**实验题目**

**实验一**

实验1：系统的频域和z域分析

设计计算机程序，产生序列并计算序列的DTFT，绘制其幅频特性和相频特性曲线；根据系统的单位脉冲响应和差分方程，计算系统的频率响应，绘制系统频率响应的幅频特性和相频特性曲线；根据系统的单位脉冲响应和差分方程，计算系统的系统函数、零极点分布；改变系统的零极点分布，观察系统频率响应的变化。

**实验二**

实验2：信号的频谱分析

设计计算机程序，产生序列并计算序列的FFT和IFFT，绘制其幅频特性和相频特性曲线；模拟产生离散系统的输入序列和单位脉冲响应，利用FFT和IFFT算法计算系统的输出响应，分析FFT的计算长度对系统输出响应的影响；模拟产生连续时间信号，选取适当的采样频率对其采样，并用FFT算法计算其频谱，分析信号的观测时间长度、FFT的计算长度对信号频谱计算结果的影响。

**实验三**

实验3：IIR数字滤波器设计及结构

设计计算机程序，根据滤波器的主要技术指标设计IIR数字巴特沃斯和切比雪夫低通、高通、带通和带阻滤波器；绘制滤波器的幅频特性和相频特性曲线，验证滤波器的设计结果是否达到设计指标要求；画出数字滤波器的直接型、级联型、并联型结构信号流图。

课外学习：课外利用编程工具，设计和分析IIR数字滤波器；设计实验内容和实验步骤，编写上机实验程序，撰写实验报告。

**实验四**

实验4：FIR数字滤波器设计及结构

设计计算机程序，根据滤波器的主要技术指标设计线性相位FIR数字低通、高通、带通和带阻滤波器；绘制滤波器的幅频特性和相频特性曲线，验证滤波器的设计结果是否达到设计指标要求；画出线性相位FIR数字滤波器的结构信号流图。

数字信号处理

实验报告

|  |  |
| --- | --- |
| 班级： |  |
| 姓名： |  |
| 学号： |  |
| 联系方式： |  |

西安电子科技大学

电子工程学院

目录

[摘要 4](#_Toc148809995)

[绪论 4](#_Toc148809996)

[1. 发展历史 4](#_Toc148809997)

[2. 现状 5](#_Toc148809998)

[3. 研究意义 5](#_Toc148809999)

[正文 5](#_Toc148810000)

[实验目的 5](#_Toc148810001)

[实验原理与方法 6](#_Toc148810002)

[实验内容及步骤 7](#_Toc148810003)

[实验内容 7](#_Toc148810004)

[实验步骤 8](#_Toc148810005)

[实验结果分析及结论总结 9](#_Toc148810006)

[第一部分 9](#_Toc148810007)

[第二部分 10](#_Toc148810008)

[第三部分 11](#_Toc148810009)

[心得及展望 13](#_Toc148810010)

[实验心得： 13](#_Toc148810011)

[展望： 14](#_Toc148810012)

[附件 14](#_Toc148810013)

# **摘要**

本报告总结了数字信号处理课程中的频谱分析实验。实验通过**Python**编程实现了信号的快速傅里叶变换、卷积运算仿真、系统频率响应分析等内容。主要实验结果如下:

1)成功生成正弦信号并利用**FFT**算法计算其频谱,绘制出信号的幅频特性和相频特性曲线,通过**IFFT**验证了**FFT**的正确性。

2)仿真离散系统的输入和单位冲激响应,采用**不同长度**的FFT计算系统频率响应,分析了FFT长度对输出响应的影响。结果表明,FFT长度与频域分辨率和**计算效率相关**。

3)模拟采集连续信号,选取适当采样频率进行离散化,通过改变信号的**观测长度**和**FFT计算长度**,分析了对频谱计算的影响。结果显示,这两者都影响频谱分辨率。

