实验1

学号	姓名	分工
21307035	邓栩瀛	完成实验任务3及对应实验报告的撰写
21307056	钟欣余	验证程序experiment_1_1.py及对应实验报告的撰写
21307037	金思琪	验证程序experiment_1_2.py及对应实验报告的撰写
21307062	郑越	完成实验任务2及对应实验报告的撰写

1. 问题描述

1. 验证程序实例中的相关程序。
2. 利用$x(t)=u(t)-u(t-2)+u(t-0.5)-u(t-1.5)$，$(-3\le t\le 3)$, 编写相关程序，绘制出$x(-2t)$，$x(t/2+1)$和$5x(t)$的波形。
3. 设

$$x[n]=\begin{cases}-1 & n<-2\\ n & -2\le n\le 1\\ \frac{1}{n} & n>1\end{cases} \tag{1}$$

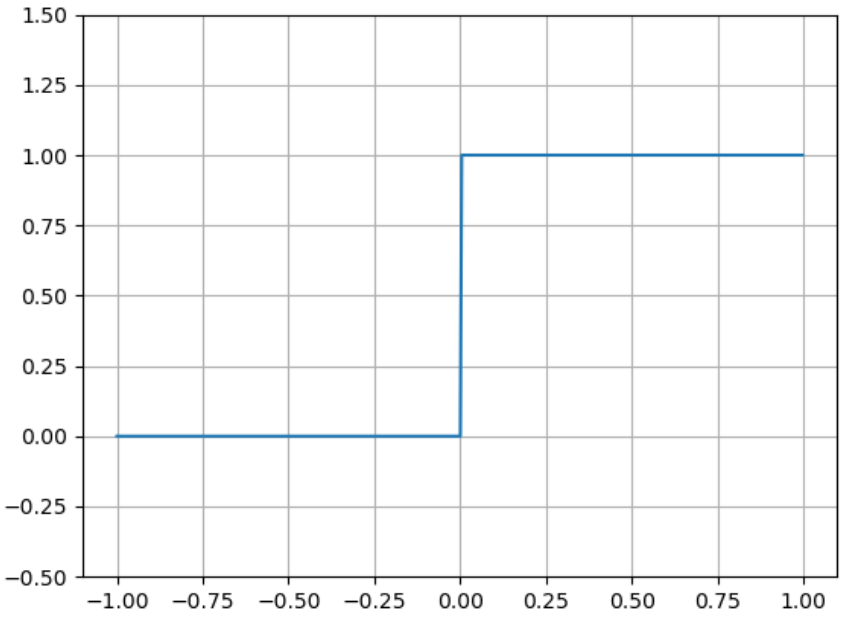
编写程序，绘制$x[-n]$, $x[2n+2]$, $x[n/2]$，$(-20\le n\le 20)$

2. 问题分析

实验原理

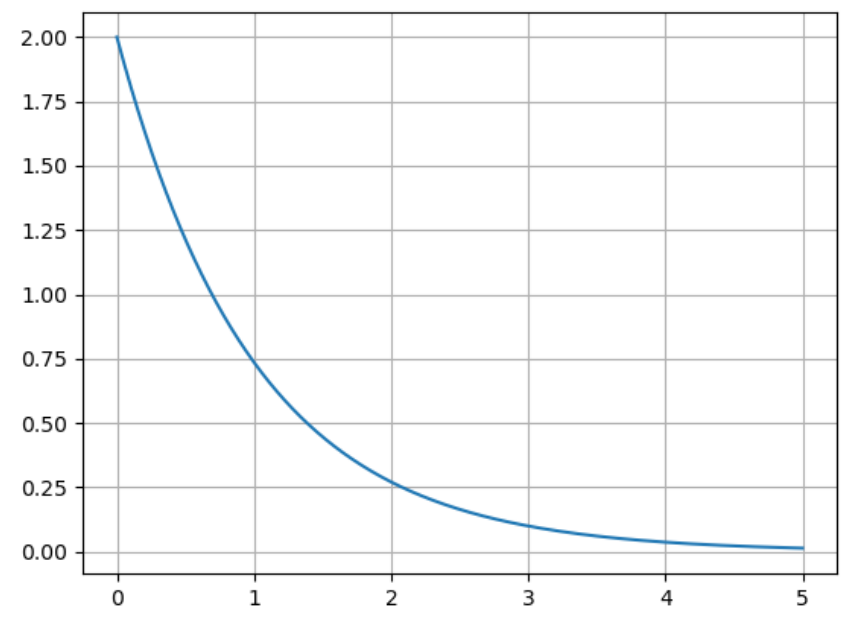
1. 连续阶跃信号的产生

```
def continuous_step_signal():
    t = np.linspace(-1, 1, 500, endpoint=False)
    plt.plot(t, np.array(t > 0, dtype=np.int))
    plt.grid()
    _ = plt.ylim(-0.5, 1.5)  # 限制 y 轴范围
    plt.show()
```



