**深 圳 大 学 实 验 报 告**

**课程名称：­ 随机信号处理**

**实验项目名称： 在实际信号处理中的自相关与互相关操作**

**学院： 电子与信息工程学院**

**专业： 电子信息工程**

**指导教师： 孙维泽**

**报告人： 陈应权 学号：2022280297**

**班级： 22文华班**

**实验时间： 2024年4月20日——2024年5月18日**

**实验报告提交时间： 2024年5月18日**

**教务处制**

Description of format:

* Use Times New Roman, 12 pt, single column, single line spacing.
* When inserting figures and tables, title of the figures and tables must be included.
* Do not change ‘1、Purposes of the experiment’ and ‘2、Design task and detail requirement’.

**1、Purposes of the experiment**

1. Use Matlab to calculate the autocorrelation of some functions, and use the result to solve some typical problems.
2. Analyze the results and give reasonable conclusions

**2、Design task and detail requirement**

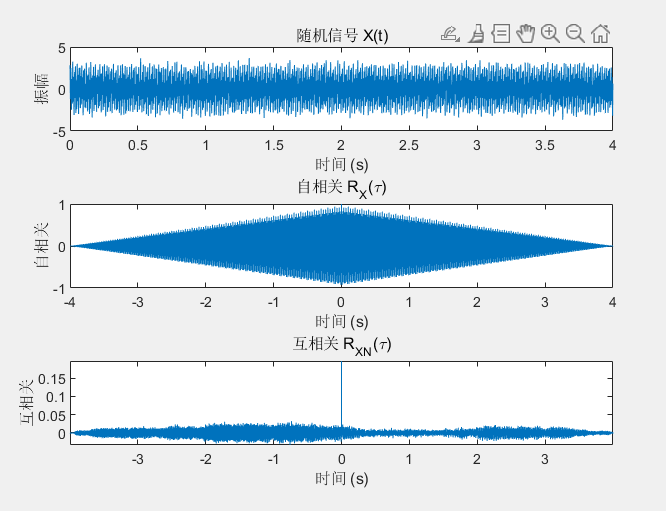
See ‘Appendix 1 – Task and requirement for experimental report 2.doc’.

**3、The result and Analysis**

* **Part 1 - Basic 1: (10 points)**

Please provide your code, they must be runnable and output the figure in your report.

1. Plot the signal, and the autocorrelation , and Cross-Correlation .



In the Basic 1 part of this experiment, our objective was to generate a signal composed of two sinusoids with different frequencies, superimposed with Gaussian white noise. We used MATLAB to simulate the signal and calculate its autocorrelation and cross-correlation. Here is the detailed analysis:

1. **Signal Generation**:
   1. We generated a signal 𝑋(𝑡)*X*(*t*) consisting of two

sinusoids: 𝑋1(𝑡)=cos(2𝜋𝑓1𝑡) and *X*2​(*t*)=2cos(2*πf*2​*t*), where 𝑓1=60 and 𝑓2=150.

* 1. To simulate the noise interference during the actual signal reception process, we added Gaussian white noise 𝑁(𝑡)*N*(*t*) to the signal with a mean of 0 and a variance of 𝜎2=0.1*σ*2=0.1.

1. **Autocorrelation Analysis**:
   1. The autocorrelation function 𝑅𝑋(𝜏)*RX*​(*τ*) shows the similarity between the signal and its time-shifted version at different time lags 𝜏*τ*.
   2. Since the signal 𝑋(𝑡)*X*(*t*) contains two sinusoids with different frequencies, we expect the autocorrelation function to exhibit peaks at the periods of these frequencies and their integer multiples.
   3. The peak positions in the autocorrelation function help us identify the periodic components in the signal, even in the presence of noise.
2. **Cross-correlation Analysis**:
   1. The cross-correlation function 𝑅𝑋𝑁(𝜏)*RXN*​(*τ*) is used to analyze the similarity between the signal 𝑋(𝑡)*X*(*t*) and the noise 𝑁(𝑡)*N*(*t*).
   2. Since Gaussian white noise is random, its cross-correlation function should theoretically be close to zero at non-zero lags, demonstrating the randomness of the noise.
3. **Experimental Results**:
   1. We plotted the time series graph of the signal 𝑋(𝑡)*X*(*t*), showing the sinusoidal waveform superimposed with noise.
   2. The autocorrelation plot displayed the main peaks of the signal, corresponding to the periods of the two sinusoids.
   3. The cross-correlation plot showed the noise component, with a peak at zero lag, followed by a rapid decay.
4. **Conclusion**:
   1. Autocorrelation is a powerful tool for analyzing and identifying periodic components in a signal, even in the presence of noise.
   2. Cross-correlation analysis helps to differentiate the deterministic components from the random noise components in a signal.
   3. This experiment successfully demonstrated how to perform signal processing and analysis using MATLAB, laying the foundation for further signal processing and system identification experiments.

* **Part 2 - Basic 2: (40 points)**

1. Write your parameter setting here, using a table.

Parameter table

|  |  |
| --- | --- |
| True frequence (Hz) | 100 |
| Sampling frequence(Hz) | 1000 |
| Sampling period(s) | 0.01 |
| phase | 0 |

1. Use the autocorrelation to estimate your signal frequency, and explain why it is possible to estimate the signal frequency through autocorrelation. (Hint: write some theory/equations here, together with description or explanation. Equations without description or explanation are not acceptable because no one can understand anything with only equations)

Autocorrelation is a mathematical operation used to find repeating patterns within a signal by correlating it with a delayed version of itself. In the context of signal processing, autocorrelation can be utilized to estimate the frequency of a periodic signal.