**关键词:数字信号处理;频谱分析;快速傅里叶变换;卷积;采样频率**

# **绪论**

数字信号处理（DSP）是一门涉及数字信号的采集、处理和分析的领域，具有广泛的应用，包括通信、音频处理、图像处理、生物医学工程、雷达、控制系统等。以下是数字信号处理的发展历史、现状和研究意义的概述：

## 1. 发展历史

20世纪初，DSP的基础开始奠定，主要用于通信领域。

20世纪60年代，数字计算机的普及使DSP技术更加实际可行。

20世纪70年代和80年代，DSP在音频、图像处理和通信领域取得显著进展。

20世纪90年代，DSP应用领域不断扩展，包括生物医学、雷达、声纳、控制系统等。

21世纪以来，DSP在无线通信、多媒体处理、机器学习等领域得到广泛应用，技术不断演进。

## 2. 现状

- DSP技术已经深入到许多领域，如5G通信、自动驾驶、智能音箱、数字音乐、医学成像等。

- 近年来，DSP在人工智能和机器学习中的应用迅速增长，包括图像识别、语音识别、自然语言处理等。

- 嵌入式DSP处理器的性能不断提高，使得实时信号处理变得更加高效。

- DSP技术也与硬件加速器（如GPU和FPGA）结合，提供更大的计算能力。

## 3. 研究意义

改善信号质量：DSP可以用于降噪、滤波和增强信号，提高信息的质量和可用性。

实时处理：DSP可以在实时应用中处理信号，如通信系统、自动控制和传感器网络。

数据分析：DSP可以用于提取信号中的有用信息，帮助决策制定和模式识别。

通信：DSP在调制、解调、错误检测和纠正等方面对通信领域至关重要。

医疗应用：DSP用于医学成像、心电图分析、生物信号处理等，有助于医疗诊断和疾病监测。

人工智能：DSP与机器学习结合，推动了图像、语音和文本处理等领域的发展。

总的来说，数字信号处理在现代科技和工程领域扮演着至关重要的角色。它不仅帮助改善了现有技术的性能，还推动了新技术的发展，为人类社会的进步和创新提供了强大的工具。

# 正文

## 实验目的

理解频谱分析的基本原理： 通过本实验，学生将理解频谱分析的基本原理，包括傅立叶变换和卷积运算，以及如何使用FFT和IFFT算法进行频谱分析。

学习如何模拟离散系统： 学生将学会模拟离散系统，包括创建输入序列和单位脉冲响应，以便进一步分析系统的输出响应。

分析FFT计算长度的影响： 通过变化FFT计算长度，学生将了解计算长度对频谱分析的影响，包括分辨率和计算时间。

学会进行信号采样和频谱分析： 学生将学习如何对连续时间信号进行采样，然后使用FFT算法计算信号的频谱，以分析频谱中的主要成分。

## 实验原理与方法

1. 生成序列并计算FFT：

原理： 傅立叶变换用于将时域信号转换为频域信号，FFT（快速傅立叶变换）是一种高效的算法，用于计算离散序列的频谱。

方法：编写程序生成一个离散序列，例如正弦波、方波或随机信号。

使用FFT算法计算该序列的频谱。

绘制序列的幅频特性和相频特性曲线。

2. 生成离散系统的输入序列和单位脉冲响应：

原理： 离散系统的行为可以描述为卷积运算，其中输入信号与系统的单位脉冲响应卷积得到输出信号。

方法：模拟一个离散系统，例如一个差分方程或差分方程的系数。

创建一个输入序列，可以是随机信号或特定信号。

使用卷积运算，计算系统的单位脉冲响应。

3. 利用FFT和IFFT计算系统的输出响应：

原理： 使用FFT和IFFT来计算卷积，通过卷积定理，可以将卷积运算转换为频域的乘法运算。

方法：使用输入序列和单位脉冲响应，通过FFT计算系统的输出响应。

绘制系统的输出响应的幅频特性和相频特性曲线。

4. 分析FFT计算长度对系统输出响应的影响：

原理： FFT计算长度影响频谱计算的分辨率，以及计算时间。