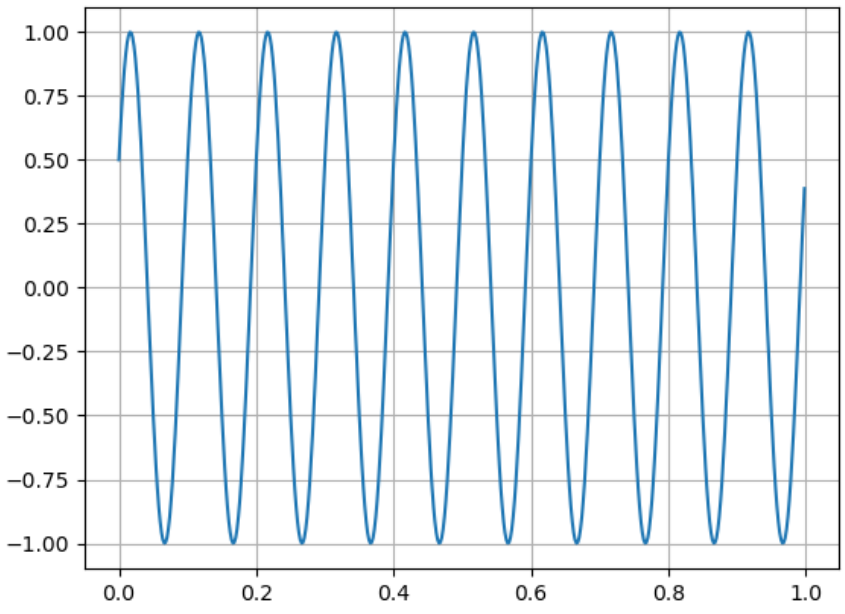
2. 连续指数信号的产生

```
def continuous_exp_signal():
    t = np.linspace(0, 5, 5000, endpoint=False)
    plt.plot(t, 2 * np.exp(-1 * t))
    plt.grid()
    plt.show()
```



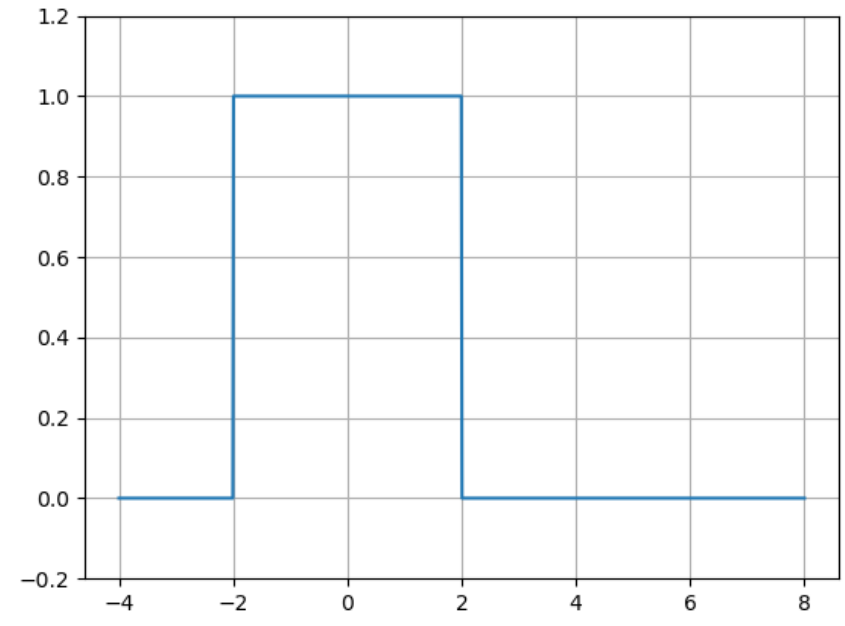
3. 连续正弦信号的产生

```
def continuous_sin_signal():
    t = np.linspace(0, 1, 500, endpoint=False)
    A = 1  # 振幅
    phi = np.pi / 6  # 初相位
    fre = 10  # 频率
    plt.plot(t, A * np.sin(t * 2 * np.pi * fre + phi))
    plt.grid()
    plt.show()
```



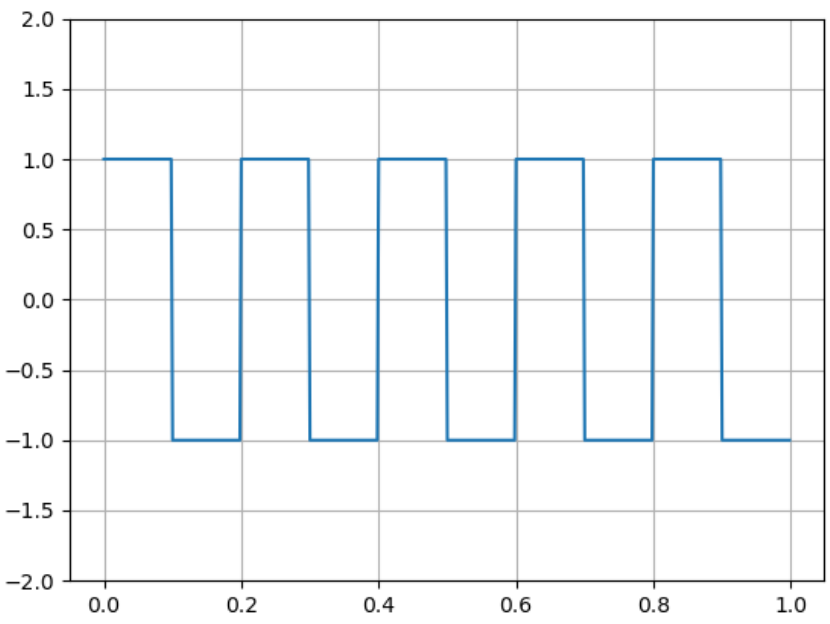
4. 连续矩形脉冲信号的产生

```
def rect_wave(t, c, c0):  # 起点为c0，宽度为c的矩形波
    if t >= (c + c0):
        r = 0.0
    elif t < c0:
        r = 0.0
    else:
        r = 1
    return r

def continuous_rectangular_pulse_signal():
    x = np.linspace(-4, 8, 1000)
    y = np.array([rect_wave(t, 4.0, -2.0) for t in x])
    _ = plt.ylim(-0.2, 1.2)
    plt.plot(x, y)
    plt.grid()
    plt.show()
```



