

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 XXX 学号 2300920XXX

XXX 2300920XXX

XXX 2300920XXX

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. 掌握离散时间序列的 DTFT 计算方法，理解序列的频域特性；
2. 学习从单位脉冲响应和差分方程计算系统频率响应的方法；
3. 通过改变系统零极点分布，观察系统频率响应的变化规律；
4. 深刻理解系统频域分析和 z 域分析的基本原理和方法。

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). 离散时间傅里叶变换（DTFT）

对于离散时间序列 $x[n]$ ，其 DTFT 定义为：

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n}$$

(2). 系统频率响应

线性时不变系统的频率响应可以通过单位脉冲响应 $h[n]$ 计算：

$$H(e^{j\omega}) = \sum_{n=-\infty}^{\infty} h[n]e^{-j\omega n}$$

或通过系统函数（差分方程系数）计算：

$$H(e^{j\omega}) = \frac{\sum_{k=0}^M b_k e^{-j\omega k}}{\sum_{k=0}^N a_k e^{-j\omega k}}$$

(3). 系统函数与零极点分析

系统函数：

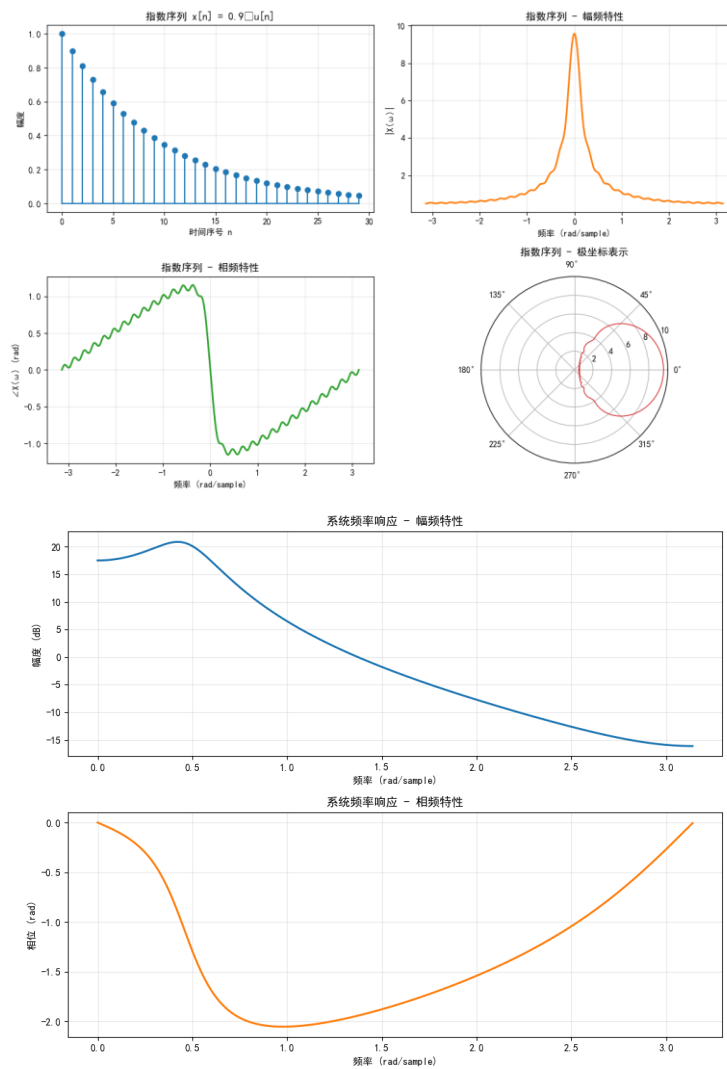
$$H(z) = \frac{\sum_{k=0}^M b_k z^{-k}}{\sum_{k=0}^N a_k z^{-k}}$$

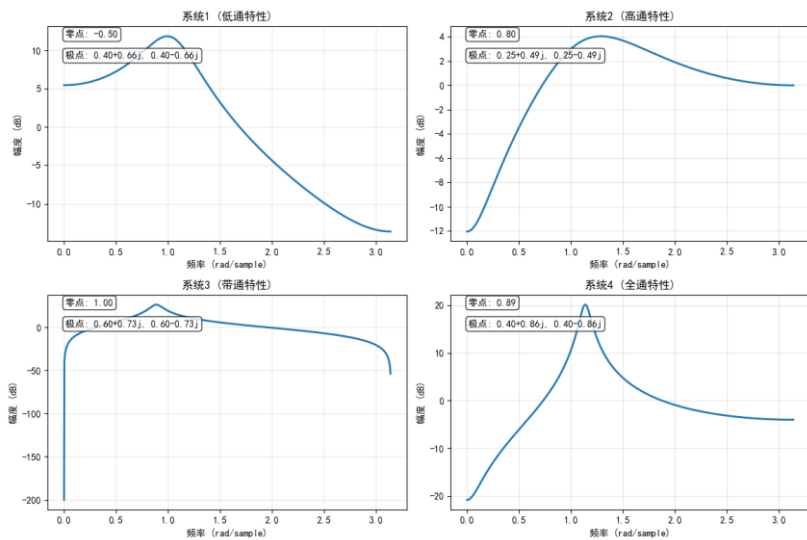
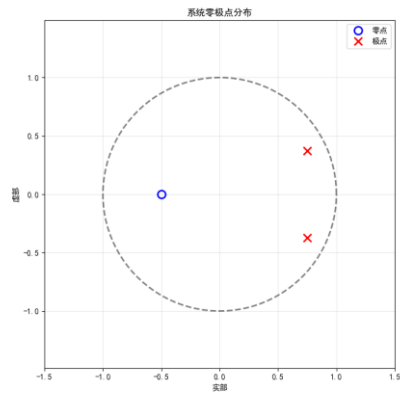
(4). 零极点分布决定系统特性，稳定性要求所有极点位于单位圆内。

