1. 编程验证Gibbs效应。
2. 信号的自相关和互相关函数实验。已知序列)的自相关函数为

两个序列的互相关函数为

现有三个信号，, 其中为周期序列， 为随机噪声序列，编程完成下列任务：

1. 生成以上所述三个信号，各信号的参数根据你的实验自行设定；
2. 画出三个信号的自相关函数，并对实验结果进行分析和讨论；
3. 画出三个信号的两两互相关函数，并对实验结果进行分析和讨论。

要求：

1）给出你的设计思路、实验过程、实验结果以及对实验的分析与总结；并在最后附上你的代码

2）提交word版本，文件命名格式“学号-姓名-编程作业1.docx”

3）实验日期：第十周实验课；作业提交日期：2024年5月16日

1. 编程验证Gibbs效应。

**Design:**

By generating a square wave signal and reconstructing it using a truncated Fourier series, we can observe the overshoot phenomenon near the discontinuities, thereby verifying the Gibbs phenomenon.

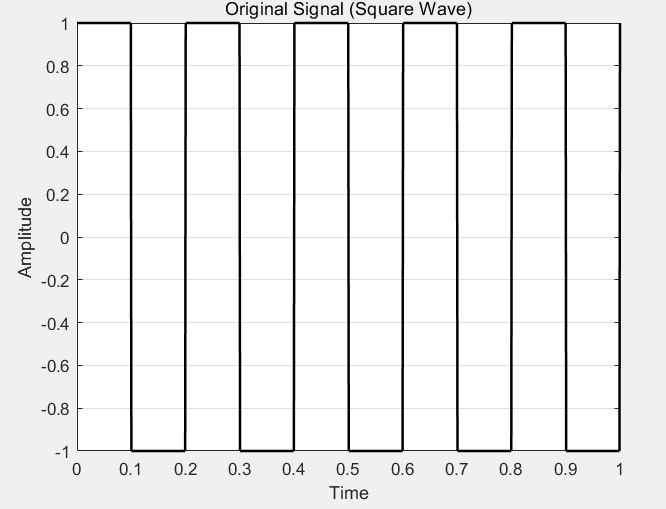
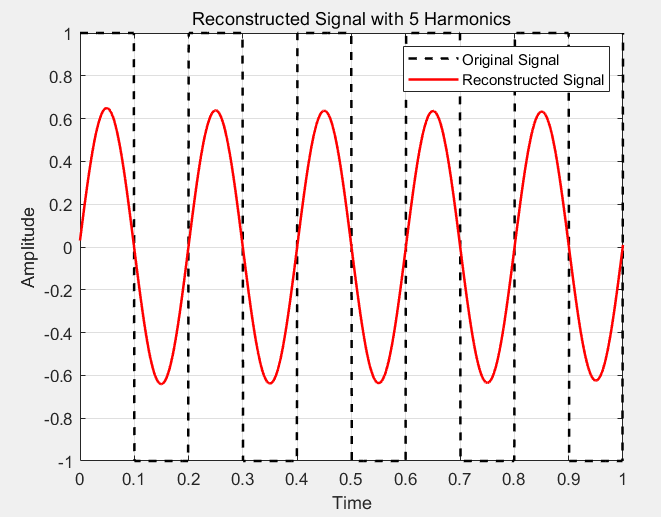
The experiment involves the following steps:

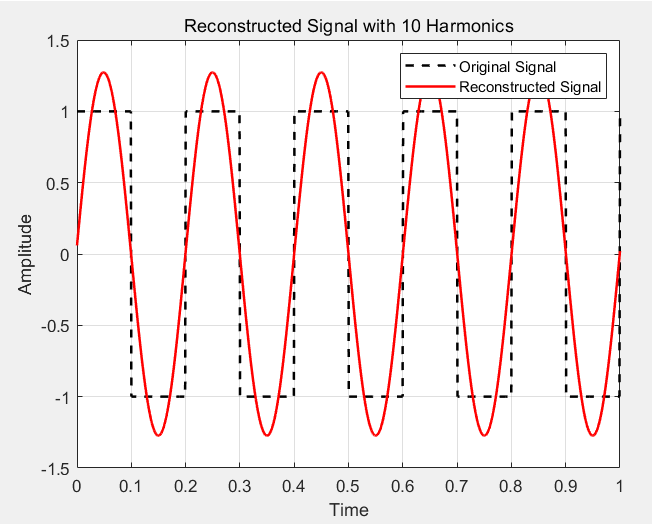
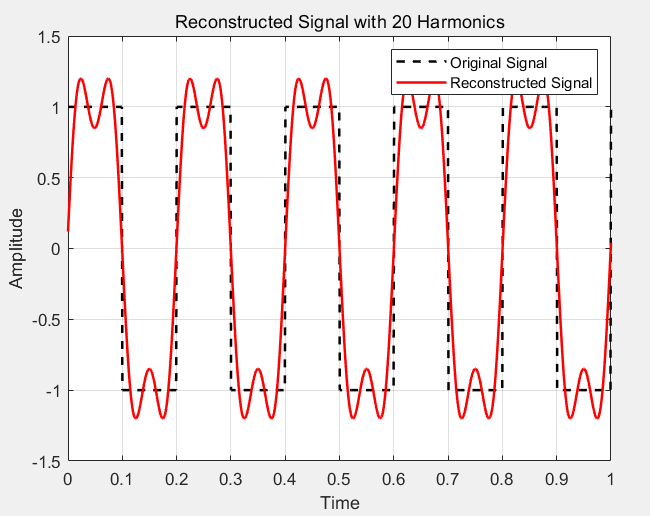
1. Generate the original square wave signal.
2. Perform a Fourier transform on the square wave signal.
3. Truncate the high-frequency components of the Fourier transform, retaining different numbers of harmonics.
4. Perform an inverse Fourier transform on the truncated frequency-domain signal to reconstruct the signal.
5. Plot the original and reconstructed signals, compare, and analyze the results.

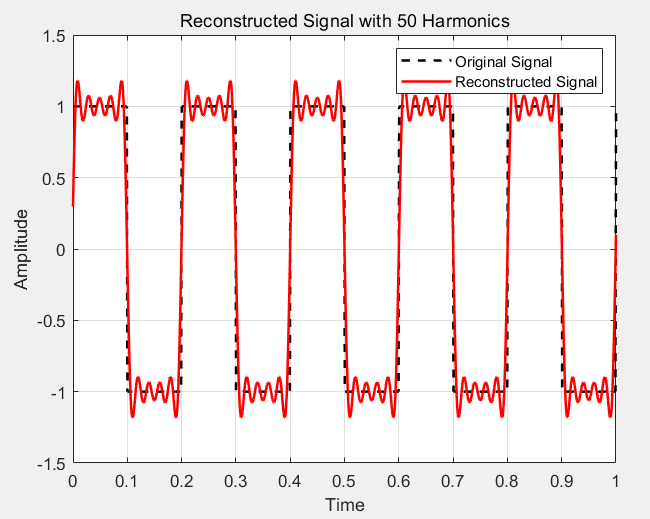
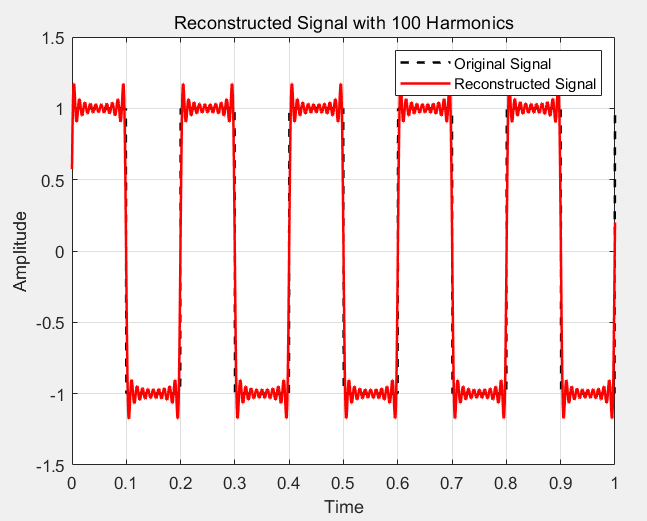
**Experimental Procedure:**