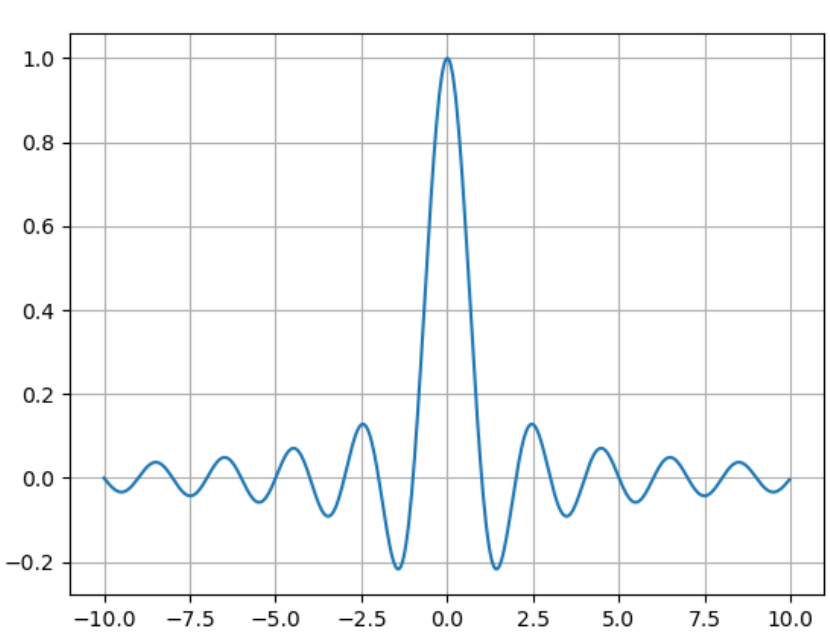
5. 连续周期矩形波信号的产生

```
def continuous_rectangle_signal():
    t = np.linspace(0, 1, 500, endpoint=False)
    '''
    sg.square(w0*t,duty=0.5)
    频率为w0，占空比默认0.5
    '''
    # 产生一个振幅为1，频率为5Hz，占空比为50%的周期方波
    plt.plot(t, sg.square(2 * np.pi * 5 * t))
    plt.ylim(-2, 2)  # y的范围
    plt.grid()
    plt.show()
```



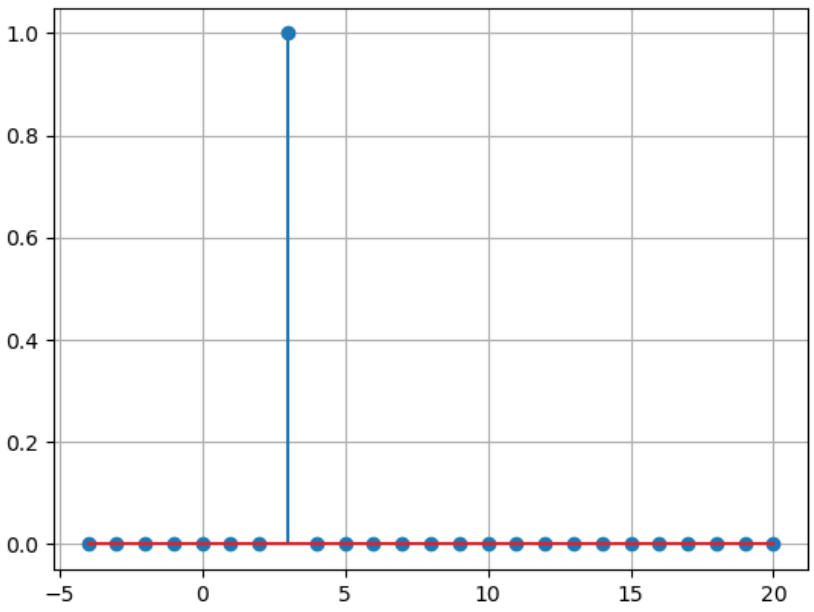
6. 连续抽样信号的产生

```
def continuous_sample_signal():
    # 使用sinc(x)计算抽样信号，sinc(x)=sin(pi*x)/(pi*x)
    t = np.linspace(-10, 10, 500, endpoint=False)
    plt.plot(t, np.sin(t))
    plt.grid()
    plt.show()
```



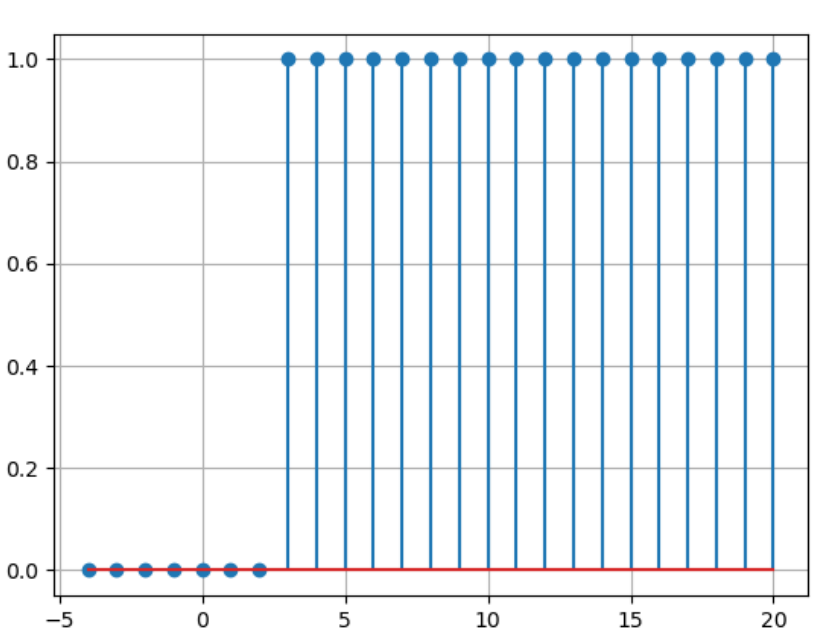
7. 单位脉冲序列的产生

```
def unit_pulse():
    # 定义时间序列
    n = np.linspace(-4, 21, 25, endpoint=False)
    # 产生单位脉冲序列
    # np.zeros((1,n))产生1行n列由0组成的矩阵
    # np.ones((1,n))产生1行n列由1组成的矩阵
    x = np.append(np.zeros((1, 7)), np.ones((1, 1)))
    x = np.append(x, np.zeros((1, 17)))
    plt.stem(n, x, use_line_collection=True)
    plt.grid()
    plt.show()
```



