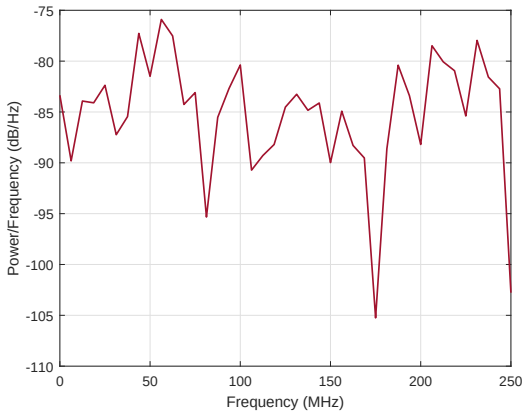
Methods	$\hat{f}_0$
Orthogonal Matching Pursuit	30039920.159681
Compressive Sampling Matching Pursuit	27145708.582834
Subspace Pursuit	32634730.538922



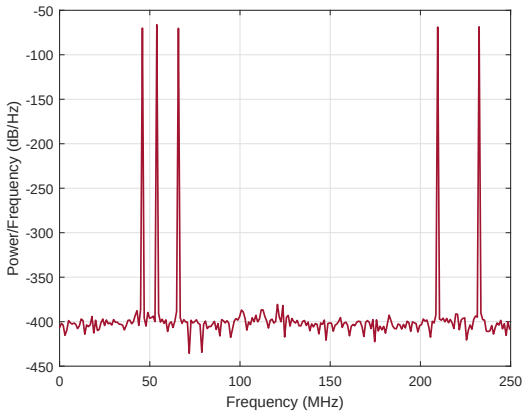
(a) Undersampled signal



(b) CS-reconstructed signal



(c) PSD of Undersampled signal



(d) PSD of CS-reconstructed signal

Figure 12 The time-domain waveforms and power spectra of the undersampled signal and the CS-reconstructed signal are presented below. These results are based on the data provided in this question.

The results of the estimation demonstrate that the OMP algorithm is capable of achieving a higher degree of accuracy in the estimation of the signal frequency f_0. The aforementioned experimental model is applied to the data set corresponding to the fourth question. Given that the signal is presented in the form of undersampling in this question, the processing associated with the complete signal is omitted here.

Figure 12 illustrates the outcomes of the modelling process applied to the experimental data set depicted in Figure 11. The sole difference between the two is that the data from the fourth question were utilised in the modelling process.

Table 6 Frequency estimation results (Hz) in each time interval are estimated either directly using the undersampled signal or using the CS reconstructed signal. These results are based on the data provided in this question.

Index of time interval	Undersampled Signal	CS-reconstructed Signal
1	56250000.000000	53892215.568862
2	56250000.000000	54890219.560878
3	56250000.000000	53892215.568862
4	56250000.000000	54890219.560878
5	56250000.000000	55888223.552894
6	56250000.000000	54890219.560878
7	56250000.000000	55888223.552894
8	56250000.000000	54890219.560878
9	56250000.000000	54890219.560878
10	56250000.000000	54890219.560878
Mean	56250000.000000	54890219.560878

The frequency estimation achieved by reconstructing the signal based on compressive sensing was found to remain valid. Additionally, the results presented in Table 6 were computed in a manner consistent with those shown in Table 4. Furthermore, the efficacy of the three reconstruction algorithms was verified, as illustrated in Table 7.

Table 7 Frequency estimation results (Hz) due to three CS signal reconstruction algorithms. These results are based on the data provided in this question.

Methods	$\hat{f}_0$
Orthogonal Matching Pursuit	54890219.560878
Compressive Sampling Matching Pursuit	55688622.754491
Subspace Pursuit	50099800.399202

5. Strengths and Weakness

5.1 Strengths

■ In this paper, the noises are classified as either Gaussian or non-Gaussian, and the frequencies are estimated using a variety of models.

■ In this paper, we utilise the particle swarm intelligent optimisation algorithm with a compressed perception technique to accurately estimate the frequency, thereby addressing the issue of undersampling.

5.2 Weakness

■ It should be noted that the estimation algorithm presented in this paper provides only an approximate estimation of the frequency. Furthermore, the accuracy of the algorithm at the Hz level still requires improvement.

■ The model presented in this paper is contingent upon the quantity of data employed, and there is a possibility that both insufficient and excessive data may result in estimation failure.

References

[1] S.M. Kay, Fundamentals of Statistical Signal Processing: volume 1, estimation theory, Upper Saddle River: Prentice Hall PTR, 136-211, 2017.