方法：对相同的输入序列，使用不同的FFT计算长度（例如256点、512点），比较输出响应的差异。

观察输出响应的分辨率、计算时间等变化，以分析FFT计算长度对输出的影响。

5. 模拟产生连续时间信号和采样：

原理： 连续时间信号可以通过采样转换为离散信号，采样频率应满足奈奎斯特采样定理。

方法：选择一个连续时间信号，例如正弦波。

选择适当的采样频率，确保采样频率至少是信号频率的两倍。

通过离散采样，将连续时间信号转换为离散序列。

6. 使用FFT算法计算信号的频谱：

原理： 使用FFT算法可以高效地计算离散信号的频谱。

方法：对采样后的离散序列使用FFT算法，计算信号的频谱。

绘制信号的频谱，分析频谱中的主要成分。

7. 分析信号的观测时间长度和FFT计算长度的影响：

原理： 观测时间长度和FFT计算长度会影响频谱分析的精度和计算效率。

方法：改变信号的观测时间长度，比较不同长度下的频谱计算结果。

同样，改变FFT计算长度，观察对频谱的影响。

## 实验内容及步骤

### 实验内容

**实验2.1：计算序列的FFT和IFFT**

1. **生成输入序列：** 首先，生成一个离散信号序列，你可以选择正弦波、方波、随机信号等。这将作为输入信号进行处理。
2. **FFT计算：** 使用FFT算法计算输入序列的频谱。你可以使用现成的FFT库或编写自己的FFT函数。
3. **绘制幅频特性和相频特性曲线：** 绘制输入序列的幅频特性和相频特性曲线。你可以使用绘图库（例如Matplotlib）来可视化表示这些频谱特性。
4. **IFFT计算：** 为了验证FFT的正确性，执行逆FFT（IFFT）操作，将频谱反变换回原始序列。
5. **绘制IFFT的结果：** 绘制IFFT后的序列，确保它与原始输入序列相匹配。

**实验2.2：模拟离散系统和卷积**

1. **定义离散系统：** 模拟一个离散系统，这可以是差分方程或差分方程的系数。例如，你可以定义一个简单的数字滤波器。
2. **创建输入序列：** 创建一个输入序列，可以使用随机信号、单位脉冲、或者其他合适的输入信号。
3. **卷积计算：** 使用卷积运算计算系统的输出响应。这可以通过将输入信号和系统响应进行卷积来实现。
4. **FFT计算系统的输出：** 使用FFT算法计算系统的输出响应的频谱。
5. **绘制幅频特性和相频特性曲线：** 绘制系统的输出响应的幅频特性和相频特性曲线，以可视化表示系统的频谱特性。

**实验2.3：分析FFT的计算长度对系统输出响应的影响**

1. **改变FFT计算长度：** 对相同的输入序列，使用不同的FFT计算长度（例如256点、512点）进行FFT计算。
2. **比较结果：** 比较不同计算长度下的输出响应的差异。观察输出响应的分辨率、计算时间等变化，以分析FFT计算长度对输出的影响。

**实验2.4：模拟产生连续时间信号和频谱分析**

1. **选择连续时间信号：** 选择一个连续时间信号，例如正弦波。
2. **选择采样频率：** 选择适当的采样频率，确保采样频率至少是信号频率的两倍，以避免混叠。
3. **进行采样：** 通过离散采样，将连续时间信号转换为离散序列。这通常涉及将信号与采样脉冲进行卷积或进行直接采样。

**实验2.5：使用FFT算法计算信号的频谱**

1. **FFT计算频谱：** 对采样后的离散序列使用FFT算法，计算信号的频谱。
2. **绘制频谱：** 绘制信号的频谱，以可视化表示信号的频域特性。

**实验2.6：分析信号的观测时间长度和FFT的计算长度对频谱计算结果的影响**

1. **改变观测时间长度：** 改变信号的观测时间长度，比较不同长度下的频谱计算结果。
2. **改变FFT计算长度：** 同样，改变FFT计算长度，观察对频谱的影响。

