西安电子科技大学

数字信号处理 实验报告

实验名称 系统的频域和 z 域分析

人工智能 学院 23200XX 23200XX 班

姓名 X X X 学号 2300920XXX

X X X 2300920XXX

X X X 2200920XXX

实验报告内容基本要求及参考格式

一、实验目的

二、实验所用仪器（或实验环境）

三、实验基本原理及步骤

四、实验结果

五、实验代码

六、实验问题及解决方案

张三：

李四：

王五：

一、实验目的

1. 掌握离散序列的 FFT/IFFT 计算方法并验证其可逆性；

2. 绘制指数序列的幅频特性和相频特性曲线；

3. 利用 FFT 实现快速卷积计算线性时不变系统的输出响应；

4. 分析 FFT 计算长度（零填充）对系统输出精度和频谱分辨率的影响；

5. 通过对连续时间指数衰减信号采样，观察采样频率、观测时长、FFT 长度对频谱估计质量的影响。

二、实验所用仪器（或实验环境）

1.需要环境为：Python 3.x

2.需要安装的 Python 库：numpy, matplotlib, scipy

三、实验基本原理及步骤

1.实验基本原理

(1). 指数序列 $\boldsymbol{x[n]=\alpha^n(|\alpha|<1)}$ 的 DTFT 具有典型的低通特性。

(2). 线性卷积的频域等价形式：$\boldsymbol{Y[k]=X[k]\cdot H[k]}$有限长度 FFT 相当于对信号加矩形窗，观测时长决定频率分辨率，零填充长度决定谱线密度。

(3). 连续指数信号 $\boldsymbol{x(t)=e^{-\alpha t}}$ 的连续时间傅里叶变换幅度谱为 $\frac{\boldsymbol{1}}{\sqrt{\boldsymbol{\alpha^2+(2\pi f)^2}}}$