[2] Z.Y. Bao, J.Z. Yu, S. Yang, Intelligent Optimisation Algorithms and their MATLAB Examples (2nd Edition), BeiJing: Publishing House of Electronics Industry, 154-160, 2018.

[3] D. L. Donoho, ”Compressed sensing,” in IEEE Transactions on Information Theory, vol. 52, no. 4, pp. 1289-1306, 2006. doi: 10.1109/TIT.2006.871582.

[4] Joel A. Tropp and Anna C. Gilbert. Signal Recovery From Random Measurements Via Orthogonal Matching Pursuit[J]. IEEETransactions on Information Theory, VOL. 53, NO. 12, 2007. doi: 10.1109/TIT.2007.909108.

[5] D.Needell, J.A. Tropp. CoSaMP: Iterative signal recoveryfrom incomplete and inaccurate samples. Communications of theACM, 53(12) 93-100, 2010. doi: 10.1145/1859204.1859229.

[6] W. Dai and O. Milenkovic, ”Subspace Pursuit for Compressive Sensing Signal Reconstruction,” in IEEE Transactions on Information Theory, vol. 55, no. 5, pp. 2230-2249, 2009. doi: 10.1109/TIT.2009.2016006.

Appendix

■ Symbol Description

Symbol	Description
A	Amplitude
ϕ	Phase
f_0	Frequency
$x[n]$	Discrete Sampled Signal Sequence
$z[n]$	Discrete Noise Signal Sequence

■ Assumptions

In this paper, we must assume that the amplitude of the noise does not necessarily follow a Gaussian distribution, but must have a mean of zero. Furthermore, we must assume that the signal is stationary and that the estimated parameters are all constant and not time-varying.

- MATLAB CODE for Q1

Listing 1: **Q1.m**

```
clc
clear
close all;

load data_Q1.mat;

t = data_Q1(:,1);
x = data_Q1(:,2);

Ts = 2e-9;
Fs = 1 / Ts;
A = 4;
f_signal = 30e6;
phi = deg2rad(45);

x_signal = A*sin(2*pi*f_signal*t + phi);
z = x - x_signal;

%% Fig 1
figure;
plot(t*1000,x);
grid on;
xlabel('Time (ms)');
ylabel('Amplitude');

figure;
plot(t*1000,z);
grid on;
xlabel('Time (ms)');
ylabel('Amplitude');

figure;
histogram(z);
grid on;
xlabel('Amplitude of noise');
ylabel('Counts');

mu_z = mean(z);
sigma_z = std(z);
sigma_2_z = sigma_z^2;

fprintf('mu_z = %f\t sigma_2_z = %f\n',mu_z,sigma_2_z);
```

```
%% Fig 2

[Z,f] = myFFT(z,Fs);

figure;
plot(f/1e6,Z,'Color',[0.6350 0.0780 0.1840]);
grid on;
xlabel('Frequency (MHz)');
ylabel('Amplitude');

[P,f] = myPeriodogram(z,Fs);

figure;
plot(f/1e6,pow2db(P),'Color',[0.6350 0.0780 0.1840]);
grid on;
xlabel('Frequency (MHz)');
ylabel('Power/Frequency (dB/Hz)');

figure;
spectrogram(z,[],[],[],Fs,'yaxis')
```

