**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](#_Toc148806958)

[绪论 4](#_Toc148806959)

[1. 发展历史 4](#_Toc148806960)

[2. 现状 5](#_Toc148806961)

[3. 研究意义 5](#_Toc148806962)

[正文 5](#_Toc148806963)

[实验目的 5](#_Toc148806964)

[实验原理与方法 6](#_Toc148806965)

[实验原理 6](#_Toc148806966)

[实验方法 6](#_Toc148806967)

[实验内容及步骤 7](#_Toc148806968)

[实验内容 7](#_Toc148806969)

[实验步骤 8](#_Toc148806970)

[实验结果分析及结论总结 8](#_Toc148806971)

[第一部分 8](#_Toc148806972)

[第二部分 9](#_Toc148806973)

[第三部分 10](#_Toc148806974)

[心得及展望 11](#_Toc148806975)

[心得 11](#_Toc148806976)

[展望 12](#_Toc148806977)

[附件 12](#_Toc148806978)

# **摘要**

这是一个关于数字信号处理的实验报告，主要包括四个实验部分，分别涵盖了频域分析、信号的频谱分析、IIR数字滤波器设计和FIR数字滤波器设计。在每个实验部分，你进行了具体的实验操作，使用编程工具生成信号、计算频谱、分析系统的频率响应等内容。以下是一些具体的观察和总结：

- 在第一部分实验中，你学习了如何生成序列并计算其DTFT，绘制了幅频特性和相频特性曲线。这有助于理解信号在频域中的特性。

- 第二部分实验中，你计算了系统的频率响应，使用了传递函数的系数，然后绘制了系统的幅频特性和相频特性曲线。这对于理解系统的滤波特性和稳定性非常重要。

- 在第三部分，你修改了系统的零极点分布，观察了系统频率响应的变化。这展示了零极点分布对系统行为的影响。

总的来说，这些实验帮助理解数字信号处理的基本概念和技术，并培养了实际应用数字信号处理知识的能力。

# **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

1.理论理解：实验可以帮助学生理论上理解数字信号处理中的频域分析和z域分析的基本概念。通过实际操作和计算，学生可以深入了解如何应用数学和信号处理理论来分析和理解数字系统的行为。

2.实际应用：数字信号处理是许多工程领域的关键技术，包括通信、音频处理、图像处理、控制系统等。实验帮助学生将理论知识应用到实际问题中，培养他们解决实际工程问题的能力。