2.实验步骤

(1). 指数序列的 FFT/IFFT 分析（含重建误差与极坐标图）

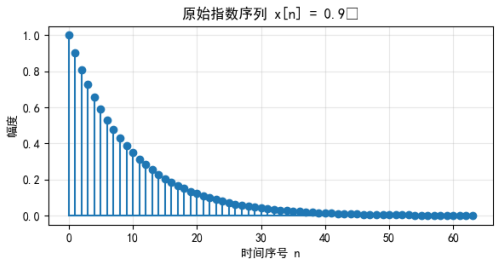
(2). 利用 FFT 计算 IIR 系统输出响应（与时域卷积对比）

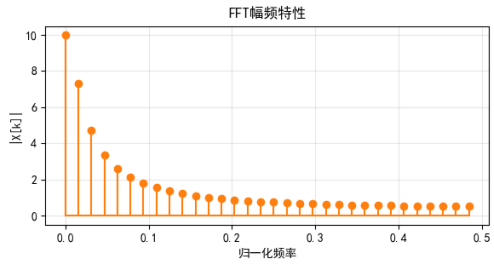
(3). 不同 FFT 长度对快速卷积精度的影响

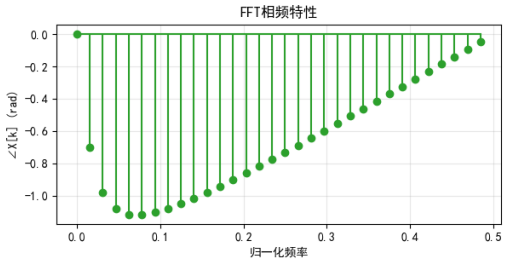
(4). 连续指数信号在不同采样率下的采样效果展示

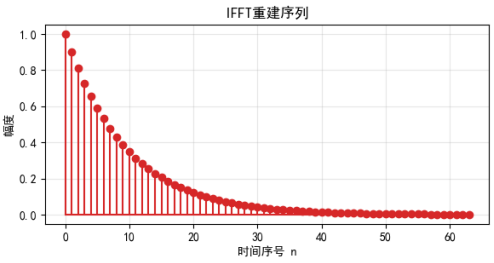
(5). 观测时长与 FFT 长度对频谱估计精度的综合影响（与理论频谱对比）

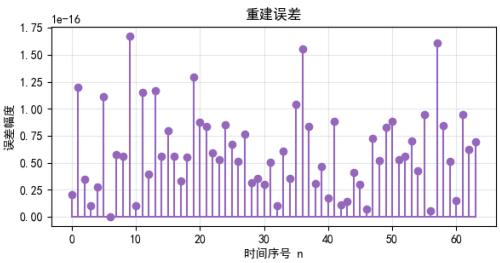
四、实验结果

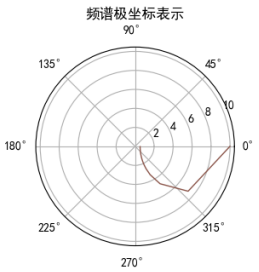

原始指数序列 x[n] = 0.9□
幅度
时间序号 n

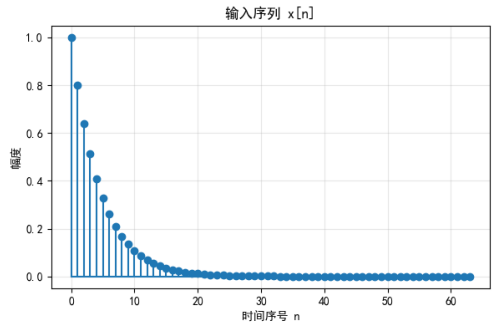

FFT幅频特性
|X[k]|
归一化频率

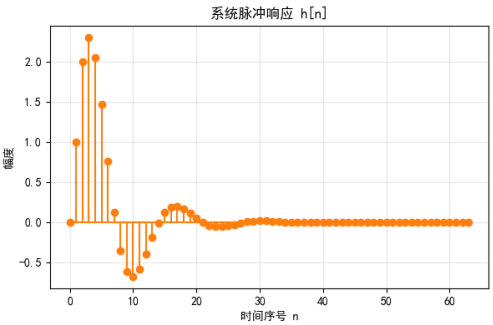

FFT相频特性
∠X[k] (rad)
归一化频率