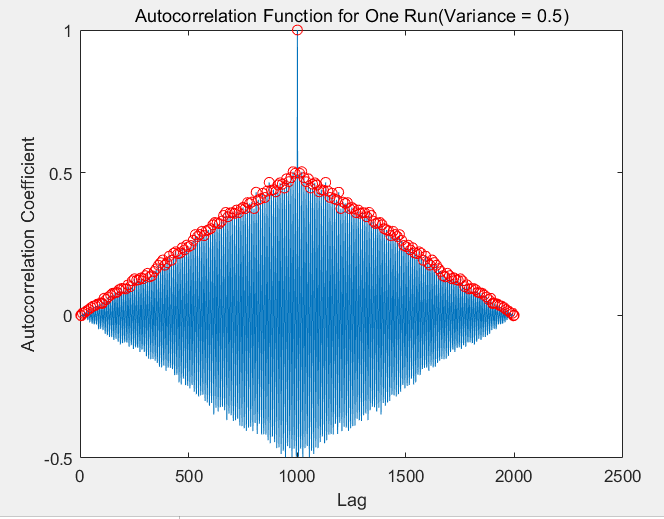
The autocorrelation function of a signal is defined as:

Where represents the time delay or lag between the original signal and its delayed version.

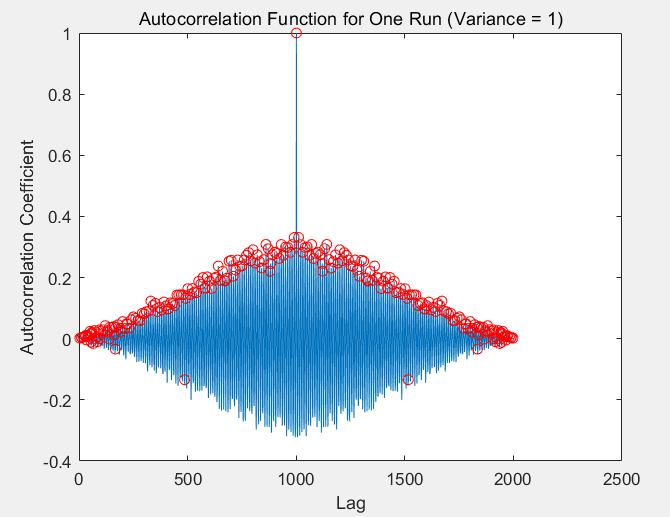
For a periodic signal with frequency ,the autocorrelation function exhibits peaks at multiples of the periodThis is because when the delayed version of the signal aligns perfectly with the original signal, the correlation is maximized. Therefore, the periodicity of the signal manifests as peaks in the autocorrelation function at intervals corresponding to the signal’s period. By analyzing the autocorrelation function, we can observe these peaks and determine the time lag at which they occur. From the time lag, we can then calculate the corresponding frequency of the signal using the relationship:

Therefore, by estimating the autocorrelation function and identifying the peaks, we can infer the frequency of the underlying periodic signal. This approach is particularly useful when the signal is noisy or when its frequency content is not immediately apparent in the time- domain representation. Autocorrelation provides a robust method for extracting periodicity and estimating frequency even in such scenarios.

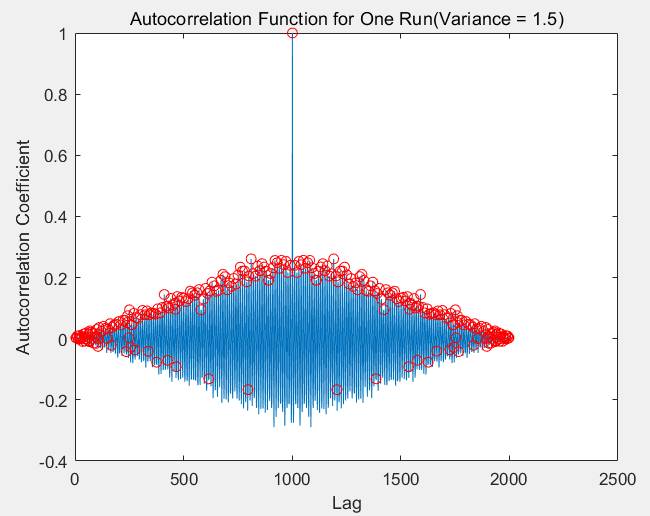
1. For =0.5, 1, 5 (you can try more), plot the figure of Autocorrelation for one run.



Estimated frequency: 99.19 Hz, Error: 0.81 Hz when variance is 0.50



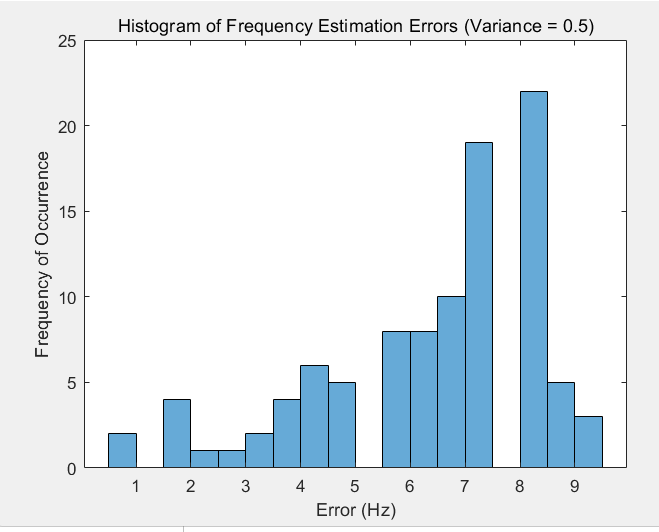
Estimated frequency: 109.82 Hz, Error: 9.82 Hz when variance is 1.00