### 实验步骤

生成序列并计算FFT：

选择一个离散序列，这可以是模拟信号或实际采集的数据。

使用编程工具（如MATLAB、**Python**的**NumPy**/SciPy库等）编写程序来计算序列的FFT。

绘制FFT的幅频特性和相频特性曲线。

生成离散系统的输入序列和单位脉冲响应：

模拟一个离散系统，例如差分方程或差分方程的系数。

生成输入序列，这可以是随机信号或特定信号。

计算系统的单位脉冲响应，这可以使用系统的差分方程。

利用FFT和IFFT计算系统的输出响应：

使用输入序列和单位脉冲响应，通过卷积计算系统的输出响应。这可以使用FFT和逆FFT（IFFT）来加速计算。

绘制系统的输出响应的幅频特性和相频特性曲线。

分析FFT计算长度对系统输出响应的影响：

对相同的输入序列，使用不同的FFT计算长度，比较输出响应的差异。可以尝试不同的FFT长度，例如256点、512点等。

观察输出响应的分辨率、计算时间等变化，以分析FFT计算长度对输出的影响。

模拟产生连续时间信号和采样：

选择一个连续时间信号，例如正弦波或方波。

选取适当的采样频率，根据奈奎斯特采样定理，确保信号的采样频率至少是信号频率的两倍。

通过离散采样，将连续时间信号转换为离散序列。

使用FFT算法计算信号的频谱：

对采样后的离散序列使用FFT算法，计算信号的频谱。

绘制信号的频谱，分析信号频谱中的主要成分。

分析信号的观测时间长度和FFT计算长度的影响：

改变信号的观测时间长度，比较不同长度下的频谱计算结果。

同样，改变FFT计算长度，观察对频谱的影响。

## 实验结果分析及结论总结

### 第一部分

**绘制FFT的幅频特性和相频特性曲线。**

**生成离散系统的输入序列和单位脉冲响应：**

信号：

# 生成输入序列（示例为一个正弦波信号）

N = 256  # 序列长度

n = np.arange(N)  # 时间索引

f = 5.0  # 正弦波的频率

x = np.sin(2 \* np.pi \* f \* n / N)  # 生成正弦波信号

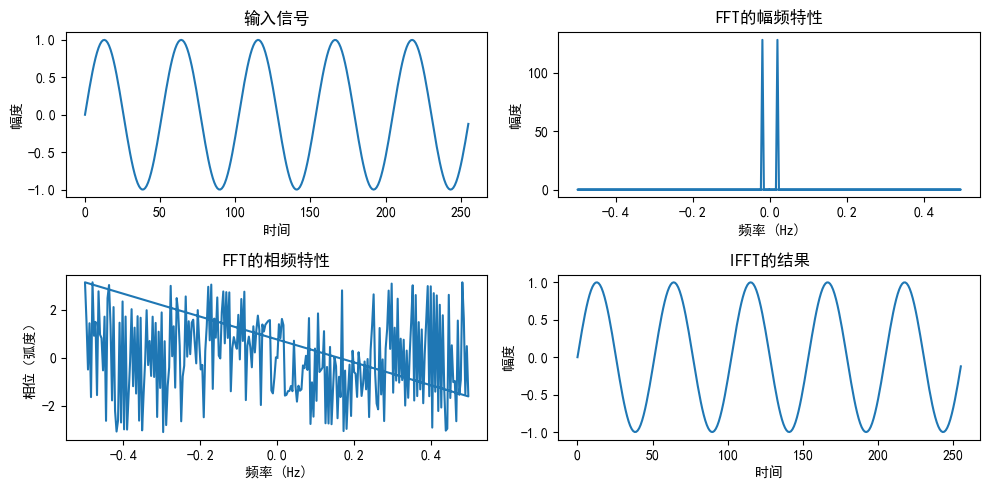


图1信号幅度谱和相位谱

|  |
| --- |
| 关键代码 |
| X = np.fft.fft(x)  x\_reconstructed = np.fft.ifft(X)  fs = 1.0 # 采样频率  frequencies = np.fft.fftfreq(N, 1/fs) |