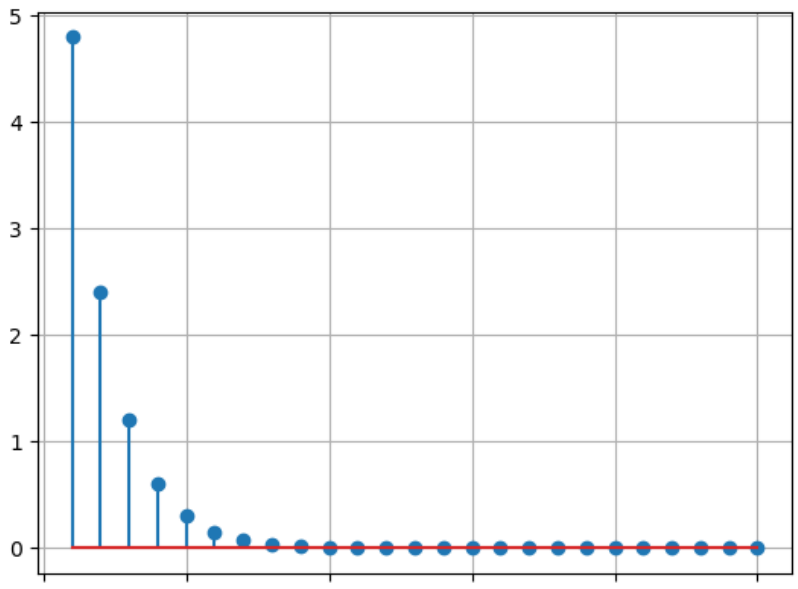
8. 单位阶跃序列的产生

```
def unit_step():
    n = np.linspace(-4, 21, 25, endpoint=False)
    x = np.append(np.zeros((1, 7)), np.ones((1, 18)))
    plt.stem(n, x, use_line_collection=True)
    plt.grid()
    plt.show()
```



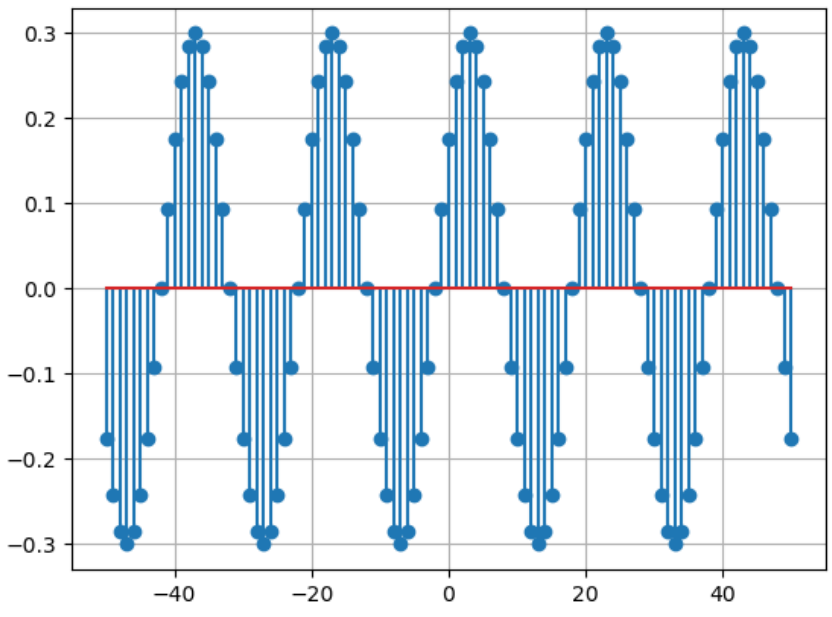
9. 指数序列的产生

```python
def exp():
    n = np.linspace(-4, 21, 25, endpoint=False)
    x = 0.3 * np.power(1 / 2, n)  # 0.3*0.5^n
    plt.stem(n, x, use_line_collection=True)
    plt.grid()
    plt.show()
```