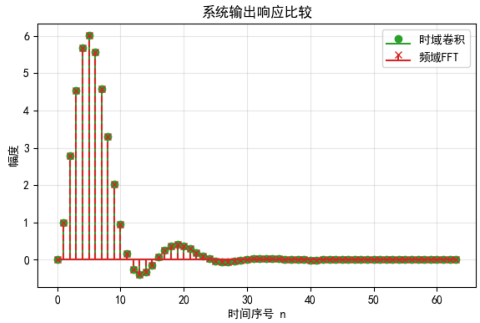

IFFT重建序列
幅度
时间序号 n

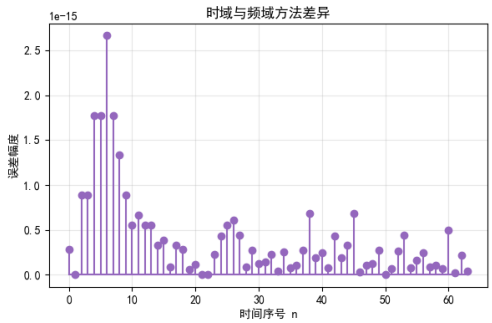

重建误差
1e-16
误差幅度
时间序号 n

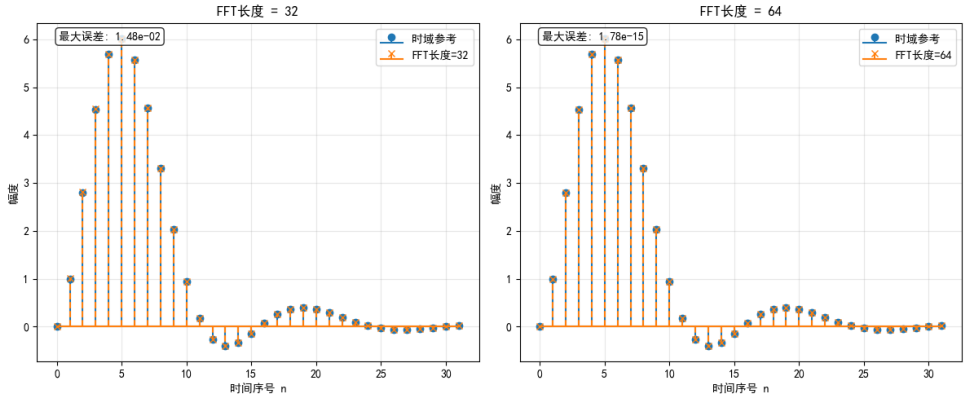

频谱极坐标表示


输入序列 x[n]
幅度
时间序号 n

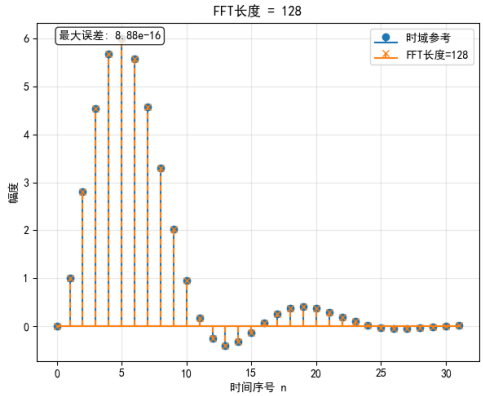

系统脉冲响应 h[n]
幅度
时间序号 n

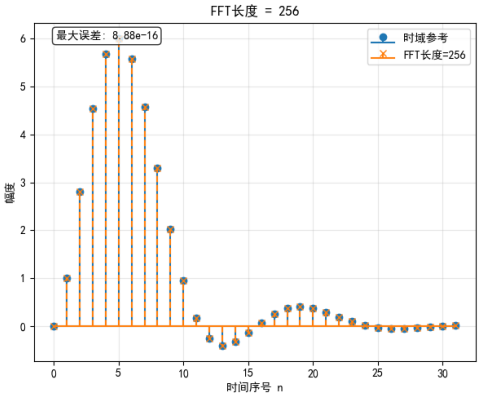

系统输出响应比较
时域卷积
频域FFT
幅度
时间序号 n

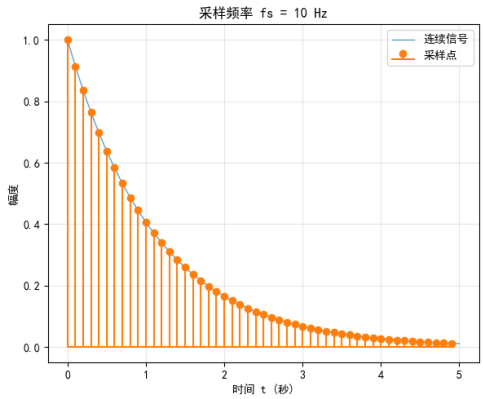

时域与频域方法差异
1e-15
误差幅度
时间序号 n

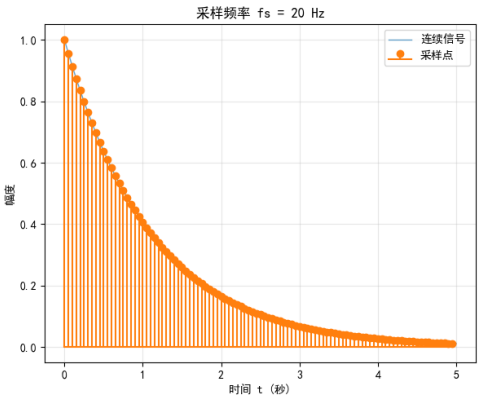

FFT长度 = 32
最大误差: 1.48e-02
时域参考
FFT长度=32
幅度
时间序号 n

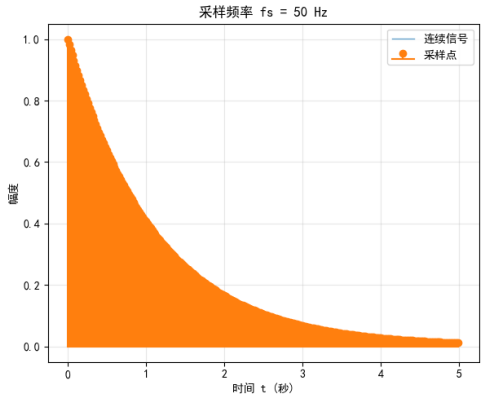
FFT长度 = 64
最大误差: 1.78e-15
时域参考
FFT长度=64
幅度
时间序号 n