### 第二部分

|  |
| --- |
| # 生成输入序列（示例为一个正弦波信号）  N = 256  # 序列长度  n = np.arange(N)  # 时间索引  f = 5.0  # 正弦波的频率  x = np.sin(2 \* np.pi \* f \* n / N)  # 生成正弦波信号  # 生成单位脉冲响应  impulse\_response = np.zeros(N)  impulse\_response[0] = 1 |

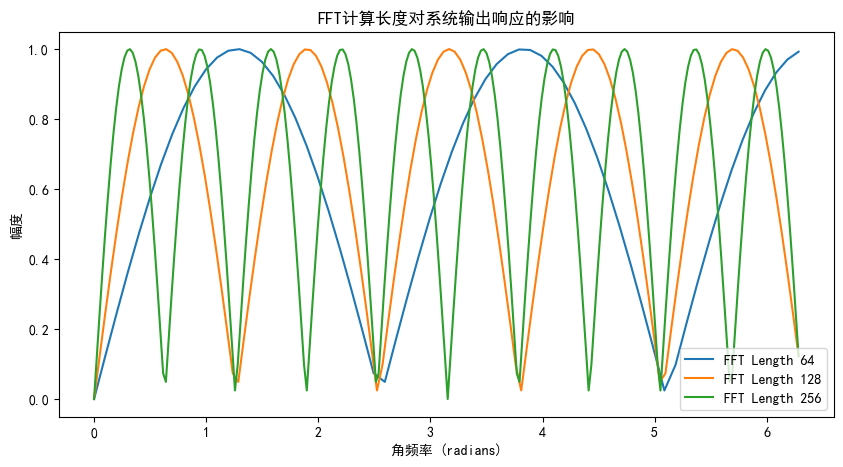


图2 第二部分绘图

|  |
| --- |
| 关键代码： |
| # 计算FFT      X = np.fft.fft(x, fft\_length)      H = np.fft.fft(impulse\_response, fft\_length)      # 计算系统的输出响应频谱      Y = X \* H      # 计算系统的输出响应时域      y = np.fft.ifft(Y, fft\_length)      # 绘制输出响应的幅频特性      w = np.linspace(0, 2 \* np.pi, fft\_length)      plt.plot(w, np.abs(y), *label*=*f*'FFT Length {fft\_length}') |