- MATLAB CODE for Q2

Listing 2: **Q2.m**

```
clc
clear
close all;

warning off;
load data_Q2.mat;

t = data_Q2(:,1);
x = data_Q2(:,2);
N = 10;

A = 2;
phi = deg2rad(0);

f0 = linspace(0.05,0.45,200);

figure;
hold on;
for i=1:4
```

```
    CRBL_t = zeros(length(f0),1);
    for j=1:length(f0)
        n = 0:N-1;
        temp = 2*pi*n.*cos(2*pi*f0(j)*n + phi);
        temp = temp.^2;
        temp = sum(temp);
        CRBL_t(j) = i/(A^2*temp);
    end
    plot(f0,CRBL_t,'LineWidth',1.2);
end
grid on;
xlabel('Frequency (Hz)');
ylabel('CRLB');
legend('\sigma_z^2 = 1.0','\sigma_z^2 = 2.0','\sigma_z^2 = 3.0','\sigma_z^2 =
    4.0','Location','best');
set(gca,'yScale','log');

%% MLE

Ts = 2e-9;
Fs = 1 / Ts;

[P,f] = myPeriodogram(x,Fs);

figure;
plot(f/1e6,pow2db(P),'Color',[0.6350 0.0780 0.1840]);
grid on;
xlabel('Frequency (MHz)');
ylabel('Power/Frequency (dB/Hz)');

[~,index_max] = max(P);
f0_mle = f(index_max);
fprintf('MLE for f0 is %f (Hz)\n',f0_mle);

x_signal = A*sin(2*pi*f0_mle*t + phi);
z = x - x_signal;

figure;
histogram(z);
grid on;
xlabel('Amplitude of noise');
ylabel('Counts');

[P,f] = myPeriodogram(z,Fs);
```

```
figure;
plot(f/1e6,pow2db(P),'Color',[0.6350 0.0780 0.1840]);
grid on;
xlabel('Frequency (MHz)');
ylabel('Power/Frequency (dB/Hz)');

% [c,lags] = xcorr(z);
% figure;
% stem(lags,c);

%% MOM

r_zz_1 = linspace(-0.1,0.1,20);
f0_mom = zeros(length(r_zz_1),1);

for k=1:length(f0_mom)
    temp = 0;
    for j=1:length(x)-1
        temp = temp + x(j)*x(j+1);
    end
    temp = 2*temp - 2*(length(x) - 1) * r_zz_1(k);
    temp = temp / (A^2*(length(x) - 1));

    f0_mom_t = acos(temp) / (2*pi);
    f0_mom(k) = f0_mom_t * Fs;
end

fprintf('MOM for f0 is %f (Hz)\n',f0_mom(10));

figure;
plot(r_zz_1,f0_mom/1e6,'-o','LineWidth',1.2,'Color',[0.6350 0.0780 0.1840]);
grid on;
xlabel('Value of r_{zz}[1]');
ylabel('Frequency (MHz)');

r_zz_1 = [-0.02 -0.01 0 0.02 0.05];
N = [1e3 1e4 1e5 1e5 length(x)-1] + 1;
f0_mom_RMSE = zeros(length(N),1);

mark_t = {'-o','-d','-s','-x','-*'};

figure;
hold on;
for k=1:length(r_zz_1)
    for i=1:length(N)
```

```
        x_t = x(1:N(i));
        temp = 0;
        for j=1:length(x_t)-1
            temp = temp + x_t(j)*x_t(j+1);
        end
        temp = 2*temp - 2*(length(x_t) - 1) * r_zz_1(k);
        temp = temp / (A^2*(length(x_t) - 1));

        f0_mom_t = acos(temp) / (2*pi);
        f0_mom_t = f0_mom_t * Fs;

        % RMSE
        x_signal = A*sin(2*pi*f0_mom_t*t(1:N(i)) + phi);
        f0_mom_RMSE(i) = sqrt(mean((x_signal - x_t).^2));
    end
    plot(N,f0_mom_RMSE,mark_t{k},'LineWidth',1.2);
end
grid on;
xlabel('Number of x[n]');
ylabel('RMSE');
set(gca,'XScale','log');
set(gca,'YScale','log');
legend('r_{zz}[1] = -0.02','r_{zz}[1] = -0.01','r_{zz}[1] = 0','r_{zz}[1] =
    0.02','r_{zz}[1] = 0.05','Location','best');
box on;

% x_signal = A*sin(2*pi*f0_mom*t + phi);
% z = x - x_signal;
%
%
%
% [P,f] = myPeriodogram(z,Fs);
%
% figure;
% plot(f/1e6,pow2db(P),'Color',[0.6350 0.0780 0.1840]);
% grid on;
% xlabel('Frequency (MHz)');
% ylabel('Power/Frequency (dB/Hz)');
```

Listing 3: **Q2_PSO.m**

```
clc
clear
close all;