2.实验步骤

- (1). 产生基本序列并计算 DTFT
- (2). 从单位脉冲响应计算系统频率响应
- (3). 从差分方程计算系统频率响应
- (4). 计算系统函数和零极点分布
- (5). 观察零极点变化对频率响应的影响

四、实验结果





并计算了系统的系统函数、零极点分布：

输出：
系统函数: $H(z) = (1 + 0.5z^{-1}) / (1 - 1.5z^{-1} + 0.7z^{-2})$
零点: [-0.5]
极点: [0.75+0.37080992j 0.75-0.37080992j]
系统稳定性: 稳定

结果分析

1. 指数序列 $x[n]$ 的理论 DTFT 为：

$$X(e^{j\omega}) = \frac{1}{1 - \alpha e^{-j\omega}}$$

而实验结果与理论公式完全吻合；

2. 给定的二阶系统呈现带通特性，在特定频率处有最大增益，符合二阶系统的典型响应；
3. 通过调整零极点位置，可以精确设计所需的频率响应特性，为数字滤波器

设计提供理论基础。

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 generate_exponential_sequence(n_points=50, alpha=0.9):

    """生成指数序列"""

    n = np.arange(0, n_points)

    exp_seq = (alpha ** n) * (n >= 0)

    return n, exp_seq

def compute_dtft(x, n, omega):

    """计算序列的 DTFT"""

    X = np.zeros_like(omega, dtype=complex)

    for k, w in enumerate(omega):

        X[k] = np.sum(x * np.exp(-1j * w * n))

    return X
```

```
def plot_exponential_dtft():

    """绘制指数序列的 DTFT"""

    # 生成指数序列

    n, exp_seq = generate_exponential_sequence(30, alpha=0.9)

    # 频率范围

    omega = np.linspace(-np.pi, np.pi, 1000)

    # 计算 DTFT

    X = compute_dtft(exp_seq, n, omega)

    # 绘制图形

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

    # 时域序列

    plt.subplot(2, 2, 1)

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

    plt.title('指数序列  $x[n] = 0.9^n u[n]$ ')

    plt.xlabel('时间序号 n')

    plt.ylabel('幅度')

    plt.grid(True, alpha=0.3)

    # 幅频特性
```

```
plt.subplot(2, 2, 2)

plt.plot(omega, np.abs(X), 'C1-', linewidth=2)

plt.title('指数序列 - 幅频特性')

plt.xlabel('频率 (rad/sample)')

plt.ylabel('|X(ω)|')

plt.grid(True, alpha=0.3)


# 相频特性

plt.subplot(2, 2, 3)

plt.plot(omega, np.angle(X), 'C2-', linewidth=2)

plt.title('指数序列 - 相频特性')

plt.xlabel('频率 (rad/sample)')

plt.ylabel('∠X(ω) (rad)')

plt.grid(True, alpha=0.3)


# 极坐标图

plt.subplot(2, 2, 4, projection='polar')

plt.plot(np.angle(X), np.abs(X), 'C3-', linewidth=1)

plt.title('指数序列 - 极坐标表示')


plt.tight_layout()