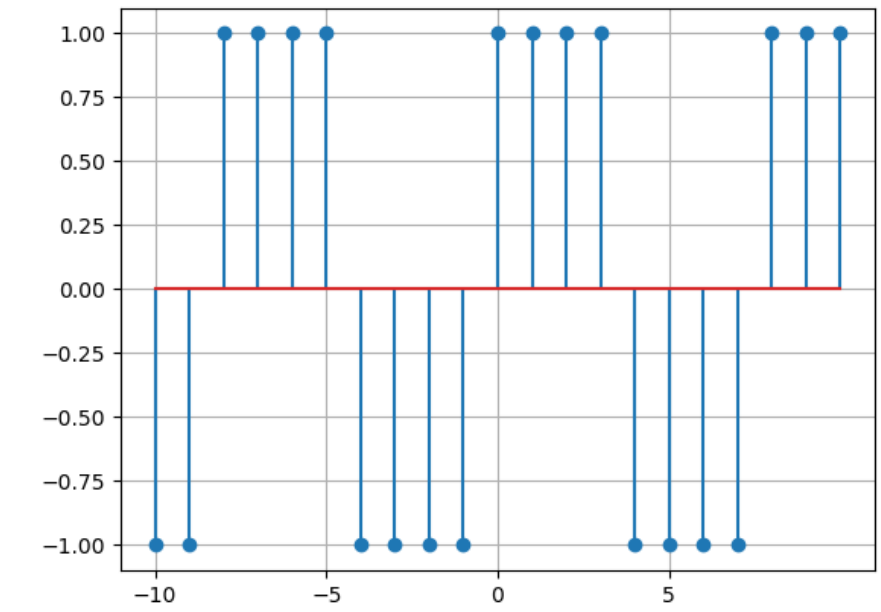
10. 正弦序列的产生

```python
def sin():
    n = np.linspace(-50, 51, 101, endpoint=False)
    omega = np.pi / 10  # 频率
    x = 0.3 * np.sin(omega * n + np.pi / 5)  # 产生正弦序列
    plt.stem(n, x, use_line_collection=True)
    plt.grid()
    plt.show()
```



11. 离散周期矩形波序列的产生

```python
def discrete_rectangle_wave():
    # 振幅为1，基频为rad，占空比为50%
    n = np.linspace(-10, 11, 21, endpoint=False)
    rad = np.pi / 4
    plt.stem(n, sg.square(rad * n, 0.5))
    plt.xticks(np.arange(-10, 10, step=5.0))  # 横坐标间距
    plt.grid()
    plt.show()
```



3. 实验代码

1、进行平移、翻转和尺度变换

（1）非周期三角波信号

```python
# 导入需要的 library 库
import numpy as np  # 科学计算
import matplotlib.pyplot as plt  # 画图

# noinspection PyPep8Naming
def triangle_wave(x, width, skew):
    # 幅度为hc=1，宽度为width,斜度为skew的三角波，skew范围[-1, 1]，当skew=0，产生对称的三角波信号
    # 产生幅度为hc，宽度为width，且以0为中心左右各展开width/2大小，斜度为skew的三角波。
    if not (-1 <= skew <= 1):
        raise Exception("skew value ERROR!")   # skew范围不对，抛出异常
    hc = 1  # 三角波默认最大幅度，可以通过外部直接乘一个幅度值改变，该点横坐标通过下式计算
    xPoint = width / 2 * skew  # 三角波信号拐点横坐标，即上升沿和下降沿的横坐标

    if (x >= width / 2) or (x <= -width / 2):  # 宽度之外的值为0
        r = 0.0
    elif x > xPoint:  # 下降沿的函数
        r = -(x - xPoint) / (width / 2 - xPoint) + hc
    else:  # 上升沿的函数
        r = (x - xPoint) / (width / 2 + xPoint) + hc
    return r

x = np.linspace(-3, 3, 1000)  # 定义时间序列
y = np.array([triangle_wave(t, 4.0, 0.5) for t in x])  # x(t)信号
y2 = np.array([triangle_wave(2 * t, 4.0, 0.5) for t in x])  # x(2t)信号
```

```
y3 = np.array([triangle_wave(1 - 2 * t, 4.0, 0.5) for t in x])  # x(1-2t)信号
fig, axs = plt.subplots(3, 1, figsize=(10, 10))  # 通过figsize调整图大小
plt.subplots_adjust(wspace=0, hspace=0.4)  # 通过hspace调整子图间距
plt.subplot(311)  # 绘制x(t)信号的子图
plt.grid()  # 显示网格
plt.title('x(t)')  # x(t)信号的子图title
plt.plot(x, y)  # 绘制x(t)信号
plt.subplot(312)  # 绘制x(t)信号的子图
plt.grid()  # 显示网格
plt.title('x(2t)')  # x(2t)信号的子图title
plt.plot(x, y2)  # 绘制x(2t)信号
plt.subplot(313)  # 绘制x(1-2t)信号的子图
plt.grid()  # 显示网格
plt.title('x(1-2t)')  # x(1-2t)信号的子图title
plt.plot(x, y3)  # 绘制x(1-2t)信号
plt.show()  # 显示图像
```

(2)对离散指数序列$x[n]=Aa^n$以及$x[0.5n]$

```
# 导入需要的 library 库
import numpy as np  # 科学计算
import matplotlib.pyplot as plt  # 画图

A = 2  # 信号幅度
a = -0.5  # 指数信号底数
n = np.linspace(1, 10, 10)  # 离散时间序列
xn = A * np.power(a, n)  # 计算x(n)信号
fig, axs = plt.subplots(2, 1, figsize=(10, 5))  # 通过figsize调整图大小
plt.subplots_adjust(wspace=0, hspace=0.4)  # 通过hspace调整子图间距
plt.subplot(211)  # 绘制x(n)信号的子图
plt.stem(n, xn, use_line_collection=True)  # 绘制x(n)信号
plt.grid()  # 显示网格
_ = plt.title('x[n]')  # x[n]信号title
xn2 = A * np.power(a, 0.5 * n)  # 计算x(0.5n)信号
plt.subplot(212)  # 绘制x(0.5n)信号的子图
plt.stem(n, xn2, use_line_collection=True)  # 绘制x(0.5n)信号
plt.grid()  # 显示网格
_ = plt.title('x[0.5n]')  # x[0.5n]信号title
plt.show()  # 显示图像
```