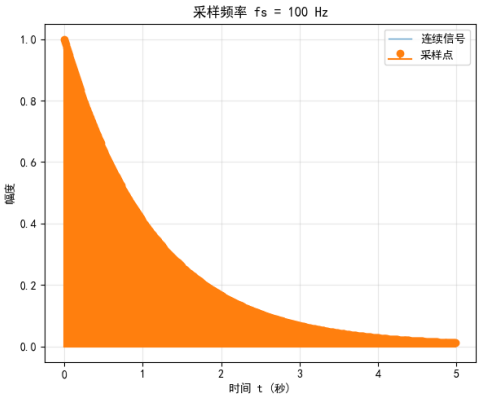

FFT长度 = 128
最大误差: 8.88e-16
时域参考
FFT长度=128
幅度
时间序号 n

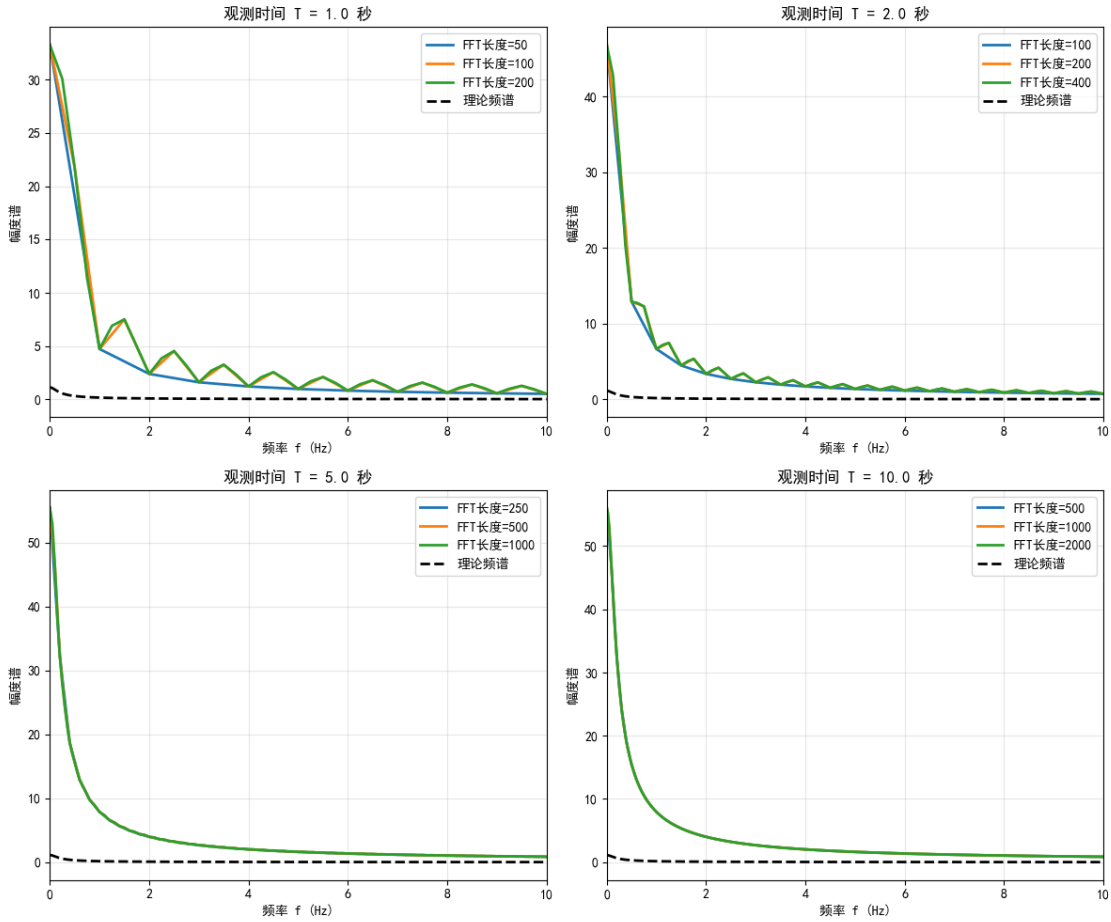

FFT长度 = 256
最大误差: 8.88e-16
时域参考
FFT长度=256
幅度
时间序号 n


采样频率 fs = 10 Hz
连续信号
采样点
幅度
时间 t (秒)


采样频率 fs = 20 Hz
连续信号
采样点
幅度
时间 t (秒)


采样频率 fs = 50 Hz
连续信号
采样点
幅度
时间 t (秒)


采样频率 fs = 100 Hz
连续信号
采样点
幅度
时间 t (秒)



结果分析

1. 指数序列 FFT 呈现典型的低通特性，IFFT 重建误差在$\mathbf{10^{-15}}$量级（浮点精度极限）；
2. FFT 实现的频域卷积与时域线性卷积结果几乎完全一致（误差<$\mathbf{10^{-14}}$）；
3. FFT 长度不足（L=32）时出现明显循环卷积失真；L≥64 后误差迅速降至机器精度；
4. 采样频率越高，采样点越密集，越能反映连续信号细节；
5. 观测时间越长，频谱分辨率越高；FFT 长度增加相当于频域插值，使谱线更平滑。