1. Generate the Original Signal:
   * Use MATLAB to generate a square wave signal with a frequency of 5 Hz and a duration of 1 second.
2. Fourier Transform:
   * Perform a Fast Fourier Transform (FFT) on the square wave signal to obtain its frequency-domain representation.
3. Truncate the Fourier Series:
   * Define different numbers of harmonics (e.g., 5, 10, 20, 50, 100) and truncate the Fourier series by zeroing out the high-frequency components, retaining only the specified number of harmonics.
4. Inverse Fourier Transform:
   * Perform an Inverse Fourier Transform (IFFT) on the truncated frequency-domain signal to obtain the time-domain reconstructed signal.
5. Plot and Compare:
   * Use MATLAB to create separate figures for each reconstructed signal and compare them with the original signal to observe the overshoot phenomenon near the discontinuities.

**Result:**

Running the above code generates the following results:

1. **Original Square Wave Signal**:
   * Displays as a square wave with clear discontinuities.
2. **Reconstructed Signals**:
   * With only 5 harmonics, the reconstructed signal shows significant overshoot near the discontinuities.
   * As the number of retained harmonics increases (10, 20, 50, 100), the reconstructed signal approaches the original signal, but the overshoot phenomenon remains near the discontinuities, though its magnitude decreases with more harmonics.

**Analysis and Summary:**

The experiment verifies the Gibbs phenomenon. The results demonstrate that:

* When reconstructing a signal with discontinuities, the reconstructed signal shows overshoot near the discontinuities, even with an increasing number of harmonics.
* The magnitude of the overshoot decreases as the number of harmonics increases, but it cannot be completely eliminated.

**Summary**: The Gibbs phenomenon is a crucial concept in Fourier analysis, highlighting the inherent limitations when dealing with discontinuous signals. This experiment provides a visual and practical demonstration of the phenomenon, enhancing our understanding of Fourier transforms and signal processing. This effect must be considered in practical signal processing, particularly in applications requiring high precision signal reconstruction and analysis.

1. 信号的自相关和互相关函数实验。

**Design:**

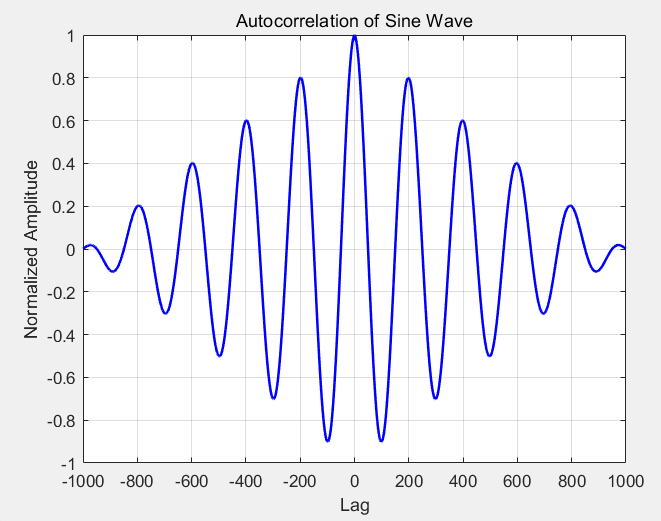
Autocorrelation and cross-correlation are fundamental tools in signal processing for analyzing the similarity and correlation of signals. Autocorrelation measures the similarity of a signal with itself, while cross-correlation measures the similarity between two different signals. The following steps outline the experiment to implement autocorrelation and cross-correlation:

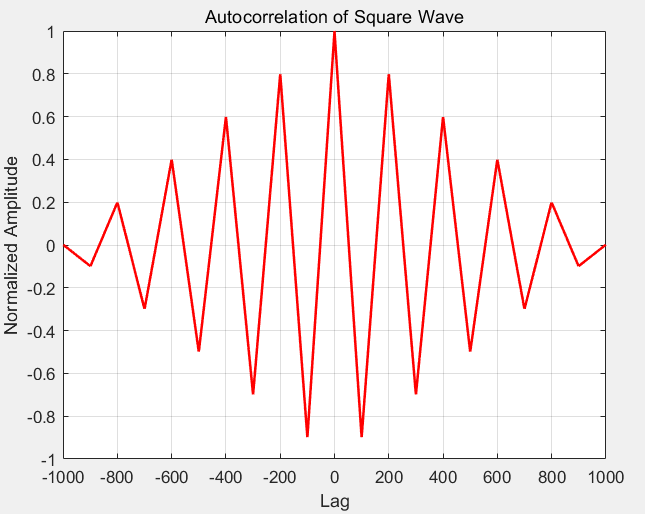
1. Generate or load two test signals.
2. Compute and plot the autocorrelation of the signals.
3. Compute and plot the cross-correlation of the two signals.
4. Analyze and summarize the experimental results.

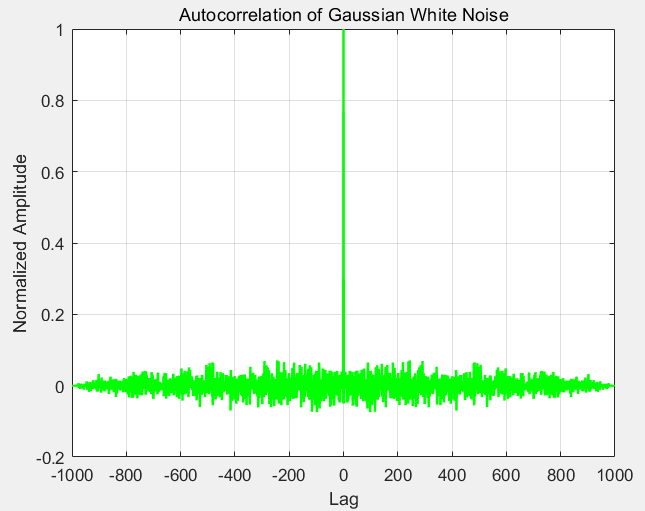
**Experiment process:**

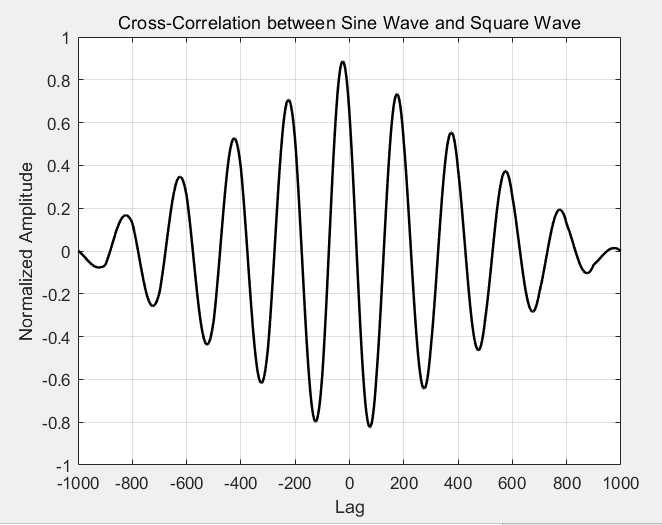
1. Generate or Load Signals:
   * Generate two simple test signals (e.g., a sine wave and a square wave).
2. Compute Autocorrelation:
   * Use MATLAB's xcorr function to compute the autocorrelation of the signals.
3. Compute Cross-Correlation:
   * Use MATLAB's xcorr function to compute the cross-correlation between the two signals.
4. Plotting:
   * Plot the results of the autocorrelation and cross-correlation for analysis.