2、利用$x(t)=u(t)-u(t-2)+u(t-0.5)-u(t-1.5)$，$(-3\le t\le 3)$, 编写相关程序，绘制出$x(-2t)$，$x(t/2+1)$和$5x(t)$的波形。

```
import numpy as np
import matplotlib.pyplot as plt

# 定义给定的函数
def x(t):
    return u(t) - u(t - 2) + u(t - 0.5) - u(t - 1.5)

# 定义单位阶跃函数u(t)
def u(t):
    return (t >= 0).astype(float)

# 绘制x(-2t)的波形
def draw_1():
```

```
    t = np.linspace(-3, 3, 1000)
    x1 = x(-2 * t)
    plt.figure()
    plt.plot(t, x1)
    plt.title('x(-2t)')
    plt.xlabel('t')
    plt.ylabel('x(-2t)')
    plt.grid(True)
    plt.show()

# 绘制x(t/2+1)的波形
def draw_2():
    t = np.linspace(-3, 3, 1000)
    x2 = x(t / 2 + 1)
    plt.figure()
    plt.plot(t, x2)
    plt.title('x(t/2 + 1)')
    plt.xlabel('t')
    plt.ylabel('x(t/2 + 1)')
    plt.grid(True)
    plt.show()

# 绘制5x(t)的波形
def draw_3():
    t = np.linspace(-3, 3, 1000)
    x3 = 5 * x(t)
    plt.figure()
    plt.plot(t, x3)
    plt.title('5x(t)')
    plt.xlabel('t')
    plt.ylabel('5x(t)')
    plt.grid(True)
    plt.show()

draw_1()
draw_2()
draw_3()
```

3、设

$$x[n]=\begin{cases}-1 & n<-2\\ n & -2\le n\le 1\\ \frac{1}{n} & n>1\end{cases} \tag{2}$$

编写程序，绘制$x[-n],x[2n+2],x[n/2](-20\le n\le 20)$

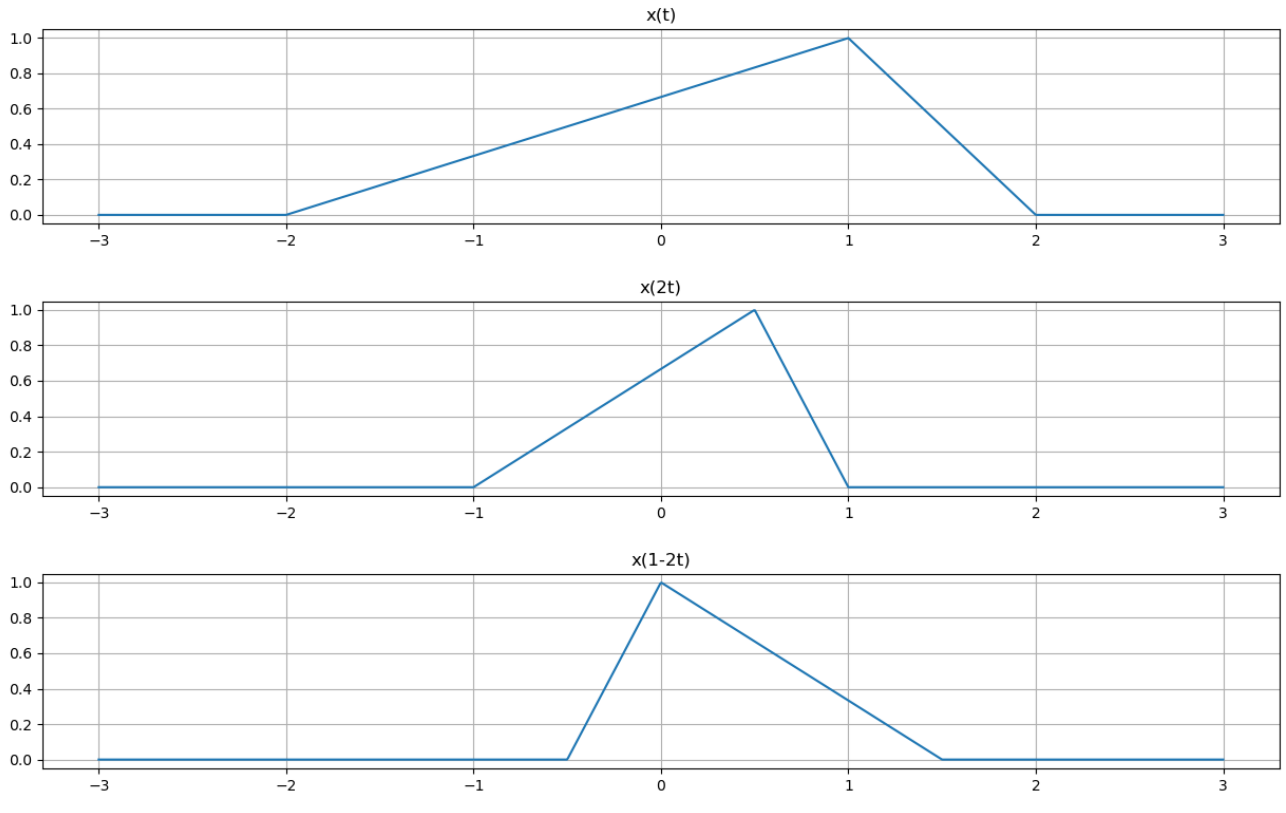
```
import numpy as np
import matplotlib.pyplot as plt

# 定义信号x[n]
def x(n):
    if n < -2:
        return -1
    elif -2 <= n <= 1:
        return n
    else:
        return 1 / n
```

```python
# 绘制x[-n]
n = np.arange(-20, 21)
xn = [x(-i) for i in n]
plt.stem(n, xn)
plt.title('x[-n]')
plt.xlabel('n')
plt.ylabel('x[-n]')
plt.grid()
plt.show()

# 绘制x[2n+2]
n = np.arange(-20, 21)
xn = [x(2 * i + 2) for i in n]
plt.stem(n, xn)
plt.title('x[2n+2]')
plt.xlabel('n')
plt.ylabel('x[2n+2]')
plt.grid()
plt.show()

# 绘制x[n/2]
n = np.arange(-20, 21)
xn = [x(0.5 * i) for i in n]
even_n = n[::2]  # 取出偶数点的n值
even_xn = [xn[i] for i in range(len(xn)) if i % 2 == 0]  # 取出偶数点的xn值
plt.stem(even_n, even_xn)
plt.title('x[n/2]')
plt.xlabel('n')
plt.ylabel('x[n/2]')
plt.grid()
plt.show()
```