五、实验代码

程序代码：

```
import numpy as np

import matplotlib.pyplot as plt

from scipy import signal

# 设置中文字体

plt.rcParams['font.sans-serif'] = ['SimHei']

plt.rcParams['axes.unicode_minus'] = False

def exponential_sequence_fft_analysis():

    """指数序列的 FFT 和 IFFT 分析"""

    # 生成指数序列

    N = 64

    n = np.arange(N)

    alpha = 0.9

    x = alpha ** n

    # 计算 FFT

    X = np.fft.fft(x)

    freq = np.fft.fftfreq(N)

    # 计算 IFFT（验证可逆性）

    x_recon = np.fft.ifft(X)
```

```
    # 绘制结果
    plt.figure(figsize=(12, 10))
    # 原始序列
    plt.subplot(3, 2, 1)
    plt.stem(n, x, linefmt='C0-', markerfmt='C0o', basefmt='C0-')
    plt.title('原始指数序列 x[n] = 0.9ⁿ')
    plt.xlabel('时间序号 n')
    plt.ylabel('幅度')
    plt.grid(True, alpha=0.3)
    # FFT 幅频特性
    plt.subplot(3, 2, 2)
    plt.stem(freq[:N // 2], np.abs(X)[:N // 2], linefmt='C1-', markerfmt='C1o', basefmt='C1-')
    plt.title('FFT 幅频特性')
    plt.xlabel('归一化频率')
    plt.ylabel('|X[k]|')
    plt.grid(True, alpha=0.3)
    # FFT 相频特性
    plt.subplot(3, 2, 3)
```

```
    plt.stem(freq[:N // 2], np.angle(X)[:N // 2], linefmt='C2-', markerfmt='C2o', basefmt='C2-')

    plt.title('FFT 相频特性')

    plt.xlabel('归一化频率')

    plt.ylabel('∠X[k] (rad)')

    plt.grid(True, alpha=0.3)

    # IFFT 重建序列

    plt.subplot(3, 2, 4)

    plt.stem(n, np.real(x_recon), linefmt='C3-', markerfmt='C3o', basefmt='C3-')

    plt.title('IFFT 重建序列')

    plt.xlabel('时间序号 n')

    plt.ylabel('幅度')

    plt.grid(True, alpha=0.3)

    # 重建误差

    plt.subplot(3, 2, 5)

    error = np.abs(x - x_recon)

    plt.stem(n, error, linefmt='C4-', markerfmt='C4o', basefmt='C4-')

    plt.title('重建误差')

    plt.xlabel('时间序号 n')
```

```
    plt.ylabel('误差幅度')
    plt.grid(True, alpha=0.3)
    # 极坐标表示
    plt.subplot(3, 2, 6, projection='polar')
    plt.plot(np.angle(X)[:N // 2], np.abs(X)[:N // 2], 'C5-', linewidth=1)
    plt.title('频谱极坐标表示')
    plt.tight_layout()
    plt.show()
    print("FFT/IFFT 重建最大误差:", np.max(error))
    return x, X, x_recon
def system_response_fft_analysis():
    """利用 FFT 计算系统输出响应"""
    # 定义系统参数
    b = [1, 0.5]  # 系统分子系数
    a = [1, -1.5, 0.7]  # 系统分母系数
    # 生成输入序列（指数序列）
    N = 64
    n = np.arange(N)
    alpha = 0.8
```

```
x = alpha ** n

# 计算系统脉冲响应

_, h = signal.dimpulse((b, a, 1), n=N)

h = h[0].flatten()

# 方法 1：时域卷积

y_conv = np.convolve(x, h, mode='full')[:N]

# 方法 2：频域相乘（使用 FFT）

# 扩展序列长度以避免循环卷积

L = N + len(h) - 1

X_fft = np.fft.fft(x, L)

H_fft = np.fft.fft(h, L)

Y_fft = X_fft * H_fft

y_fft = np.fft.ifft(Y_fft)[:N]

# 绘制结果

plt.figure(figsize=(12, 8))

# 输入序列

plt.subplot(2, 2, 1)

plt.stem(n, x, linefmt='C0-', markerfmt='C0o', basefmt='C0-')

plt.title('输入序列  x[n]')
```

```
    plt.xlabel('时间序号 n')
    plt.ylabel('幅度')
    plt.grid(True, alpha=0.3)
    # 系统脉冲响应
    plt.subplot(2, 2, 2)
    plt.stem(n, h, linefmt='C1-', markerfmt='C1o', basefmt='C1-')
    plt.title('系统脉冲响应 h[n]')