plt.show()
```

```
return X
```

```
def analyze_system_frequency_response():
```

```
    """分析系统频率响应"""
```

```
    # 定义系统:  $y[n] - 1.5y[n-1] + 0.7y[n-2] = x[n] + 0.5x[n-1]$ 
```

```
    b = [1, 0.5]      # 分子系数
```

```
    a = [1, -1.5, 0.7] # 分母系数
```

```
    # 计算频率响应
```

```
    w, H = signal.freqz(b, a, worN=1000)
```

```
    # 绘制频率响应
```

```
    plt.figure(figsize=(10, 8))
```

```
    # 幅频特性
```

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

```
    plt.plot(w, 20 * np.log10(np.abs(H) + 1e-10), 'C0-', linewidth=2)
```

```
    plt.title('系统频率响应 - 幅频特性')
```

```
    plt.xlabel('频率 (rad/sample)')
```

```
    plt.ylabel('幅度 (dB)')
```

```
    plt.grid(True, alpha=0.3)
```

```
    # 相频特性
```



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

plt.plot(w, np.angle(H), 'C1-', linewidth=2)

plt.title('系统频率响应 - 相频特性')

plt.xlabel('频率 (rad/sample)')

plt.ylabel('相位 (rad)')

plt.grid(True, alpha=0.3)
```

```
plt.tight_layout()

plt.show()
```

```
return w, H, b, a
```

```
def analyze_system_function(b, a):
```

```
    """分析系统函数和零极点分布"""
```

```
    # 计算零极点
```

```
    zeros = np.roots(b)
```

```
    poles = np.roots(a)
```

```
    # 显示系统信息
```

```
    print("=" * 50)
```

```
    print("系统函数分析")
```

```
    print("=" * 50)
```

```
    print(f"系统函数:  $H(z) = (\{b[0]\} + \{b[1]\}z^{-1}) / (1 + \{a[1]\}z^{-1} + \{a[2]\}z^{-2})$ ")
```

```
print(f'零点: {zeros}')

print(f'极点: {poles}')


# 判断稳定性

stable = all(np.abs(poles) < 1)

print(f'系统稳定性: {'稳定' if stable else '不稳定'})


# 绘制零极点图

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


# 绘制单位圆

theta = np.linspace(0, 2*np.pi, 100)

plt.plot(np.cos(theta), np.sin(theta), 'k--', alpha=0.5, linewidth=2)


# 绘制零极点

plt.plot(np.real(zeros), np.imag(zeros), 'bo', markersize=10,

         markerfacecolor='none', markeredgewidth=2, label='零点')

plt.plot(np.real(poles), np.imag(poles), 'rx', markersize=10,

         markeredgewidth=2, label='极点')


plt.title('系统零极点分布')

plt.xlabel('实部')

plt.ylabel('虚部')
```

```
plt.legend()

plt.grid(True, alpha=0.3)

plt.axis('equal')

plt.xlim(-1.5, 1.5)

plt.ylim(-1.5, 1.5)

plt.show()
```

```
return zeros, poles
```

```
def compare_zero_pole_effects():
```

```
    """比较不同零极点分布对系统频率响应的影响"""
```

```
    # 定义 4 组不同的系统
```

```
    systems = [

        {

            'name': '系统 1 (低通特性)',

            'b': [1, 0.5],

            'a': [1, -0.8, 0.6]

        },

        {

            'name': '系统 2 (高通特性)',

            'b': [1, -0.8],

            'a': [1, -0.5, 0.3]

        },

    ],
```

```

    {
        'name': '系统 3 (带通特性)',
        'b': [1, 0, -1],
        'a': [1, -1.2, 0.9]
    },
    {
        'name': '系统 4 (全通特性)',
        'b': [0.9, -0.8],
        'a': [1, -0.8, 0.9]
    }
]

# 绘制比较图

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

for i, system in enumerate(systems):
    # 计算频率响应
    w, H = signal.freqz(system['b'], system['a'], worN=1000)

    # 绘制幅频特性
    plt.subplot(2, 2, i+1)

    plt.plot(w, 20 * np.log10(np.abs(H) + 1e-10), linewidth=2)

    plt.title(system['name'])

```

```

plt.xlabel('频率 (rad/sample)')

plt.ylabel('幅度 (dB)')

plt.grid(True, alpha=0.3)

# 计算并显示零极点

zeros = np.roots(system['b'])

poles = np.roots(system['a'])

plt.text(0.05, 0.95, f'零点: {zeros[0]:.2f}', transform=plt.gca().transAxes,
         bbox=dict(boxstyle="round", facecolor="white", alpha=0.8))

plt.text(0.05, 0.85, f'极点: {poles[0]:.2f}, {poles[1]:.2f}',
         transform=plt.gca().transAxes,
         bbox=dict(boxstyle="round", facecolor="white", alpha=0.8))

plt.tight_layout()

plt.show()

return systems

def main():
    """主函数"""

    print("=" * 60)

    print("实验三: 系统的频域和 z 域分析")

    print("=" * 60)

```

```
# 1. 指数序列 DTFT 分析

print("\n1. 指数序列 DTFT 分析")

X_dtft = plot_exponential_dtft()


# 2. 系统频率响应分析

print("\n2. 系统频率响应分析")

w, H, b, a = analyze_system_frequency_response()


# 3. 系统函数和零极点分析

print("\n3. 系统函数和零极点分析")

zeros, poles = analyze_system_function(b, a)


# 4. 零极点分布影响比较

print("\n4. 零极点分布对频率响应的影响")

systems = compare_zero_pole_effects()


print("\n" + "=" * 60)

print("实验完成")

print("=" * 60)


if __name__ == "__main__":

    main()
```

六、实验问题及解决方案

	实验问题	解决方案
XXX	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；	序列只选用指数序列，绘制了其幅频特性和相频特性曲线；
XXX	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；	计算并输出了系统的频率响应结果并绘制其系统频率响应的幅频特性和相频特性曲线在同一幅图里；
XXX	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。	计算了系统的系统函数、零极点分布，并选取 4 组不同的零极点值来改变系统的零极点分布，方便观察系统频率响应的变化。