FFT（快速傅里叶变换）的计算长度对系统输出响应的影响通常涉及到频域分辨率和计算效率两个主要方面。以下是对FFT计算长度对系统输出响应的影响的分析：

1. **频域分辨率**：

**较短的FFT计算长度**：使用较短的FFT计算长度会导致较低的频域分辨率。这意味着在频谱图中，你将难以区分紧密的频率成分，因为FFT无法提供足够的细节来准确表示它们。频域分辨率越低，你将看到更宽的频率峰值，可能会模糊系统的频率响应。

**较长的FFT计算长度**：使用较长的FFT计算长度将提供更高的频域分辨率，因为FFT具有更多的频率点。这允许你更好地分辨频率成分，并提供更准确的频率响应信息。然而，较长的FFT可能需要更多的计算资源。

1. **计算效率**：

**较短的FFT计算长度**：使用较短的FFT计算长度可以显著提高计算效率，因为需要计算的点数较少。这适用于实时系统或需要快速计算的应用。然而，这可能会降低频域分辨率。

**较长的FFT计算长度**：较长的FFT计算长度可能需要更多的计算时间和内存。这在一些计算资源受限的环境下可能不切实际。但它提供了更准确的频域分辨率。

### 第三部分

|  |
| --- |
| 关键代码： |
| # 绘制频谱图  plt.figure(*figsize*=(12, 8))  for observation\_length in observation\_lengths:      for fft\_length in fft\_lengths:          num\_samples = int(observation\_length \* fs)          signal\_segment = discrete\_signal[:num\_samples]          # 计算FFT          spectrum = np.abs(np.fft.fft(signal\_segment, fft\_length))          # 计算频率轴          freq\_axis = np.fft.fftfreq(fft\_length, 1/fs)          # 绘制频谱          plt.subplot(len(observation\_lengths), len(fft\_lengths), observation\_lengths.index(observation\_length) \* len(fft\_lengths) + fft\_lengths.index(fft\_length) + 1)          plt.plot(freq\_axis, spectrum)          plt.title(*f*'Observation Time: {observation\_length}s, FFT Length: {fft\_length}')          plt.xlabel('Frequency (Hz)')          plt.ylabel('Magnitude')  plt.tight\_layout()  plt.show() |