```

```
load data_Q2.mat;

t = data_Q2(:,1);
x_data = data_Q2(:,2);

D_pso = 1;
N_pso = 8;      % Number of p
T_pso = 8;      % Iteration
c1 = 1.5;
c2 = 1.5;
Wmax = 0.8;
Wmin = 0.6;
Xmax = 1*ones(1,D_pso);
Xmin = (-1)*ones(1,D_pso);
Vmax = 0.0002*ones(1,D_pso);
Vmin = -0.0002*ones(1,D_pso);

x = rand(N_pso,D_pso).*(Xmax - Xmin) + Xmin;
v = rand(N_pso,D_pso).*(Vmax - Vmin) + Vmin;

p = x;
pbest = ones(N_pso,1);
for i=1:N_pso
    pbest(i) = fitFun(x(i,:),t,x_data);
end

g = ones(1,D_pso);
gbest = inf;
for i=1:N_pso
    if(pbest(i) < gbest)
        g = p(i,:);
        gbest = pbest(i);
    end
end
gb = ones(1,T_pso);


for i=1:T_pso
    for j=1:N_pso

        if(fitFun(x(j,:),t,x_data) < pbest(j))
            p(j,:) = x(j,:);
            pbest(j) = fitFun(x(j,:),t,x_data);
        end
```

```
        if(pbest(j) < gbest)
            g = p(j,:);
            gbest = pbest(j);
        end

        w = Wmax - (Wmax - Wmin)*i/T_pso;

        v(j,:) = w*v(j,:) + c1*rand*(p(j,:) - x(j,:)) + c2*rand*(g-x(j,:));
        x(j,:) = x(j,:) + v(j,:);

        for ii=1:D_pso
            if(v(j,ii) > Vmax(ii)) || (v(j,ii) < Vmin(ii))
                v(j,ii) = rand*(Vmax(ii) - Vmin(ii)) + Vmin(ii);
            end
            if(x(j,ii) > Xmax(ii)) || (x(j,ii) < Xmin(ii))
                x(j,ii) = rand*(Xmax(ii) - Xmin(ii)) + Xmin(ii);
            end
        end
    end

    gb(i) = gbest;
end
optX_pso = g;
optFit_pso = gb;

figure;
plot(1:length(optFit_pso),optFit_pso,'LineWidth',1.2);
grid on;
xlabel('Iteration times');
ylabel('RMSE');
set(gca,'YScale','log');

fprintf('frequency by PSO is: f = %f\n',optX_pso*(1e6) + 40e6);

function output = fitFun(f,t,x_data)
    f = f*(1e6) + 40e6;
    A = 2;
    phi = deg2rad(0);
    x_signal = A*sin(2*pi*f*t + phi);
    output = sqrt(mean((x_signal - x_data).^2));
end
```