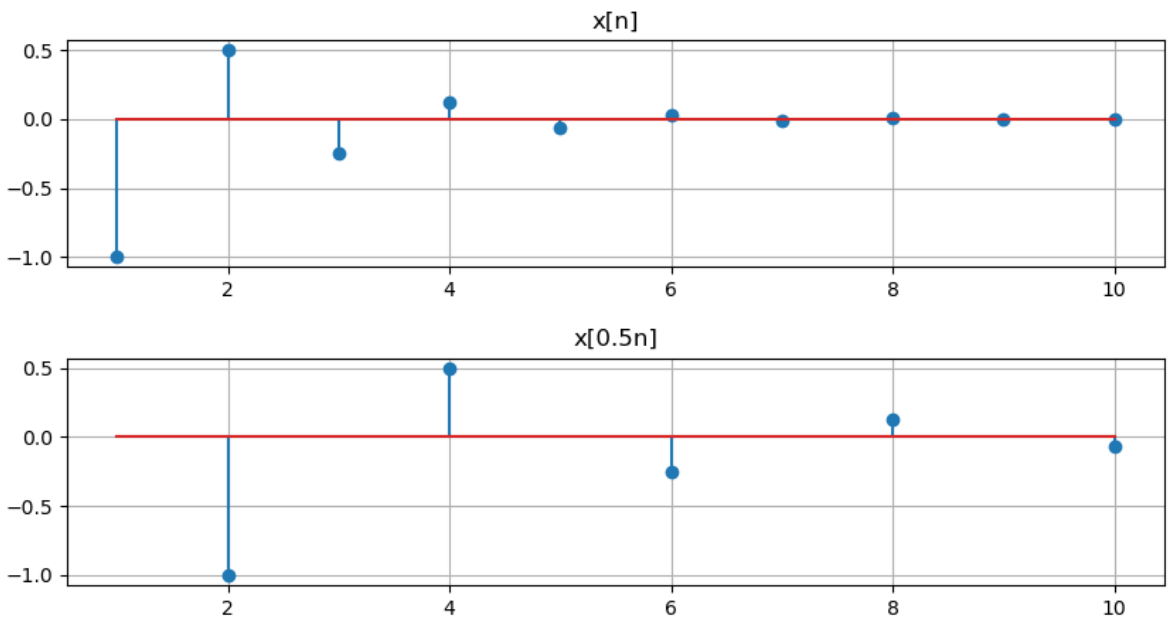
4. 实验结果

1、进行平移、翻转和尺度变换

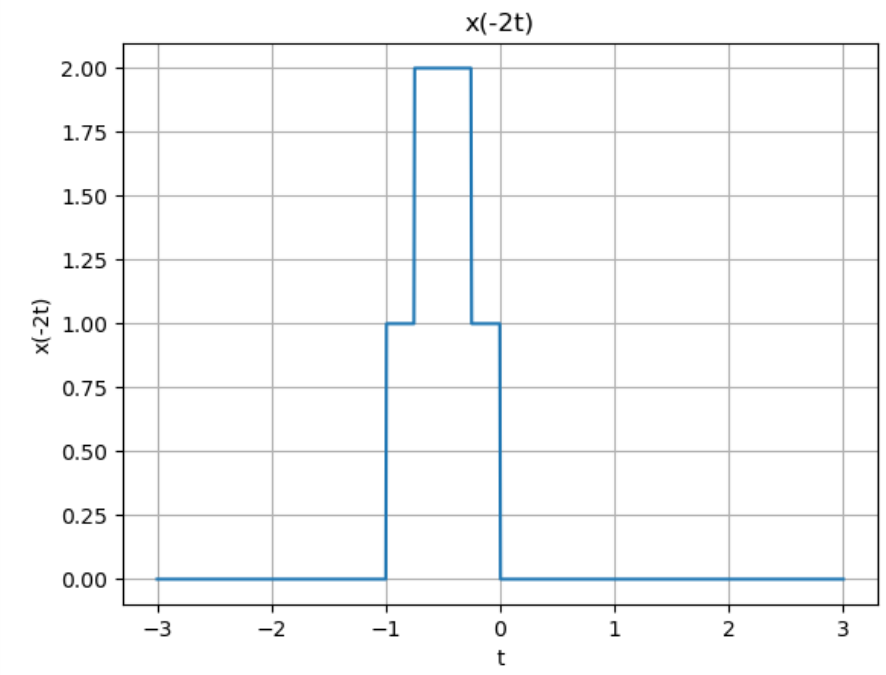
（1）非周期三角波信号,x(t)、x(2t)和x(1-2t)