3.系统分析：实验的一个重要目的是让学生能够分析数字系统的频率响应、稳定性和滤波特性。这有助于他们理解和设计数字系统，以满足特定的工程需求。

4.数据处理和可视化技能：通过实验，学生将学会如何使用计算机程序来生成、处理和可视化数字信号，这是数字信号处理工程师和科学家的重要技能。

5.零极点分析：实验中的零极点分布分析有助于学生理解数字系统的稳定性和频率响应特性，以及如何改变这些特性。

## 实验原理与方法

### 实验原理

1. DTFT（离散时间傅立叶变换）：\*\* DTFT用于将一个离散信号转换为频域信号。DTFT可以通过以下公式计算：

这里，表示信号的频域表示，是输入信号。

2. 单位脉冲响应和差分方程：通过单位脉冲响应和差分方程，你可以描述一个系统的行为。差分方程通常是系统的离散时间表示。

3. 系统函数、零点和极点：系统函数是差分方程的z变换，它描述了系统的频率响应。系统的零点和极点对频率响应和系统的稳定性有重要影响。

### 实验方法

1. 生成序列和计算DTFT：

- 选择一个序列，如单位脉冲响应或其他输入信号。

- 使用计算机编程工具（如MATLAB、Python等）编写代码来计算DTFT。通常可以使用FFT等算法来进行数值计算。

2. 绘制幅频特性和相频特性曲线：

- 使用计算得到的DTFT结果，绘制幅频特性和相频特性曲线。这可以通过绘制频率与幅度或相位的关系图来完成。

3. 计算系统的频率响应：

- 基于系统的单位脉冲响应和差分方程，计算系统的频率响应。你可以使用z变换或差分方程的离散时间傅立叶变换。

4. 绘制系统频率响应的幅频特性和相频特性曲线：

- 使用计算得到的系统频率响应，绘制幅频特性和相频特性曲线，类似于DTFT的绘制过程。

5. 计算系统函数、零极点分布：

- 使用z变换或差分方程，计算系统的传递函数或系统函数，然后确定零点和极点的位置。

6. 改变系统的零极点分布：

- 在计算得到系统函数后，可以通过改变零点和极点的位置，再次计算系统的频率响应，以观察系统响应如何随零点和极点的变化而变化。

7. 总结实验结果：

- 总结你的实验结果，包括观察到的频率响应变化、零极点对系统响应的影响，以及系统的稳定性和滤波特性等。

## 实验内容及步骤

### 实验内容

1. 生成序列并计算序列的DTFT：

- 选择一个离散序列，如单位脉冲响应或其他输入信号。

- 使用计算机编程工具（如MATLAB、Python等）编写程序来计算序列的离散时间傅立叶变换（DTFT）。

- 绘制生成序列的幅频特性曲线和相频特性曲线。

2. 计算系统的频率响应：

- 提供系统的单位脉冲响应（通常是差分方程的响应）或系统的传递函数。

- 使用z变换或差分方程的离散时间傅立叶变换，计算系统的频率响应。

- 绘制系统频率响应的幅频特性曲线和相频特性曲线。

3. 计算系统函数、零极点分布：

- 使用z变换或差分方程，计算系统的传递函数或系统函数。

- 确定系统的零点和极点的位置。

4. 改变系统的零极点分布：

- 修改系统的零点和极点的位置，可以增加、减少或移动它们。

- 重新计算系统的频率响应，并绘制新的幅频特性和相频特性曲线。

### 实验步骤

1. 使用编程工具（如MATLAB或Python）编写程序来生成你选择的序列（例如单位脉冲响应），并计算其DTFT。绘制幅频特性和相频特性曲线。

2. 提供系统的单位脉冲响应或差分方程。使用相同的编程工具计算系统的频率响应，然后绘制幅频特性和相频特性曲线。

3. 使用系统的单位脉冲响应或差分方程，计算系统函数和确定零点和极点的位置。

4. 修改系统的零点和极点的位置，然后重新计算系统的频率响应。比较原始系统和修改后系统的频率响应。

5. 撰写实验报告，包括实验的目的、方法、结果和结论。在报告中附上生成的图表和曲线。

## 实验结果分析及结论总结

### 第一部分

本次实验使用python代码编写，创建DTFT类进行变换得到结果如下：

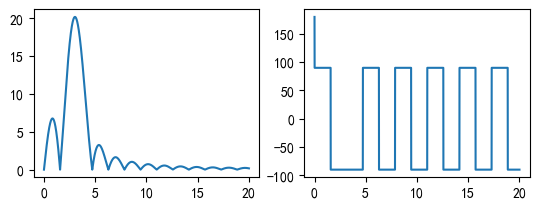


图1 sin(nπ)信号幅度谱和相位谱

|  |
| --- |
| 原始函数是sin(nπ) 定义域是0-2以0.1为间隔。  以下是关键的函数部分其余代码附在附录中 |
| *def* xjw(*self*,*fre*:list):          '''          :param fre: 频率自变量fre序列          :return:          '''          # 实现yvalues为X（jw）频谱值          ## 遍历频率自变量          for f in *fre*:              p = 0              # 遍历时间序列，得到累加和              for x in *self*.xvalues:                  p = np.sin(x\*np.pi)\*math.e\*\*(-1*j*\*f\*x) + p  *self*.yvalues.append(p) # 对应每一个自变量频率的因变量值 |

### 第二部分

|  |
| --- |
| # 系统的传递函数系数  b = [1, 0.5]  # 输出系数  a = [1, -0.7]  # 输入系数  计算出频率响应函数绘制曲线如下： |