- MATLAB CODE for Q3

Listing 4: **Q3.m**

```
clc
clear
close all;

load data_Q3.mat;

t = data_Q3(:,1);
x = data_Q3(:,2);
N = length(x);

Ts = 2e-9;
Fs = 1 / Ts;

[P,f] = myPeriodogram(x,Fs);
[~,index_max] = max(P);

f0_0 = f(index_max);
A0 = abs(sum(x.*exp(-1i*2*pi*f0_0/Fs*(0:N-1)')))*2 / N;
phi_0 = atan2(-sum(x.*sin(2*pi*f0_0/Fs*(0:N-1)')),sum(x.*cos(2*pi*f0_0/Fs*(0:N-1)')))
    + pi/2;

K = 25;
eps = 1;

A = zeros(K,1);
phi = zeros(K,1);
f0 = zeros(K,1);

A(1) = A0;
phi(1) = phi_0;
f0(1) = f0_0;

eta_A = 0.8;
eta_phi = 0.2;
eta_f0 = 1.8;

J = @(A,phi,f0)sqrt(mean((x - A*sin(2*pi*f0*t + phi)).^2));
fprintf('k = %d, J_now = %f\n',0,J(A(1),phi(1),f0(1)));
for k=1:K-1
    dA = -mean((x - A(k)*sin(2*pi*f0(k)*t + phi(k))).^2 .* sin(2*pi*f0(k)*t + phi(k)))
        / J(A(k),phi(k),f0(k));
    A(k+1) = A(k) - eta_A*dA;
    dphi = -mean((x - A(k+1)*sin(2*pi*f0(k)*t + phi(k))) .* A(k+1) .* cos(2*pi*f0(k)*t
```

```
        + phi(k))) / J(A(k+1),phi(k),f0(k));
    phi(k+1) = phi(k) - eta_phi*dphi;
    df = -mean((x - A(k+1)*sin(2*pi*f0(k)*t + phi(k+1))) .* A(k+1) .* (0:N-1)' ...
        .* cos(2*pi*f0(k)*t + phi(k+1))) / J(A(k+1),phi(k+1),f0(k));
    f0(k+1) = f0(k) - eta_f0*df;

    J_now = J(A(k+1),phi(k+1),f0(k+1));
    fprintf('k = %d, J_now = %f\n',k,J_now);
    if(J_now <= eps)
        break;
    else
        if(J_now >= J(A(k),phi(k),f0(k)))
            A(k+1) = abs(sum(x.*exp(-1i*2*pi*f0(k)/Fs*(0:N-1)')))*2 / N;
            phi(k+1) =
                atan2(-sum(x.*sin(2*pi*f0(k)/Fs*(0:N-1)')),sum(x.*cos(2*pi*f0(k)/Fs*(0:N-1)')))
                + pi/2;

            r_zz_1 = 0.01;

            temp = 0;
            for j=1:length(x)-1
                temp = temp + x(j)*x(j+1);
            end
            temp = 2*temp - 2*(length(x) - 1) * r_zz_1;
            temp = temp / (A(k)^2*(length(x) - 1));

            f0_mom_t = acos(temp) / (2*pi);
            f0(k+1) = f0_mom_t * Fs;
        end
    end
end

fprintf('A = %f\t phi = %f\t f_0 = %f\n',A(1),phi(1),f0(1));

figure;
plot(1:length(A)-1,A(2:end),'-o','LineWidth',1.2);
grid on;
xlabel('Iteration times');
ylabel('A');

figure;
plot(1:length(A)-1,phi(2:end),'-d','LineWidth',1.2,'Color',[0.8500 0.3250 0.0980]);
grid on;
xlabel('Iteration times');
ylabel('\phi');
```

```
figure;
plot(1:length(A)-1,f0(2:end)/(1e6),'-s','LineWidth',1.2,'Color',[0.6350 0.0780
    0.1840]);
grid on;
xlabel('Iteration times');
ylabel('f_0 (MHz)');
set(gca,'YScale','log');
ylim([min(f0(2:end)/(1e6))-1e-5 max(f0(2:end)/(1e6)) + 1e-5]);


%%
clear

load data_Q3.mat;

t = data_Q3(:,1);
x = data_Q3(:,2);
N = length(x);

Ts = 2e-9;
Fs = 1 / Ts;

[P,f] = myPeriodogram(x,Fs);
[~,index_max] = max(P);

f0_0 = f(index_max);
A0 = abs(sum(x.*exp(-1i*2*pi*f0_0/Fs*(0:N-1)')))*2 / N;
phi_0 = atan2(-sum(x.*sin(2*pi*f0_0/Fs*(0:N-1)')),sum(x.*cos(2*pi*f0_0/Fs*(0:N-1)')))
    + pi/2;

K = 25;
eps = 1;

A = zeros(K,1);
phi = zeros(K,1);
f0 = zeros(K,1);

A(1) = A0;
phi(1) = phi_0;
f0(1) = f0_0;

eta_A = 0.8;
eta_phi = 0.2;
eta_f0 = 1.8;
```

```
J = @(A,phi,f0)sqrt(mean((x - A*sin(2*pi*f0*t + phi)).^2));
fprintf('k = %d, J_now = %f\n',0,J(A(1),phi(1),f0(1)));

r_zz_1_all = [0.05 0.01 0.002 0.001];
RMSE_all = zeros(K-1,length(r_zz_1_all));

for p=1:length(r_zz_1_all)
    for k=1:K-1
        dA = -mean((x - A(k)*sin(2*pi*f0(k)*t + phi(k))).^2 .* sin(2*pi*f0(k)*t +
            phi(k))) / J(A(k),phi(k),f0(k));
        A(k+1) = A(k) - eta_A*dA;
        dphi = -mean((x - A(k+1)*sin(2*pi*f0(k)*t + phi(k))) .* A(k+1) .*
            cos(2*pi*f0(k)*t + phi(k))) / J(A(k+1),phi(k),f0(k));
        phi(k+1) = phi(k) - eta_phi*dphi;
        df = -mean((x - A(k+1)*sin(2*pi*f0(k)*t + phi(k+1))) .* A(k+1) .* (0:N-1)' ...
            .* cos(2*pi*f0(k)*t + phi(k+1))) / J(A(k+1),phi(k+1),f0(k));
        f0(k+1) = f0(k) - eta_f0*df;

        J_now = J(A(k+1),phi(k+1),f0(k+1));
        fprintf('k = %d, J_now = %f\n',k,J_now);
        if(J_now <= eps)
            break;
        else
            if(J_now >= J(A(k),phi(k),f0(k)))
                A(k+1) = abs(sum(x.*exp(-1i*2*pi*f0(k)/Fs*(0:N-1)')))*2 / N;
                phi(k+1) =
                    atan2(-sum(x.*sin(2*pi*f0(k)/Fs*(0:N-1)')),sum(x.*cos(2*pi*f0(k)/Fs*(0:N-1)')))
                    + pi/2;

                r_zz_1 = r_zz_1_all(p);

                temp = 0;
                for j=1:length(x)-1
                    temp = temp + x(j)*x(j+1);
                end
                temp = 2*temp - 2*(length(x) - 1) * r_zz_1;
                temp = temp / (A(k)^2*(length(x) - 1));

                f0_mom_t = acos(temp) / (2*pi);
                f0(k+1) = f0_mom_t * Fs;
            end
        end
        RMSE_all(k,p) = J(A(k+1),phi(k+1),f0(k+1));
    end
```

```
end

figure;
plot(1:length(A)-1,RMSE_all(:,1),'-.','LineWidth',1.2,'Color',[0.4660 0.6740 0.1880]);
hold on;
plot(1:length(A)-1,RMSE_all(:,2),'-.','LineWidth',1.2,'Color',[0.9290 0.6940 0.1250]);
plot(1:length(A)-1,RMSE_all(:,3),'-.','LineWidth',1.2,'Color',[0.4940 0.1840 0.5560]);
plot(1:length(A)-1,RMSE_all(:,4),'-.','LineWidth',1.2,'Color','k');
grid on;
xlabel('Iteration times');
ylabel('RMSE');
set(gca,'YScale','log');
legend('r_{zz}[1] = 0.05','r_{zz}[1] = 0.01','r_{zz}[1] = 0.002','r_{zz}[1] =
    0.001','Location','best');

%%

[P,f] = myPeriodogram(x,Fs);

figure;
plot(f/1e6,pow2db(P),'Color',[0.6350 0.0780 0.1840]);
grid on;
xlabel('Frequency (MHz)');
ylabel('Power/Frequency (dB/Hz)');

x_signal = A(1)*sin(2*pi*f0(1)*t + phi(1));
z = x - x_signal;

figure;
histogram(z);
grid on;
xlabel('Amplitude of noise');
ylabel('Counts');

[P,f] = myPeriodogram(z,Fs);

figure;
plot(f/1e6,pow2db(P),'Color',[0.6350 0.0780 0.1840]);
grid on;
xlabel('Frequency (MHz)');
ylabel('Power/Frequency (dB/Hz)');
```