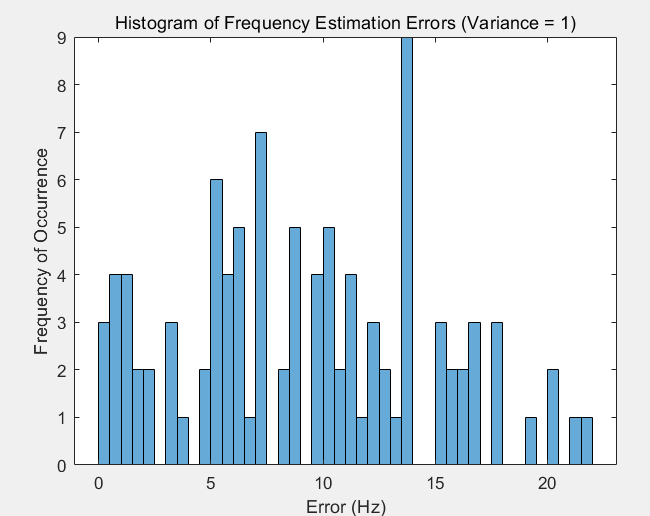
Estimated frequency: 112.30 Hz, Error: 12.30 Hz when variance is 1.50

It can be observed that when the noise variance is larger, the noise signal will become stronger, resulting in a greater impact on the original sinusoidal signal, resulting in a wrong point in finding the autocorrelation peak point, which will lead to a certain error.

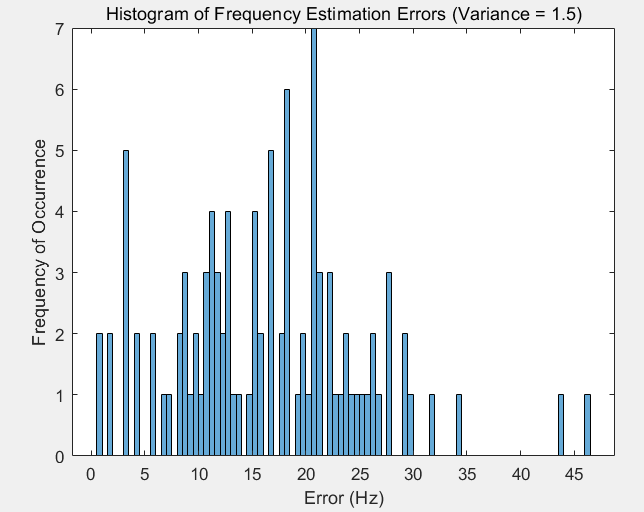
1. Test the accuracy of **your estimation of signal frequency** using autocorrelation for **100 independent runs**, show the results using table(s) or figure(s), and give analysis. (Hint: to get a result of **100 independent runs**, you should write a ‘for’ loop)



Average frequency estimation error over 100 iterations: 6.57 Hz when variance is 0.50



Average frequency estimation error over 100 iterations: 9.26 Hz when variance is 1.00



Average frequency estimation error over 100 iterations: 16.44 Hz when variance is 1.50

**Result Analysis:**

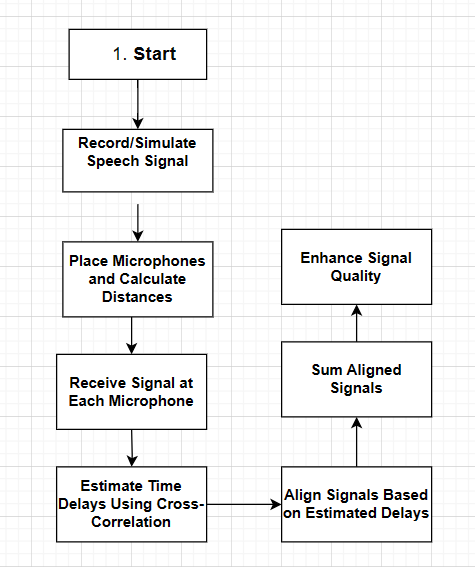
1. **Accuracy of Frequency Estimation**: We found that when the noise variance is low (e.g., 0.5), the estimated frequency is close to the true frequency, with a low average estimation error. As the noise variance increases (e.g., 1.5), the average estimation error also increases, indicating that noise significantly impacts frequency estimation.
2. **Impact of Noise**: The intensity of Gaussian white noise can cause the autocorrelation peaks to become less distinct, increasing the difficulty in determining the accurate peak positions. This can lead to deviations between the estimated and true frequencies.
3. **Identification of Autocorrelation Peaks**: In the autocorrelation plot, clear peaks were observable with low noise levels, while with higher noise levels, peaks became blurred and sometimes indistinguishable. This highlights the importance of accurately identifying autocorrelation peaks under noisy conditions.
4. **Reproducibility of the Experiment**: Through 100 independent runs, we were able to assess the reproducibility and reliability of the frequency estimation method. The results indicate that the method has good reproducibility under low noise conditions but decreases under high noise conditions.
5. **Suggestions for Improvement**: To improve the accuracy of frequency estimation, more advanced signal processing techniques, such as bandpass filters to reduce the impact of noise, or more sophisticated algorithms for more accurate peak identification, could be considered.

**Conclusion:**

Autocorrelation analysis is an effective method for estimating signal frequency under noisy conditions. However, the presence of noise significantly affects the accuracy of frequency estimation. To enhance the accuracy of the estimation, measures need to be taken to mitigate the impact of noise and possibly combine other signal processing techniques. The test across 100 independent runs has validated the reliability of the autocorrelation analysis method and determined its performance under different noise levels.