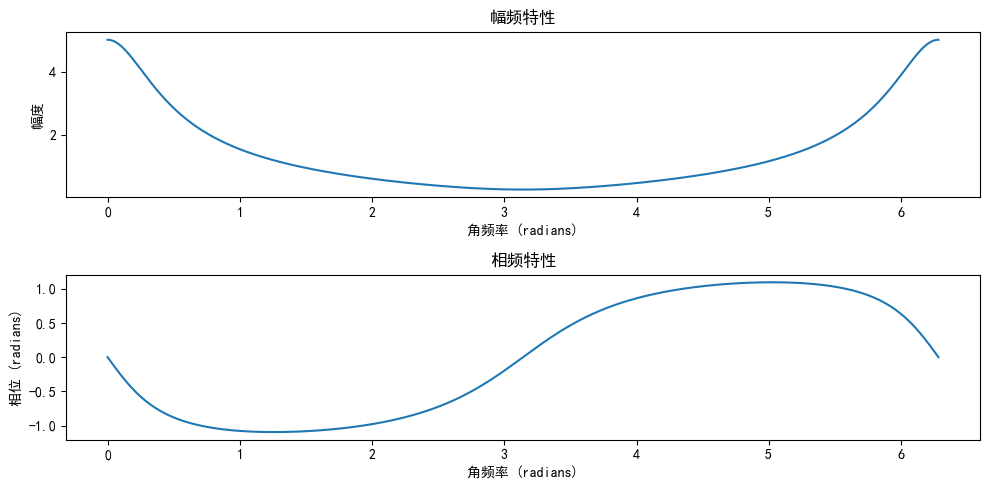


图2 第二部分绘图

|  |
| --- |
| 关键代码： |
| # 计算频率响应  w = np.linspace(0, 2 \* np.pi, 1000)  # 角频率范围  H = np.polyval(b, np.exp(1*j* \* w)) / np.polyval(a, np.exp(1*j* \* w))  # 计算传递函数在不同频率下的值 |

### 第三部分

# 原始的传递函数系数

b\_original = [1, 0, 0]  # 原始零点在原点

a\_original = [1, -0.7]

# 新的传递函数系数

b\_new = [1, -0.5, 0.25]  # 新的零点分布

a\_new = [1, -0.7]