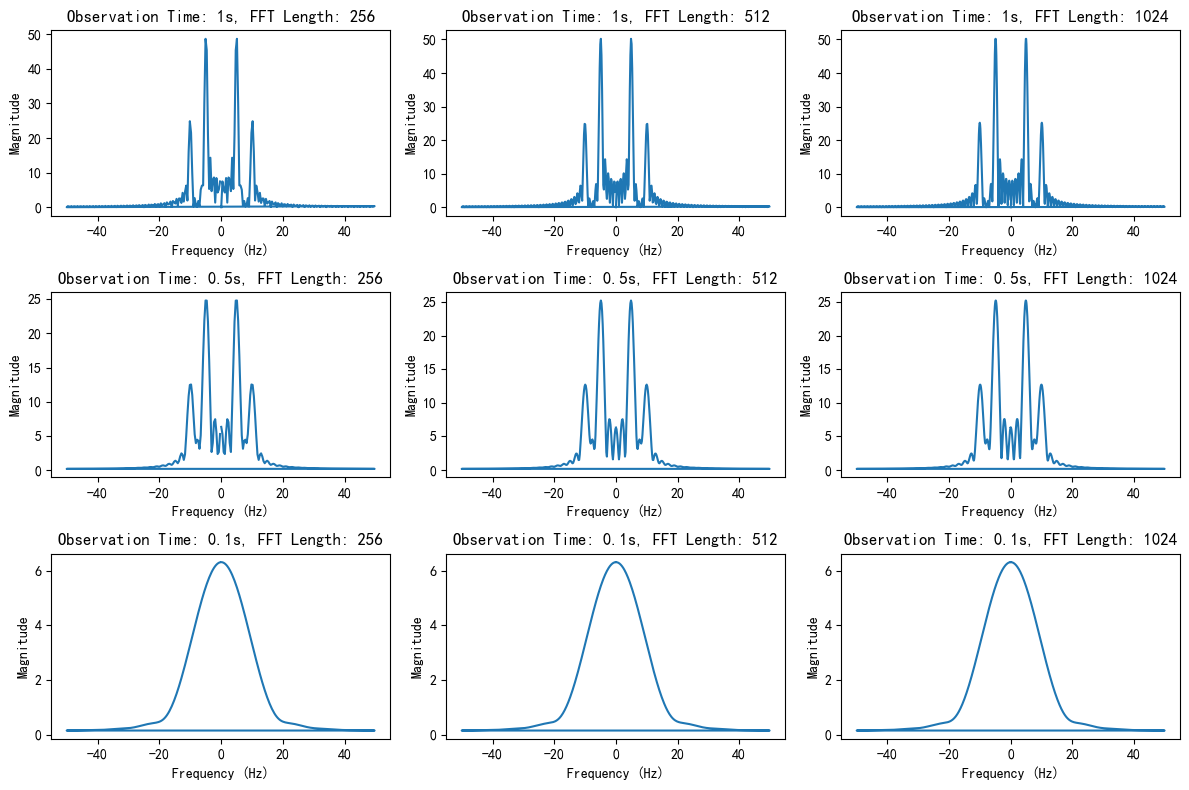


图3 变化前后幅频特性变化

1. **信号的观测时间长度：**

**短时间观测**：如果观测时间长度较短，即采样时间有限，会导致频谱分析的频率分辨率较低。这意味着我们将无法准确分辨信号中不同频率分量的特性。频谱图中的峰值可能会模糊，难以精确定位。

**长时间观测**：较长的观测时间长度可以提高频率分辨率，允许我们更准确地分析信号中的频率分量。然而，长时间观测也可能导致频谱图中的时间窗口过长，无法捕捉信号的快速变化。这可能会导致在频谱图中看到信号特性的平滑化。

1. **FFT的计算长度：**

**短FFT计算长度**：如果选择较短的FFT计算长度，频谱分析将对信号的细节和快速变化缺乏分辨能力。频谱图可能会显示出截断效应，导致频谱图中的泄漏现象，即频谱分量在频谱图中出现在不应该出现的频率上。

**长FFT计算长度**：较长的FFT计算长度可以提高频谱分辨率，使我们能够更好地分辨不同频率分量。这有助于减小频谱图中的泄漏效应。然而，计算较长的FFT可能需要更多的计算时间和内存。

## 心得及展望

### 实验心得：

在本次实验中，我学到了很多关于信号处理和频谱分析的基本概念和方法。以下是我从实验中获得的一些关键收获：

FFT和IFFT的理解： 通过编写和使用FFT算法，我深入了解了傅里叶变换和傅里叶逆变换的原理。这使我能够将时域信号转换为频域，并在需要时反转回时域。

频谱分析： 我学会了如何使用FFT来分析信号的频谱特性，包括幅频特性和相频特性。这对于了解信号的频域特性非常重要，特别是在信号处理和通信领域。

卷积和离散系统： 通过模拟离散系统和卷积操作，我了解了信号在系统中的传递和响应过程。这对于了解信号传输和过滤在实际应用中的作用非常有帮助。

### 展望：

在实验结束后，我也有一些展望，可以进一步扩展我的知识和研究：

深入学习信号处理算法： 我计划深入学习更多信号处理算法，包括滤波、时域和频域分析方法。这将帮助我更好地理解不同类型的信号和数据。

应用于实际问题： 我希望将所学的信号处理和频谱分析技能应用于解决实际问题，例如音频处理、图像处理或通信系统优化。这将有助于将理论知识转化为实际应用。

研究新兴领域： 信号处理和频谱分析在人工智能、机器学习和大数据分析等新兴领域中扮演着重要的角色。我期待进一步探索这些领域，以了解如何将信号处理技术与现代数据科学相结合。

## 附件

|  |
| --- |
| 代码1 |
| import numpy as np  import matplotlib.pyplot as plt  # 生成输入序列（示例为一个正弦波信号）  N = 256  # 序列长度  n = np.arange(N)  # 时间索引  f = 5.0  # 正弦波的频率  x = np.sin(2 \* np.pi \* f \* n / N)  # 生成正弦波信号  # 计算FFT  X = np.fft.fft(x)  # 计算IFFT  x\_reconstructed = np.fft.ifft(X)  # 计算频率轴  fs = 1.0  # 采样频率  frequencies = np.fft.fftfreq(N, 1/fs)  # 绘制输入信号  plt.figure(*figsize*=(10, 5))  plt.subplot(2, 2, 1)  plt.plot(n, x)  plt.title('输入信号')  plt.xlabel('时间')  plt.ylabel('幅度')  # 绘制FFT的幅频特性  plt.subplot(2, 2, 2)  plt.plot(frequencies, np.abs(X))  plt.title('FFT的幅频特性')  plt.xlabel('频率 (Hz)')  plt.ylabel('幅度')  # 绘制FFT的相频特性  plt.subplot(2, 2, 3)  plt.plot(frequencies, np.angle(X))  plt.title('FFT的相频特性')  plt.xlabel('频率 (Hz)')  plt.ylabel('相位（弧度）')  # 绘制IFFT的结果  plt.subplot(2, 2, 4)  plt.plot(n, x\_reconstructed)  plt.title('IFFT的结果')  plt.xlabel('时间')  plt.ylabel('幅度')  plt.tight\_layout()  plt.show() |
|  |