- MATLAB CODE for Q4

Listing 5: **Q4.m**

```
clc
clear
close all;

warning off;
load data_Q4.mat;
load data_Q3.mat;

t_ref = data_Q3(:,1);

Ts = 2e-9;
Fs = 1 / Ts;

% time_ranges = [0 0.000000158;
% 0.00005 0.000050158;
% 0.0001 0.000100158;
% 0.00015 0.000150158;
% 0.0002 0.000200158;
% 0.00025 0.000250158;
% 0.0003 0.000300158;
% 0.00035 0.000350158;
% 0.0004 0.000400158;
% 0.00045 0.000450158];

T_all = [0 0.000000158;
    0.00005 0.000050158;
    0.0001 0.000100158;
    0.00015 0.000150158;
    0.0002 0.000200158;
    0.00025 0.000250158;
    0.0003 0.000300158;
    0.00035 0.000350158;
    0.0004 0.000400158;
    0.00045 0.000450158];

f0 = zeros(size(T_all,1),2);
fprintf('Iteration\t undersampled signal\t CS-reconstructed signal\n');
fprintf('-------------------------------------------------------------\n');
for it=1:size(T_all,1)

    time_ranges = T_all(it,:);
    t = data_Q4(:,1);
    y = data_Q4(:,2);
```

```
[~, idx_1] = min(abs(t - time_ranges(1)));
[~, idx_2] = min(abs(t - time_ranges(2)));
t_sample = t(idx_1:idx_2);
x_sample = y(idx_1:idx_2);

[~, idx_1] = min(abs(t_ref - time_ranges(1)));
[~, idx_2] = min(abs(t_ref - (time_ranges(1)+0.000001)));
t_ref_t = t_ref(idx_1:idx_2);

N = length(t_ref_t);
M = length(x_sample);
Phi = zeros(M, N);
for i = 1:M
    [~, idx] = min(abs(t_ref_t - t_sample(i)));
    Phi(i, idx) = 1;
end

D = dftmtx(N);
D = (1/N) * D';

H = Phi * D;
s = 10;
theta = CS_OMP(x_sample,H,s);
% theta = CS_CoSaMP(x_sample,H,s);
% theta = CS_SP(x_sample,H,s);
y_CS = real(D*theta);

% figure;
% plot(t_sample*1e6,x_sample,'LineWidth',1.2);
% xlabel('Time (\mu s)');
% ylabel('Amplitude');
% grid on;
% xlim([min(t_ref_t*1e6) max(t_ref_t*1e6)]);
%
% figure;
% plot(t_ref_t*1e6,y_CS,'LineWidth',1.2);
% xlabel('Time (\mu s)');
% ylabel('Amplitude');
% grid on;
% xlim([min(t_ref_t*1e6) max(t_ref_t*1e6)]);
%
% [P,f] = myPeriodogram(x_sample,Fs);
%
% figure;
% plot(f/1e6,pow2db(P),'Color',[0.6350 0.0780 0.1840],'LineWidth',1.2);
```

```
    % grid on;
    % xlabel('Frequency (MHz)');
    % ylabel('Power/Frequency (dB/Hz)');
    %
    % [P,f] = myPeriodogram(y_CS,Fs);
    %
    % figure;
    % plot(f/1e6,pow2db(P),'Color',[0.6350 0.0780 0.1840],'LineWidth',1.2);
    % grid on;
    % xlabel('Frequency (MHz)');
    % ylabel('Power/Frequency (dB/Hz)');

    [P,f] = myPeriodogram(x_sample,Fs);
    [~,index_max] = max(P);
    f0_u = f(index_max);

    [P,f] = myPeriodogram(y_CS,Fs);
    [~,index_max] = max(P);
    f0_cs = f(index_max);
    f0(it,1) = f0_u;
    f0(it,2) = f0_cs;

    fprintf('%d\t %f\t %f\n',it,f0_u,f0_cs);

end

fprintf('Undersampled: Mean of f0 = %f (Hz)\n',mean(f0(:,1)));
fprintf('CS: Mean of f0 = %f (Hz)\n',mean(f0(:,2)));
```