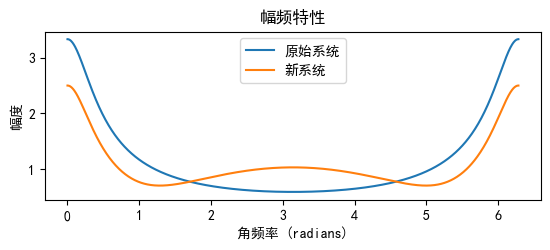


图3 变化前后幅频特性变化

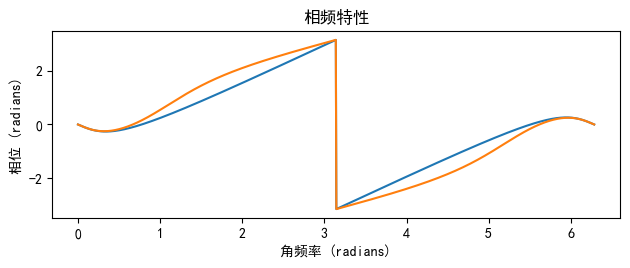


图4 变化前后相频特性变化

代码基本同上

## 心得及展望

### 心得

通过完成实验1，我学到了很多数字信号处理的基本概念和技术。以下是我在实验中的一些心得体会：

1.理解频域分析的重要性：频域分析允许我们理解信号和系统在不同频率下的行为。通过绘制幅频特性和相频特性曲线，我更好地了解了信号如何在不同频率下响应。

2.系统性能分析：实验中，我们学习了如何分析系统的频率响应和性能。这对于设计和优化数字系统非常关键。

3.零极点分析的重要性：零极点分析揭示了系统的稳定性、滤波特性和频率响应。了解零点和极点如何影响系统有助于更好地理解系统行为。

4.实际编程应用：通过编写计算机程序来计算DTFT、系统频率响应和分析零极点分布，我获得了实际应用数字信号处理知识的机会。

### 展望

在未来，我希望能够进一步探索数字信号处理领域，包括以下方面：

1.深入研究滤波器设计：我计划深入研究数字滤波器设计和优化，以更好地理解不同类型滤波器的应用。

2.实际工程应用：我希望将数字信号处理的知识应用到实际工程项目中，例如通信、音频处理或图像处理领域。

3.继续学习：数字信号处理是一个不断发展的领域，我计划继续学习新技术和工具，以保持与行业最新趋势的接轨。

4.研究和项目：我有兴趣参与研究项目，特别是与数字信号处理和机器学习相关的项目，以进一步拓展我的知识和技能。

## 附件

|  |
| --- |
| 代码1 |
| import math  import numpy as np  # 三角函数sin(nπ)的DTFT    class dtft():  def \_\_init\_\_(self,xvalues:list):  '''  :param xvalues: 时域自变量t序列  '''  self.yvalues = []  self.xvalues = xvalues    def xjw(self,fre:list):  '''  :param fre: 频率自变量fre序列  :return:  '''  # 实现yvalues为X（jw）频谱值  ## 遍历频率自变量  for f in fre:  p = 0  # 遍历时间序列，得到累加和  for x in self.xvalues:  p = np.sin(x\*np.pi)\*math.e\*\*(-1j\*f\*x) + p  self.yvalues.append(p) # 对应每一个自变量频率的因变量值    if \_\_name\_\_ == '\_\_main\_\_':  import numpy as np  from matplotlib import pyplot as plt  xvalues = np.arange(-2,2,0.1)  ###print(xvalues)  y1 = [np.sin(i\*np.pi) for i in xvalues]  #print(np.sin(-1\*np.pi))  # 频谱  DTFT = dtft(xvalues)  freOmega = np.arange(0,20,0.001)  DTFT.xjw(freOmega)  # 幅度谱  plt.subplot(223)  plt.plot(freOmega,np.abs(DTFT.yvalues))  # 相位谱  plt.subplot(224)  angle\_ = np.angle(DTFT.yvalues)/np.pi  plt.plot(freOmega,180\*angle\_)  plt.show() |

|  |
| --- |
| 代码2 |
| import numpy as np  import matplotlib.pyplot as plt  # 系统的传递函数系数  b = [1, 0.5]  # 输出系数  a = [1, -0.7]  # 输入系数  # 计算频率响应  w = np.linspace(0, 2 \* np.pi, 1000)  # 角频率范围  H = np.polyval(b, np.exp(1*j* \* w)) / np.polyval(a, np.exp(1*j* \* w))  # 计算传递函数在不同频率下的值  # 绘制幅频特性曲线  plt.figure(*figsize*=(10, 5))  plt.subplot(2, 1, 1)  plt.plot(w, np.abs(H))  plt.title('幅频特性')  plt.xlabel('角频率 (radians)')  plt.ylabel('幅度')  # 绘制相频特性曲线  plt.subplot(2, 1, 2)  plt.plot(w, np.angle(H))  plt.title('相频特性')  plt.xlabel('角频率 (radians)')  plt.ylabel('相位 (radians)')  plt.tight\_layout()  plt.show() |

|  |
| --- |
| 代码3 |
| import numpy as np  import matplotlib.pyplot as plt  # 原始的传递函数系数  b\_original = [1, 0, 0]  # 原始零点在原点  a\_original = [1, -0.7]  # 新的传递函数系数  b\_new = [1, -0.5, 0.25]  # 新的零点分布  a\_new = [1, -0.7]  # 角频率范围  w = np.linspace(0, 2 \* np.pi, 1000)  # 计算原始系统的频率响应  H\_original = np.polyval(b\_original, np.exp(1*j* \* w)) / np.polyval(a\_original, np.exp(1*j* \* w))  # 计算新系统的频率响应  H\_new = np.polyval(b\_new, np.exp(1*j* \* w)) / np.polyval(a\_new, np.exp(1*j* \* w))  # 绘制幅频特性曲线  plt.subplot(2, 1, 1)  plt.plot(w, np.abs(H\_original), *label*='原始系统')  plt.plot(w, np.abs(H\_new), *label*='新系统')  plt.legend()  plt.title('幅频特性')  plt.xlabel('角频率 (radians)')  plt.ylabel('幅度')  plt.show()  plt.subplot(2, 1, 2)  plt.plot(w, np.angle(H\_original), *label*='原始系统')  plt.plot(w, np.angle(H\_new), *label*='新系统')  plt.title('相频特性')  plt.xlabel('角频率 (radians)')  plt.ylabel('相位 (radians)')  plt.tight\_layout()  plt.show() |