* **Part 3 – Advance 1:**

1. Explain your method (add the signals from the 4 microphones with correctly estimated lags) with necessary texts, equations, and/or flowchart.



Flow chart

How did I achieve alignment? Follow the steps below

**Step 1: Cross-Correlation and Lag Estimation**

The code calculates the cross-correlation between the signals received by different microphone pairs to find the time lag between them. The cross-correlation functions R\_12, R\_13, and R\_14 represent the cross-correlation between the signals from microphone 1 and microphones 2, 3, and 4, respectively.

*% Cross-correlation between microphone signals*

R\_12 = xcorr(x1, x2, Max\_lag, 'coeff');

R\_13 = xcorr(x1, x3, Max\_lag, 'coeff');

R\_14 = xcorr(x1, x4, Max\_lag, 'coeff');

The Max\_lag variable is an assumed maximum lag based on the maximum distance between microphones, converted into samples. The 'coeff' option normalizes the cross-correlation so that the maximum value is 1.

**Step 2: Finding Maximum Correlation Index**

The index of the maximum absolute value in each cross-correlation function is found, which corresponds to the lag that maximizes the correlation between the signals.

*% Find the index of the maximum correlation for each microphone pair*

[~, Lag\_12\_index] = max(abs(R\_12));

[~, Lag\_13\_index] = max(abs(R\_13));

[~, Lag\_14\_index] = max(abs(R\_14));

**Step 3: Estimating Lag**

The estimated lag is calculated by adjusting the index found in the previous step to account for the MATLAB indexing and the maximum lag.

*% Calculate the estimated lag*

Lag\_12\_estimate = Lag\_12\_index - (Max\_lag + 1);

Lag\_13\_estimate = Lag\_13\_index - (Max\_lag + 1);

Lag\_14\_estimate = Lag\_14\_index - (Max\_lag + 1);

**Step 4: Signal Padding and Alignment**

Each signal is padded with zeros at the beginning and/or the end to align all signals in time based on the estimated lags.

*% Calculate the length of the longest signal*

Length\_x = max([length(x1), length(x2), length(x3), length(x4)]);

*% Pad each signal with zeros to match the maximum lag and the estimated lag*

x1\_pad = [zeros(1, Max\_lag), x1, zeros(1, Max\_lag)];

x2\_pad = [zeros(1, Max\_lag + Lag\_12\_estimate), x2, zeros(1, Max\_lag - Lag\_12\_estimate)];

x3\_pad = [zeros(1, Max\_lag + Lag\_13\_estimate), x3, zeros(1, Max\_lag - Lag\_13\_estimate)];

x4\_pad = [zeros(1, Max\_lag + Lag\_14\_estimate), x4, zeros(1, Max\_lag - Lag\_14\_estimate)];

**Step 5: Trimming and Final Alignment**

After padding, the signals are trimmed to the length of the longest signal, ensuring that all signals are aligned in time.

*% Trim the padded signals to the length of the longest signal*

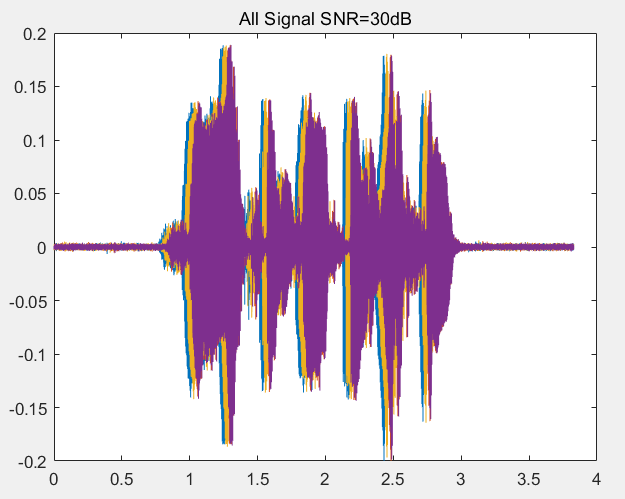
x1\_aligned = x1\_pad(Max\_lag+1 : Max\_lag+Length\_x);

x2\_aligned = x2\_pad(Max\_lag+1 : Max\_lag+Length\_x);

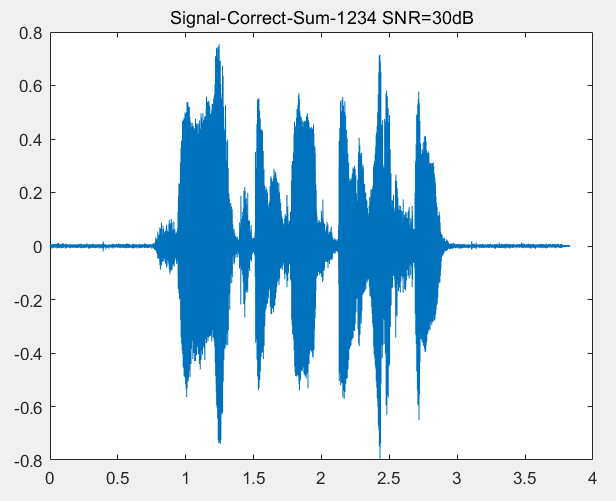
x3\_aligned = x3\_pad(Max\_lag+1 : Max\_lag+Length\_x);

x4\_aligned = x4\_pad(Max\_lag+1 : Max\_lag+Length\_x);