    plt.xlabel('时间序号 n')
    plt.ylabel('幅度')
    plt.grid(True, alpha=0.3)
    # 输出响应比较
    plt.subplot(2, 2, 3)
    plt.stem(n, y_conv, linefmt='C2-', markerfmt='C2o', basefmt='C2-', label='时域卷积')
    plt.stem(n, np.real(y_fft), linefmt='C3--', markerfmt='C3x', basefmt='C3-', label='频域 FFT')
    plt.title('系统输出响应比较')
    plt.xlabel('时间序号 n')
    plt.ylabel('幅度')
```

```
    plt.legend()
    plt.grid(True, alpha=0.3)
    # 两种方法差异
    plt.subplot(2, 2, 4)
    error = np.abs(y_conv - np.real(y_fft))
    plt.stem(n, error, linefmt='C4-', markerfmt='C4o', basefmt='C4-')
    plt.title('时域与频域方法差异')
    plt.xlabel('时间序号 n')
    plt.ylabel('误差幅度')
    plt.grid(True, alpha=0.3)
    plt.tight_layout()
    plt.show()
    print("时域卷积与频域 FFT 方法最大差异:", np.max(error))
    return x, h, y_conv, y_fft
def fft_length_effect_analysis():
    """分析 FFT 计算长度对系统输出的影响"""
    # 系统参数
    b = [1, 0.5]
    a = [1, -1.5, 0.7]
```

```
# 输入序列

N = 32

n = np.arange(N)

x = 0.8 ** n

# 系统脉冲响应

_, h = signal.dimpulse((b, a, 1), n=N)

h = h[0].flatten()

# 不同的 FFT 长度

fft_lengths = [32, 64, 128, 256]

plt.figure(figsize=(12, 10))

for i, L in enumerate(fft_lengths):

    # 频域计算

    X_fft = np.fft.fft(x, L)

    H_fft = np.fft.fft(h, L)

    Y_fft = X_fft * H_fft

    y_fft = np.fft.ifft(Y_fft)[:N]

    # 时域卷积（参考）

    y_ref = np.convolve(x, h, mode='full')[:N]

    # 绘制结果
```

```
            plt.subplot(2, 2, i + 1)

            plt.stem(n, y_ref, linefmt='C0-', markerfmt='C0o', basefmt='C0-', label='时域参考')

            plt.stem(n, np.real(y_fft), linefmt='C1--', markerfmt='C1x', basefmt='C1-', label=f'FFT 长度={L}')

            plt.title(f'FFT 长度 = {L}')

            plt.xlabel('时间序号 n')

            plt.ylabel('幅度')

            plt.legend()

            plt.grid(True, alpha=0.3)

            # 计算误差

            error = np.max(np.abs(y_ref - np.real(y_fft)))

            plt.text(0.05, 0.95, f'最大误差: {error:.2e}',

                     transform=plt.gca().transAxes,

                     bbox=dict(boxstyle="round", facecolor="white", alpha=0.8))

    plt.tight_layout()

    plt.show()

    return fft_lengths

def continuous_signal_analysis():
```

```
"""连续时间信号的采样和频谱分析"""

# 连续时间指数信号: x(t) = e^(-αt), t >= 0

alpha = 0.9  # 与离散序列相同的衰减系数

# 信号参数

T_obs = 5.0  # 观测时间长度 (秒)

# 不同的采样频率

fs_list = [10, 20, 50, 100]  # Hz

plt.figure(figsize=(12, 10))

for i, fs in enumerate(fs_list):

    # 采样参数

    Ts = 1.0 / fs  # 采样间隔

    N = int(T_obs * fs)  # 采样点数

    # 采样时间点

    t_continuous = np.linspace(0, T_obs, 1000)  # 连续时间（用于显示）

    t_sampled = np.arange(0, T_obs, Ts)  # 采样时间点

    # 连续信号和采样信号

    x_continuous = np.exp(-alpha * t_continuous)

    x_sampled = np.exp(-alpha * t_sampled)

    # 计算 FFT
```

```
            X_fft = np.fft.fft(x_sampled)

            freq = np.fft.fftfreq(len(x_sampled), Ts)

            # 绘制结果

            plt.subplot(2, 2, i + 1)

            # 连续信号和采样点