Listing 6: **Q4_Q1.m**

```
clc
clear
close all;

warning off;
load data_Q1.mat;
Ts = 2e-9;
Fs = 1 / Ts;

% time_ranges = [0 0.000000158;
% 0.00005 0.000050158;
% 0.0001 0.000100158;
% 0.00015 0.000150158;
% 0.0002 0.000200158;
```

```
% 0.00025 0.000250158;
% 0.0003 0.000300158;
% 0.00035 0.000350158;
% 0.0004 0.000400158;
% 0.00045 0.000450158];

T_all = [0 0.000000158;
0.00005 0.000050158;
0.0001 0.000100158;
0.00015 0.000150158;
0.0002 0.000200158;
0.00025 0.000250158;
0.0003 0.000300158;
0.00035 0.000350158;
0.0004 0.000400158;
0.00045 0.000450158];

f0 = zeros(size(T_all,1),2);
fprintf('Iteration\t undersampled signal\t CS-reconstructed signal\n');
fprintf('-------------------------------------------------------------\n');
for it=1:size(T_all,1)

    time_ranges = T_all(it,:);
    t = data_Q1(:,1);
    y = data_Q1(:,2);
    [~, idx_1] = min(abs(t - time_ranges(1)));
    [~, idx_2] = min(abs(t - time_ranges(2)));
    t_sample = t(idx_1:idx_2);
    x_sample = y(idx_1:idx_2);

    [~, idx_1] = min(abs(t - time_ranges(1)));
    [~, idx_2] = min(abs(t - (time_ranges(1)+0.000001)));
    t = t(idx_1:idx_2);
    y = y(idx_1:idx_2);

    N = length(y);
    M = length(x_sample);

    Phi = zeros(M, N);

    for i = 1:M
        [~, idx] = min(abs(t - t_sample(i)));
        Phi(i, idx) = 1;
    end
```

```
D = dftmtx(length(y));
D = (1/length(y)) * D';

H = Phi * D;
s = 10;
theta = CS_OMP(x_sample,H,s);
% theta = CS_CoSaMP(x_sample,H,s);% -> 27 MHz
% theta = CS_SP(x_sample,H,s); % -> 32.6 MHz
y_CS = real(D*theta);



%
% figure;
% plot(t_sample*1e6,x_sample,'LineWidth',1.2);
% xlabel('Time (\mu s)');
% ylabel('Amplitude');
% grid on;
% xlim([min(t*1e6) max(t*1e6)]);
%
% figure;
% plot(t*1e6,y,'LineWidth',1.2);
% xlabel('Time (\mu s)');
% ylabel('Amplitude');
% grid on;
% xlim([min(t*1e6) max(t*1e6)]);
%
% figure;
% plot(t*1e6,y_CS,'LineWidth',1.2);
% xlabel('Time (\mu s)');
% ylabel('Amplitude');
% grid on;
% xlim([min(t*1e6) max(t*1e6)]);
%
% [P,f] = myPeriodogram(x_sample,Fs);
%
% figure;
% plot(f/1e6,pow2db(P),'Color',[0.6350 0.0780 0.1840],'LineWidth',1.2);
% grid on;
% xlabel('Frequency (MHz)');
% ylabel('Power/Frequency (dB/Hz)');
%
% [P,f] = myPeriodogram(y,Fs);
%
% figure;
% plot(f/1e6,pow2db(P),'Color',[0.6350 0.0780 0.1840],'LineWidth',1.2);
```

```
    % grid on;
    % xlabel('Frequency (MHz)');
    % ylabel('Power/Frequency (dB/Hz)');
    %
    % [P,f] = myPeriodogram(y_CS,Fs);
    %
    % figure;
    % plot(f/1e6,pow2db(P),'Color',[0.6350 0.0780 0.1840],'LineWidth',1.2);
    % grid on;
    % xlabel('Frequency (MHz)');
    % ylabel('Power/Frequency (dB/Hz)');

    [P,f] = myPeriodogram(x_sample,Fs);
    [~,index_max] = max(P);
    f0_u = f(index_max);

    [P,f] = myPeriodogram(y_CS,Fs);
    [~,index_max] = max(P);
    f0_cs = f(index_max);
    f0(it,1) = f0_u;
    f0(it,2) = f0_cs;

    fprintf('it = %d\t %f\t %f\n',it,f0_u,f0_cs);

end

fprintf('Undersampled: Mean of f0 = %f (Hz)\n',mean(f0(:,1)));
fprintf('CS: Mean of f0 = %f (Hz)\n',mean(f0(:,2)));
```