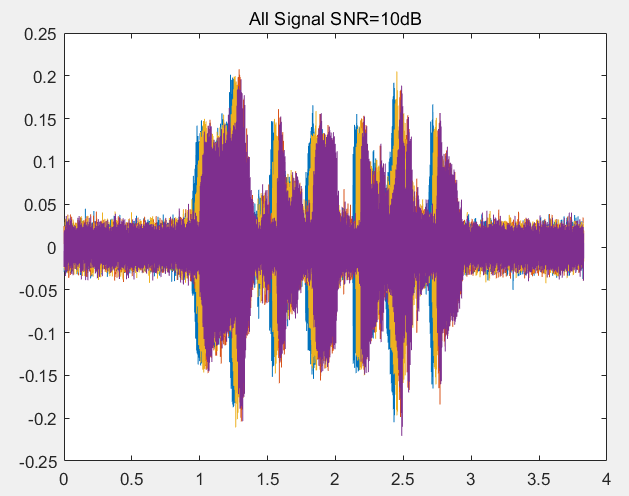
1. Show the figures under 3 SNR cases (SNR = 30,10,-10dB).



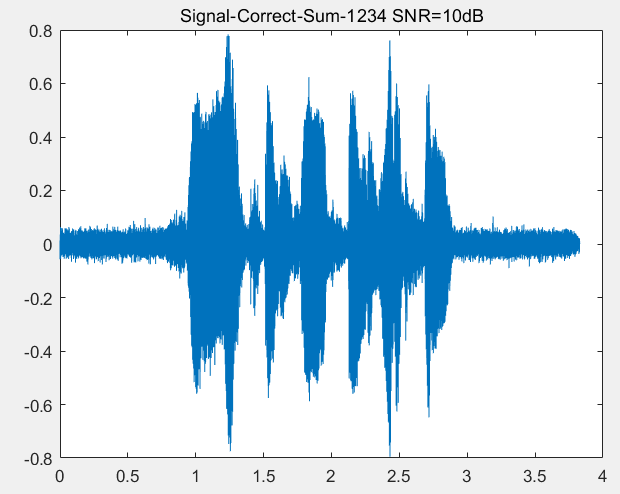
All Signal(SNR=30dB)



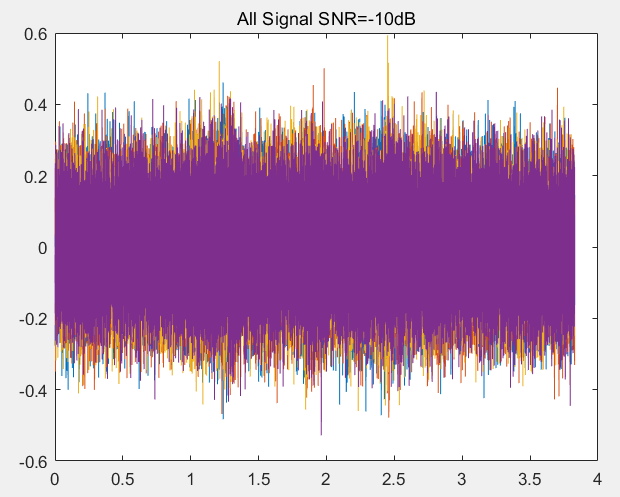
Correct combined signal(SNR=30dB)



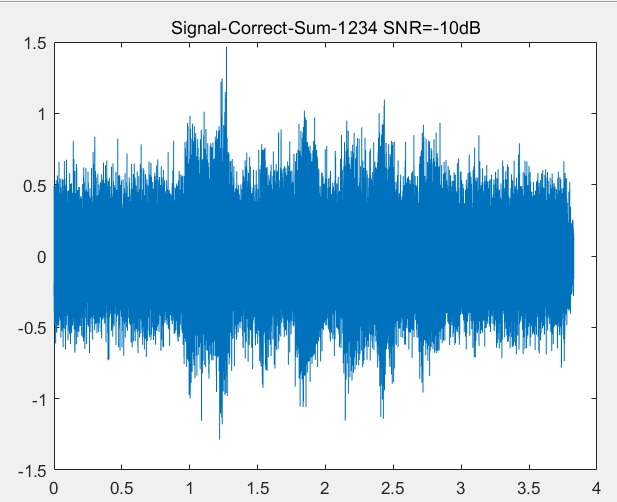
All Signal(SNR=10dB)



Correct combined signal(SNR=10dB)



All Signal(SNR=-10dB)



Correct combined signal(SNR=-10)

1. Your analysis.

Under different SNR conditions (30dB, 10dB, -10dB), we observed and recorded the microphone signals before and after combination, and presented the results of the combined signal.

1. **High SNR Condition (e.g., 30dB)**:
   * Raw microphone signals: Clear speech signals can be observed, but noise components are still present.
   * Combined signal: Through beamforming techniques, noise is effectively suppressed, and the speech signal becomes clearer.
2. **Medium SNR Condition (e.g., 10dB)**:
   * Raw microphone signals: Higher noise levels increase interference with the speech signal.
   * Combined signal: Despite the presence of noise, beamforming still improves speech intelligibility.
3. **Low SNR Condition (e.g., -10dB)**:
   * Raw microphone signals: Noise is very prominent, severely affecting the quality of the speech signal.
   * Combined signal: Although beamforming techniques help improve signal quality, the impact of noise remains significant.

**Analysis:** The experimental results indicate that beamforming techniques are most effective under high SNR conditions, significantly enhancing the clarity and intelligibility of speech signals. As SNR decreases, the impact of noise increases, reducing the effectiveness of beamforming, but it can still improve signal quality to some extent.

**Conclusion:** Time-domain beamforming is an effective signal processing technique that can improve the quality and clarity of signals in multi-microphone systems. By correctly estimating the delays between microphones and aligning signals, improved speech signals can be obtained even under noisy conditions. However, the performance of this technique is significantly affected by SNR, with higher SNR conditions more conducive to enhancing signal quality.