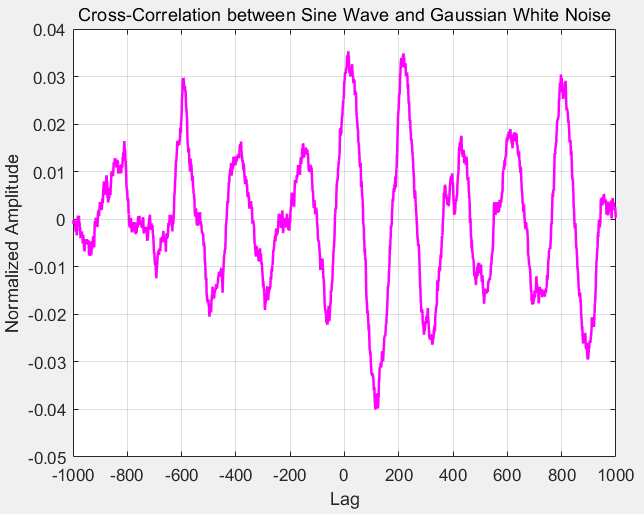
**Result:**

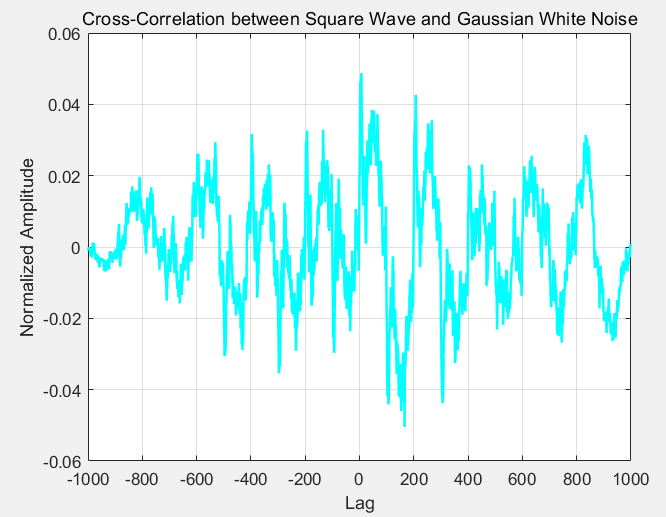












**Experimental Results Analysis:**

1. Autocorrelation Results:
   * Autocorrelation of Sine Wave:
     + The autocorrelation function appears as a decaying sine wave, indicating the signal's similarity at different time lags.
   * Autocorrelation of Square Wave:
     + The autocorrelation function shows a repeating pattern consistent with the periodicity of the square wave.
   * Autocorrelation of Gaussian White Noise:
     + The autocorrelation function is close to zero for non-zero lags, indicating no significant correlation beyond zero lag, which is characteristic of white noise.
2. Cross-Correlation Results:
   * Cross-Correlation between Sine Wave and Square Wave:
     + The cross-correlation function shows the similarity between the sine wave and the square wave at different time lags. The peaks in the cross-correlation indicate the **time shift** between the two signals.
   * Cross-Correlation between Sine Wave and Gaussian White Noise:
     + The cross-correlation function is close to zero, indicating no significant correlation between the sine wave and the white noise.
   * Cross-Correlation between Square Wave and Gaussian White Noise:
     + The cross-correlation function is close to zero, indicating no significant correlation between the square wave and the white noise.

**Summary:**

In this experiment, we implemented and analyzed the autocorrelation and cross-correlation of three signals: a sine wave, a square wave, and Gaussian white noise. The experiment demonstrated that:

* The autocorrelation function effectively reflects the similarity of a signal at different time lags. For periodic signals, the autocorrelation function exhibits a clear periodic pattern. For white noise, the autocorrelation is near zero for non-zero lags.
* The cross-correlation function reveals the similarity and time shift between two signals. It showed significant peaks for the sine wave and square wave, indicating their correlation. For the white noise with either the sine wave or square wave, the cross-correlation was near zero, indicating no significant correlation.