            plt.plot(t_continuous, x_continuous, 'C0-', linewidth=1, alpha=0.7, label='连续信号')

            plt.stem(t_sampled, x_sampled, linefmt='C1-', markerfmt='C1o', basefmt='C1-', label='采样点')

            plt.title(f'采样频率  fs = {fs} Hz')

            plt.xlabel('时间  t (秒)')

            plt.ylabel('幅度')

            plt.legend()

            plt.grid(True, alpha=0.3)

        plt.tight_layout()

        plt.show()

        return fs_list

def observation_time_fft_length_effect():

    """分析观测时间和 FFT 长度对频谱的影响"""
```

```
# 连续指数信号参数

alpha = 0.9

# 固定采样频率

fs = 50    # Hz

Ts = 1.0 / fs

# 不同的观测时间

T_obs_list = [1.0, 2.0, 5.0, 10.0]    # 秒

plt.figure(figsize=(12, 10))

for i, T_obs in enumerate(T_obs_list):

    # 采样

    N_original = int(T_obs * fs)

    t_sampled = np.arange(0, T_obs, Ts)

    x_sampled = np.exp(-alpha * t_sampled)

    # 不同的 FFT 长度

    fft_lengths = [N_original, 2 * N_original, 4 * N_original]

    plt.subplot(2, 2, i + 1)

    for L in fft_lengths:

        # 计算 FFT

        X_fft = np.fft.fft(x_sampled, L)
```

```
            freq = np.fft.fftfreq(L, Ts)
            # 绘制幅频特性（只显示正频率）
            positive_freq = freq[:L // 2]
            positive_spectrum = np.abs(X_fft)[:L // 2]
            plt.plot(positive_freq, positive_spectrum,
                     label=f'FFT 长度={L}', linewidth=2)
        # 理论频谱（连续时间傅里叶变换）
        f_theory = np.linspace(0, fs / 2, 1000)
        X_theory = 1.0 / np.sqrt(alpha ** 2 + (2 * np.pi * f_theory) ** 2)
        plt.plot(f_theory, X_theory, 'k--', linewidth=2, label='理论频谱')
        plt.title(f'观测时间 T = {T_obs} 秒')
        plt.xlabel('频率 f (Hz)')
        plt.ylabel('幅度谱')
        plt.legend()
        plt.grid(True, alpha=0.3)
        plt.xlim(0, 10)  # 限制频率范围以便观察
plt.tight_layout()
plt.show()
return T_obs_list
```

```
def main():
    """主函数"""
    print("=" * 60)
    print("实验四: 信号的频谱分析")
    print("=" * 60)
    # 1. 指数序列的 FFT 和 IFFT 分析
    print("\n1. 指数序列的 FFT 和 IFFT 分析")
    x, X, x_recon = exponential_sequence_fft_analysis()
    # 2. 利用 FFT 计算系统输出响应
    print("\n2. 利用 FFT 计算系统输出响应")
    x_input, h_system, y_conv, y_fft = system_response_fft_analysis()
    # 3. FFT 计算长度对系统输出的影响
    print("\n3. FFT 计算长度对系统输出的影响分析")
    fft_lengths = fft_length_effect_analysis()
    # 4. 连续时间信号的采样和频谱分析
    print("\n4. 连续时间信号的采样和频谱分析")
    fs_list = continuous_signal_analysis()
    # 5. 观测时间和 FFT 长度对频谱的影响
    print("\n5. 观测时间和 FFT 长度对频谱的影响分析")
```

```
    T_obs_list = observation_time_fft_length_effect()

if __name__ == "__main__":

    main()
```

六、实验问题及解决方案

	实验问题	解决方案
XXX	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；	通过在不同采样频率下对指数信号采样并计算 FFT，同时分析观测时间和 FFT 长度对频谱分辨率和精度的影响。
XXX	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	生成指数序列并计算其 FFT 和 IFFT，绘制幅频特性、相频特性及重建误差曲线来验证变换的可逆性。
XXX	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	通过时域卷积与频域 FFT 相乘两种方法计算系统输出，分析不同 FFT 长度对输出响应计算精度的影响。