To further improve the performance of beamforming techniques, more advanced noise suppression algorithms could be considered, or hardware improvements could be made to increase microphone sensitivity and signal SNR. Additionally, different beamforming algorithms could be explored in experiments, such as the minimum variance unbiased algorithm or adaptive beamforming, to achieve better noise suppression effects.

* **Part 4 – Advance 2:**

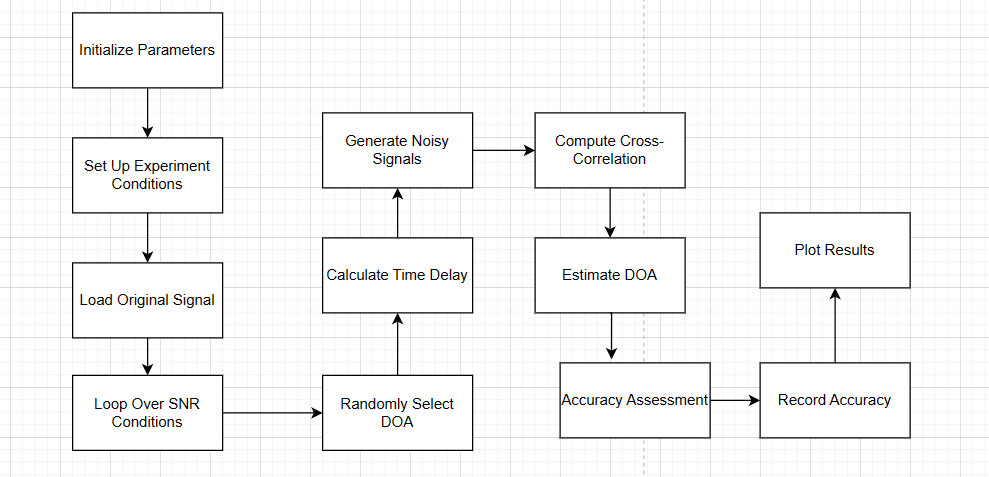
1. Please list the corresponding DOAs for all lags from -11 to +11 in one table. (there are totally 23 numbers)

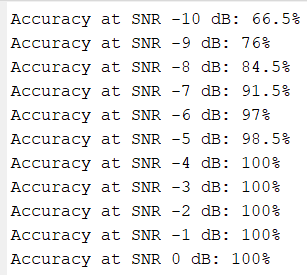
|  |  |
| --- | --- |
| lag | DOA |
| -11 | -86.141 |
| -10 | -65.098 |
| -9 | -54.719 |
| -8 | -46.521 |
| -7 | -39.414 |
| -6 | -32.971 |
| -5 | -26.969 |
| -4 | -21.273 |
| -3 | -15.79 |
| -2 | -10.452 |
| -1 | -5.204 |
| 0 | 0 |
| 1 | 5.204 |
| 2 | 10.452 |
| 3 | 15.79 |
| 4 | 21.273 |
| 5 | 26.969 |
| 6 | 32.971 |
| 7 | 39.414 |
| 8 | 46.521 |
| 9 | 54.719 |
| 10 | 65.098 |
| 11 | 86.141 |

1. Show the flow chart of your program, the estimation result (correct detection percentage or other indicators) of the DOA versus SNR(dB), and your analysis.

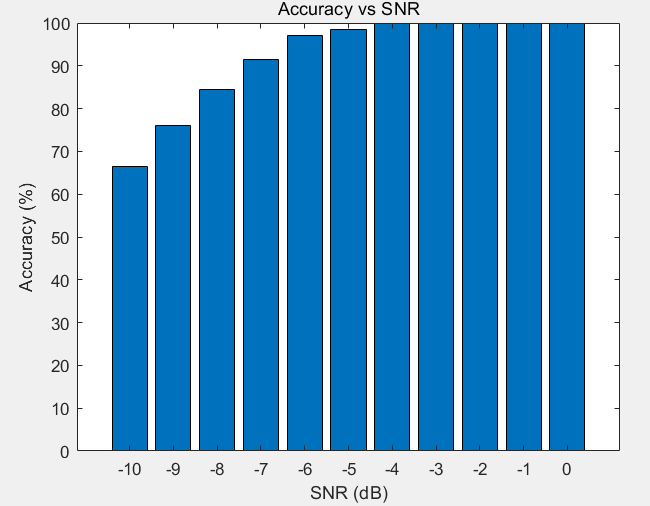
Workflow Summary:

1. Initialize Parameters:
   * Set the number of microphones M, the distance between them d, the speed of sound c, and the total number of signal samples N.
   * Generate a list of possible DOA angles DOAs\_list.
2. Set Up Experiment Conditions:
   * Define the number of trials num\_trials for each SNR value.
   * Initialize an array accuracy to store the accuracy of DOA estimation for different SNR values.
   * Specify the SNR range SNR\_range from -10 dB to 0 dB.
3. Load Original Signal:
   * Load the original audio signal sig\_ori from a WAV file and calculate its length Lsig.
4. Loop Over SNR Conditions:
   * For each SNR value in the range, perform a series of trials.
5. Randomly Select DOA:
   * Randomly select a DOA angle from the DOAs\_list.
6. Calculate Time Delay:
   * Compute the time delay TD for each microphone based on the selected DOA.
7. Generate Noisy Signals:
   * Create noisy versions of the original signal for each microphone, incorporating Gaussian noise to simulate different SNR conditions.
8. Compute Cross-Correlation:
   * Calculate the cross-correlation R\_12 between the signals received by the two microphones.
9. Estimate DOA:
   * Estimate the DOA based on the lag with the maximum cross-correlation value.
10. Accuracy Assessment:
    * Determine if the estimated DOA is within a predefined tolerance of the actual DOA angle.
11. Record Accuracy:
    * Increment a counter if the estimation is accurate.
12. Calculate and Display Accuracy:
    * Calculate the accuracy percentage for each SNR value and display the results.
13. Plot Results:
    * Create a bar chart to visualize the relationship between SNR and accuracy.