|  |
| --- |
| 代码2 |
| import numpy as np  import matplotlib.pyplot as plt  # 生成输入序列（示例为一个正弦波信号）  N = 256  # 序列长度  n = np.arange(N)  # 时间索引  f = 5.0  # 正弦波的频率  x = np.sin(2 \* np.pi \* f \* n / N)  # 生成正弦波信号  # 生成单位脉冲响应  impulse\_response = np.zeros(N)  impulse\_response[0] = 1  # 选择不同的FFT计算长度  fft\_lengths = [64, 128, 256]  # 绘制系统输出响应的幅频特性  plt.figure(*figsize*=(10, 5))  for fft\_length in fft\_lengths:      # 计算FFT      X = np.fft.fft(x, fft\_length)      H = np.fft.fft(impulse\_response, fft\_length)      # 计算系统的输出响应频谱      Y = X \* H      # 计算系统的输出响应时域      y = np.fft.ifft(Y, fft\_length)      # 绘制输出响应的幅频特性      w = np.linspace(0, 2 \* np.pi, fft\_length)      plt.plot(w, np.abs(y), *label*=*f*'FFT Length {fft\_length}')  plt.legend()  plt.title('FFT计算长度对系统输出响应的影响')  plt.xlabel('角频率 (radians)')  plt.ylabel('幅度')  plt.show() |

|  |
| --- |
| 代码3 |
| import numpy as np  import matplotlib.pyplot as plt  # 生成连续时间信号  t\_continuous = np.linspace(0, 1, 1000)  # 连续时间范围从0到1秒  continuous\_signal = np.sin(2 \* np.pi \* 5 \* t\_continuous) + 0.5 \* np.sin(2 \* np.pi \* 10 \* t\_continuous)  # 选择采样频率  fs = 100  # 采样频率为100 Hz  t\_discrete = np.arange(0, 1, 1/fs)  # 生成离散时间点  discrete\_signal = np.interp(t\_discrete, t\_continuous, continuous\_signal)  # 采样连续时间信号  # 选择不同的观测时间长度  observation\_lengths = [1, 0.5, 0.1]  # 观测时间长度分别为1秒，0.5秒，0.1秒  # 选择不同的FFT计算长度  fft\_lengths = [256, 512, 1024]  # FFT计算长度分别为256点，512点，1024点  # 绘制频谱图  plt.figure(*figsize*=(12, 8))  for observation\_length in observation\_lengths:      for fft\_length in fft\_lengths:          num\_samples = int(observation\_length \* fs)          signal\_segment = discrete\_signal[:num\_samples]          # 计算FFT          spectrum = np.abs(np.fft.fft(signal\_segment, fft\_length))          # 计算频率轴          freq\_axis = np.fft.fftfreq(fft\_length, 1/fs)          # 绘制频谱          plt.subplot(len(observation\_lengths), len(fft\_lengths), observation\_lengths.index(observation\_length) \* len(fft\_lengths) + fft\_lengths.index(fft\_length) + 1)          plt.plot(freq\_axis, spectrum)          plt.title(*f*'Observation Time: {observation\_length}s, FFT Length: {fft\_length}')          plt.xlabel('Frequency (Hz)')          plt.ylabel('Magnitude')  plt.tight\_layout()  plt.show() |