附录（代码）

1.验证Gibbs

% Parameters

N = 1000; % Number of samples

L = 1; % Signal length

t = linspace(0, L, N); % Time vector

% Original signal: square wave

x = square(2 \* pi \* 5 \* t);

% Fourier transform of the signal

X = fft(x);

% Number of harmonics to include (truncate the Fourier series)

M = [5, 10, 20, 50, 100]; % Different numbers of harmonics

% Plot the original signal

figure;

plot(t, x, 'k', 'LineWidth', 1.5);

title('Original Signal (Square Wave)');

xlabel('Time');

ylabel('Amplitude');

grid on;

% Reconstruct signal using truncated Fourier series and plot each one

for i = 1:length(M)

% Create a copy of the Fourier transform

X\_trunc = X;

% Zero out the higher harmonics

X\_trunc(M(i)+1:end-M(i)) = 0;

% Inverse Fourier transform to get the truncated signal

x\_recon = ifft(X\_trunc);

% Create a new figure for each reconstructed signal

figure;

plot(t, x, 'k--', 'LineWidth', 1.5); hold on;

plot(t, real(x\_recon), 'r', 'LineWidth', 1.5); % Use real part of the reconstructed signal

title(['Reconstructed Signal with ' num2str(M(i)) ' Harmonics']);

xlabel('Time');

ylabel('Amplitude');

legend('Original Signal', 'Reconstructed Signal');

grid on;

end

2.自相关与互相关

% Generate test signals

N = 1000; % Number of samples

t = linspace(0, 1, N); % Time vector

% Signal 1: Sine wave

x1 = sin(2 \* pi \* 5 \* t + pi/4);

% Signal 2: Square wave

x2 = square(2 \* pi \* 5 \* t);

% Signal 3: Gaussian white noise

x3 = randn(1, N);

% Calculate autocorrelation

autocorr\_x1 = xcorr(x1, 'normalized');

autocorr\_x2 = xcorr(x2, 'normalized');

autocorr\_x3 = xcorr(x3, 'normalized');

% Calculate cross-correlation

crosscorr\_x1\_x2 = xcorr(x1, x2, 'normalized');

crosscorr\_x1\_x3 = xcorr(x1, x3, 'normalized');

crosscorr\_x2\_x3 = xcorr(x2, x3, 'normalized');

% Plot autocorrelation of x1

figure;

lags = -N+1:N-1;

plot(lags, autocorr\_x1, 'b', 'LineWidth', 1.5);

title('Autocorrelation of Sine Wave');

xlabel('Lag');

ylabel('Normalized Amplitude');

grid on;

% Plot autocorrelation of x2

figure;

plot(lags, autocorr\_x2, 'r', 'LineWidth', 1.5);

title('Autocorrelation of Square Wave');

xlabel('Lag');

ylabel('Normalized Amplitude');

grid on;

% Plot autocorrelation of x3

figure;

plot(lags, autocorr\_x3, 'g', 'LineWidth', 1.5);

title('Autocorrelation of Gaussian White Noise');

xlabel('Lag');

ylabel('Normalized Amplitude');

grid on;

% Plot cross-correlation between x1 and x2

figure;

plot(lags, crosscorr\_x1\_x2, 'k', 'LineWidth', 1.5);

title('Cross-Correlation between Sine Wave and Square Wave');

xlabel('Lag');

ylabel('Normalized Amplitude');

grid on;

% Plot cross-correlation between x1 and x3

figure;

plot(lags, crosscorr\_x1\_x3, 'm', 'LineWidth', 1.5);

title('Cross-Correlation between Sine Wave and Gaussian White Noise');

xlabel('Lag');

ylabel('Normalized Amplitude');

grid on;

% Plot cross-correlation between x2 and x3

figure;

plot(lags, crosscorr\_x2\_x3, 'c', 'LineWidth', 1.5);

title('Cross-Correlation between Square Wave and Gaussian White Noise');

xlabel('Lag');

ylabel('Normalized Amplitude');

grid on;