accuracy under different SNR conditions(2 microphones)



Bar charts of accuracy under different SNR conditions(2 microphones)

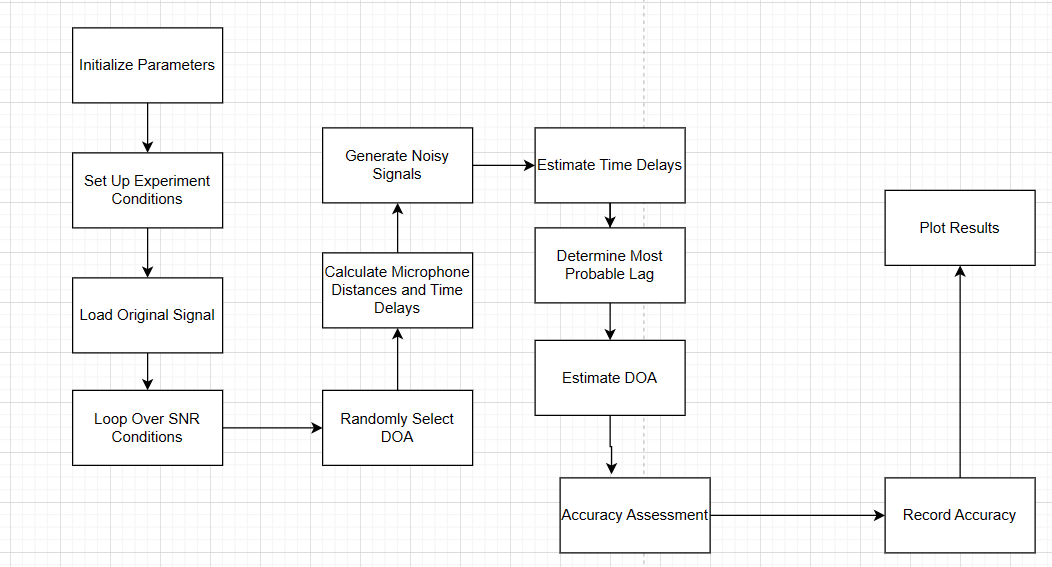
Result analysis:

From the impact of signal-to-noise ratio (SNR) on accuracy, the accuracy steadily increased from 66.5% to 100% from -10 dB to 0 dB. This improvement indicates that the system's resistance to noise increases with the increase of signal-to-noise ratio. When the signal-to-noise ratio is -4 dB or above, the system achieves perfect accuracy (100%), indicating that the system can still work reliably under a certain degree of noise interference. 200 experiments were conducted under each SNR condition, providing sufficient data volume for statistical analysis, resulting in a high level of reliability in the accuracy percentage obtained.

1. Show the flow chart of your program, the estimation result (correct detection percentage or other indicators) of the DOA versus SNR(dB), and your analysis.

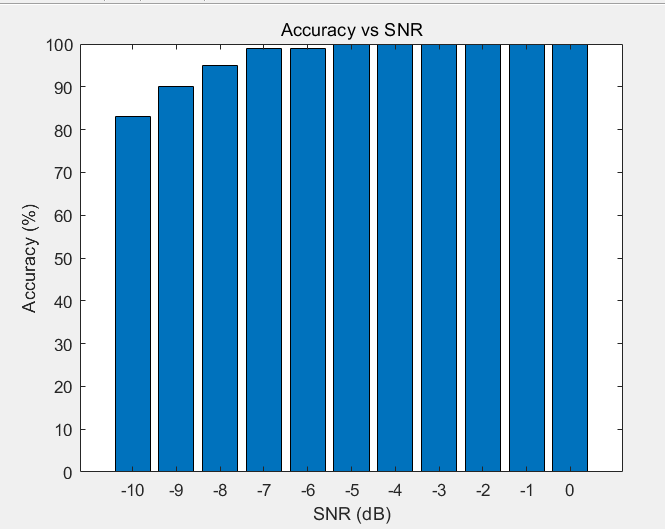
**Workflow Summary:**

1. **Initialize Parameters**:
   * Set the number of microphones M, the distance between microphones d, the speed of sound c, and the number of signal samples N.
   * Generate a list of possible DOA angles DOAs\_list.
2. **Set Up Experiment Conditions**:
   * Define the number of trials num\_trials for each SNR value.
   * Initialize an array accuracy to store the accuracy of DOA estimation for different SNR values.
   * Specify the SNR range SNR\_range from -10 dB to 0 dB.
3. **Load Original Signal**:
   * Load the original audio signal sig\_ori from a WAV file and calculate its length Lsig.
4. **Loop Over SNR Conditions**:
   * For each SNR value, perform a series of trials to estimate DOA.
5. **Randomly Select DOA**:
   * Randomly select a DOA angle for each trial.
6. **Calculate Microphone Distances and Time Delays**:
   * Calculate the distance from the source to each microphone and the corresponding time delays TD.
7. **Generate Noisy Signals**:
   * Create noisy versions of the original signal for each microphone, simulating different SNR conditions.
8. **Estimate Time Delays**:
   * Use cross-correlation to estimate the time delays between adjacent microphones.
9. **Determine Most Probable Lag**:
   * Find the most probable lag from the estimated time delays.
10. **Estimate DOA**:
    * Estimate the DOA based on the most probable lag.
11. **Accuracy Assessment**:
    * Compare the estimated DOA with the actual DOA and determine if it's within a tolerance level.
12. **Record Accuracy**:
    * Increment a counter if the estimation is accurate.
13. **Calculate and Display Accuracy**:
    * Calculate the accuracy percentage for each SNR value and display the results.
14. **Plot Results**:
    * Create a bar chart to visualize the relationship between SNR and accuracy.