- MATLAB CODE for auxiliary function

Listing 7: **myFFT.m**

```
function [X,f] = myFFT(x,Fs)
    X = fft(x);
    L = length(x);
    f = Fs/L*(0:(L/2));
    P2 = abs(X/L);
    P1 = P2(1:floor(L/2+1));
    P1(2:end-1) = 2*P1(2:end-1);
    X = P1;
end
```

Listing 8: **myPeriodogram.m**

```
function [P,f] = myPeriodogram(x,fs)
    N = length(x);
    xdft = fft(x);
    xdft = xdft(1:N/2+1);
    P = (1/(fs*N)) * abs(xdft).^2;
    P(2:end-1) = 2*P(2:end-1);
    f = 0:fs/length(x):fs/2;
end
```

Listing 9: **CS_OMP.m**

```
function [ theta ] = CS_OMP( y,A,t )

    [y_rows,y_columns] = size(y);
    if y_rows<y_columns
        y = y';
    end
    [M,N] = size(A);
    theta = zeros(N,1);

    At = zeros(M,t);
    Pos_theta = zeros(1,t);
    r_n = y;
    for ii=1:t
        product = A'*r_n;
        [val,pos] = max(abs(product));
        At(:,ii) = A(:,pos);
        Pos_theta(ii) = pos;
        A(:,pos) = zeros(M,1);
        theta_ls = (At(:,1:ii)'*At(:,1:ii))^(-1)*At(:,1:ii)'*y;
        r_n = y - At(:,1:ii)*theta_ls;
    end
    theta(Pos_theta)=theta_ls;
end
```

Listing 10: **CS_CoSaMP.m**

```
function [ theta ] = CS_CoSaMP( y,A,K )

    [y_rows,y_columns] = size(y);
    if y_rows<y_columns
        y = y';
    end
    [M,N] = size(A);
    theta = zeros(N,1);
```

```
    Pos_theta = [];
    r_n = y;
    for kk=1:K
        product = A'*r_n;
        [val,pos]=sort(abs(product),'descend');
        Js = pos(1:2*K);
        %(2) Support Merger
        Is = union(Pos_theta,Js);
        %(3) Estimation
        if length(Is)<=M
            At = A(:,Is);
        else
            if kk == 1
                theta_ls = 0;
            end
            break;
        end
        theta_ls = (At'*At)^(-1)*At'*y;
        %(4) Pruning
        [val,pos]=sort(abs(theta_ls),'descend');
        %(5) Sample Update
        Pos_theta = Is(pos(1:K));
        theta_ls = theta_ls(pos(1:K));
        r_n = y - At(:,pos(1:K))*theta_ls;
        if norm(r_n)<1e-6
            break;
        end
    end
    theta(Pos_theta)=theta_ls;
end
```

Listing 11: **CS_SP.m**

```
function [ theta ] = CS_SP( y,A,K )
    [y_rows,y_columns] = size(y);
    if y_rows<y_columns
        y = y';
    end
    [M,N] = size(A);
    theta = zeros(N,1);
    Pos_theta = [];
    r_n = y;
    for kk=1:K
        product = A'*r_n;
        [val,pos]=sort(abs(product),'descend');
```

```
        Js = pos(1:K);
        Is = union(Pos_theta,Js);
        if length(Is)<=M
            At = A(:,Is);
        else
            break;
        end
        theta_ls = (At'*At)^(-1)*At'*y;
        [val,pos]=sort(abs(theta_ls),'descend');
        Pos_theta = Is(pos(1:K));
        theta_ls = theta_ls(pos(1:K));
        r_n = y - At(:,pos(1:K))*theta_ls;
        if norm(r_n)<1e-60
            break;
        end
    end
    theta(Pos_theta)=theta_ls;
end
```