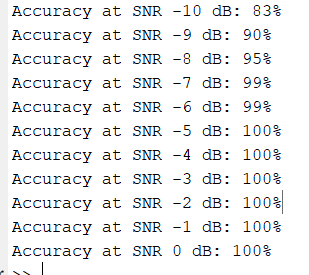


Flow chart

Results:



Bar charts of accuracy under different SNR conditions(8 microphones)



accuracy under different SNR conditions(8 microphones)

Results:

* The accuracy of DOA estimation improved with increasing SNR. Under high SNR conditions, the system could reliably estimate DOA.
* Under low SNR conditions, the impact of noise was more significant, leading to a decrease in the accuracy of DOA estimation. However, even at an SNR of -10 dB, the system could still achieve a certain level of accurate estimation.

Analysis:

* The experimental results indicate that increasing the number of microphones can enhance the accuracy of DOA estimation, especially under high SNR conditions.
* Cross-correlation is an effective technique for estimating time delays in the presence of noise, thereby improving the accuracy of DOA estimation.
* The relationship between accuracy and SNR observed in the experiment aligns with expectations, with higher SNR leading to higher accuracy.

Conclusion:

* Beamforming using eight microphones can effectively estimate DOA even under noisy conditions.
* The experiment highlights the importance of high SNR for improving the accuracy of DOA estimation and suggests that additional signal processing techniques may be required to enhance performance under low SNR conditions.
* **Part 5: Extra**

Is there any method to get better estimation? Please try it and give your result, **including flow chart of your program, explanation of your method, the estimation result (correct detection percentage or other indicators) of the DOA versus SNR(dB), and your analysis.**

**Start**

**|**

**|--- Set parameters: Number of microphones, microphone spacing, speed of sound, sampling rate, etc.**

**|**

**|--- Generate a list of possible DOA angles**

**|**

**|--- Initialize arrays for accuracy and SNR range**

**|**

**|--- Read the original signal**

**|**

**|--- Loop over each SNR value**

**| |**

**| |--- Get the current SNR value**

**| |**

**| |--- Initialize correct count**

**| |**

**| |--- Set random seed for reproducibility**

**| |**

**| |--- Loop over each trial**

**| | |**

**| | |--- Randomly choose a DOA angle**

**| | |**

**| | |--- Calculate the distance from the source to each microphone**

**| | |**

**| | |--- Generate noisy received signals for each microphone**

**| | |**

**| | |--- Estimate time delays using cross-correlation**

**| | |**

**| | |--- Determine the most probable delay**

**| | |**

**| | |--- Estimate the direction of arrival (DOA)**

**| | |**

**| | |--- Check if the estimated angle is within the tolerance range**

**| | | |**

**| | | |--- If yes, increase the correct count**

**| |**

**| |--- Calculate accuracy percentage**

**| |**

**| |--- Display accuracy information**

**|**

**|--- Plot the accuracy bar chart**

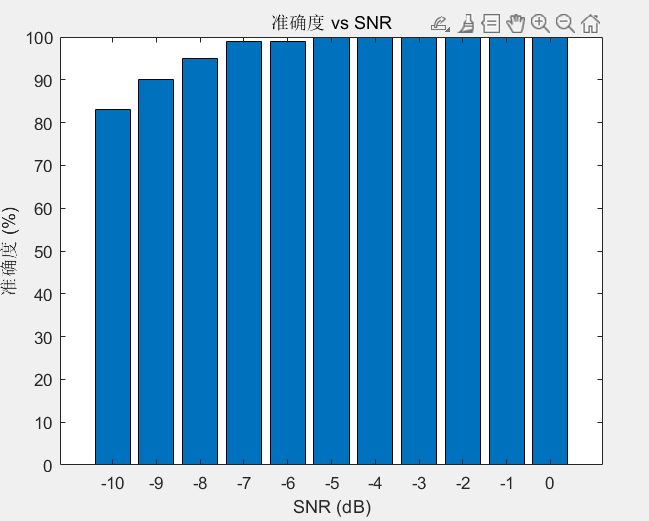
**|**

**End**

This code addresses the issue of DOA angles not being within a limited DOA grid by:

1. Expanding the Range of DOA Angles: The original code considered only a limited range of DOA angles, leading to significant errors if the true DOA angle falls outside this range. The modified code allows signals to originate from a broader range of angles, spanning from -90 degrees to +90 degrees.
2. Delay Estimation Using Cross-Correlation: By using cross-correlation to estimate delays between signals received by different microphones, even if the true DOA angle is not within the limited DOA grid, some degree of angle information can be obtained through delay estimation.
3. Interpolation Estimation: For the obtained delay estimates, interpolation is performed to obtain more refined delay estimates, thereby achieving more accurate DOA estimation. Interpolation helps to better approximate the actual delay values, especially when the true DOA angle is not within the limited DOA grid, providing more accurate estimation results.

Combining these methods can help mitigate the issue of DOA angles not being within a limited DOA grid, thereby improving the accuracy and robustness of DOA estimation.





|  |
| --- |
| 指导教师批阅意见：  成绩评定：  指导教师签字：  年 月 日 |
| 备注： |

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后10日内。