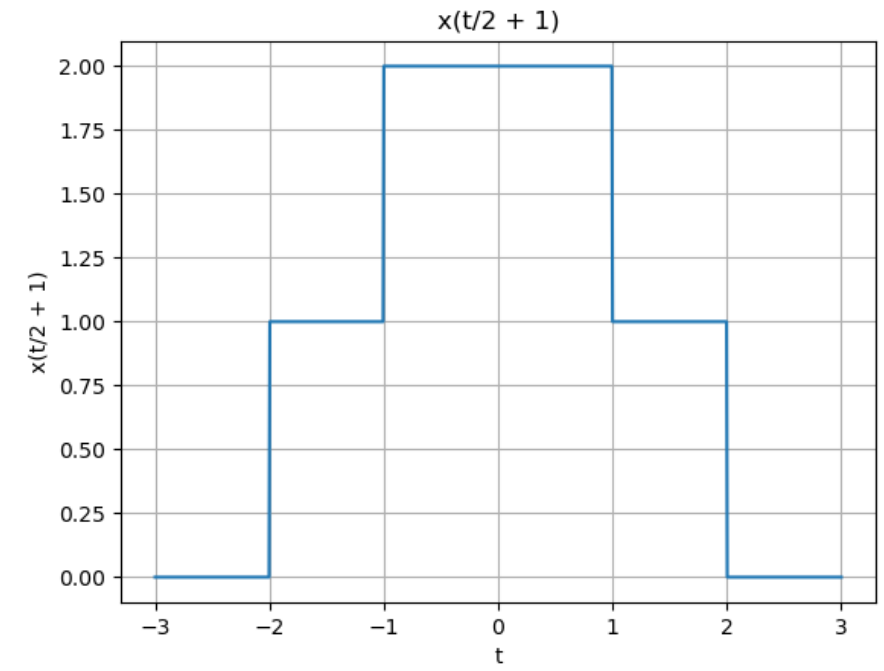


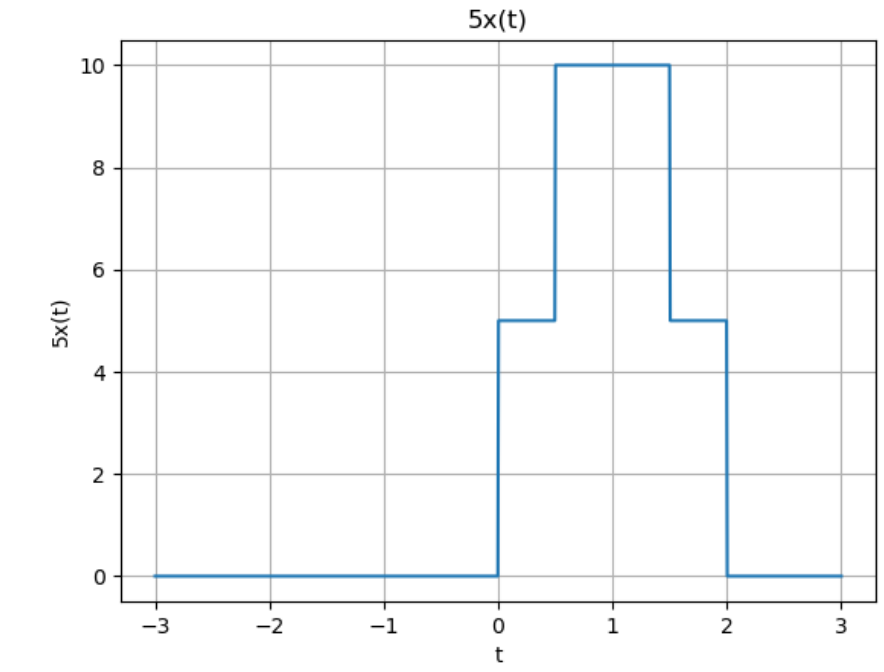
(2)对离散指数序列$x[n] = Aa^n$以及$x[0.5n]$



2、利用$x(t) = u(t) - u(t-2) + u(t-0.5) - u(t-1.5)$，$(-3 \le t \le 3)$, 绘制出$x(-2t)$，$x(t/2+1)$和$5x(t)$的波形。



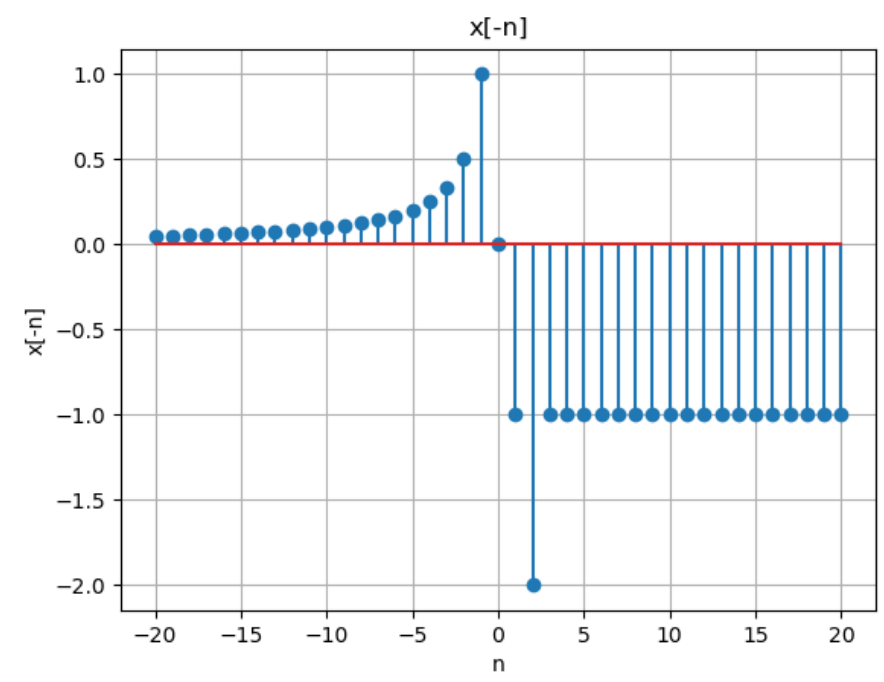


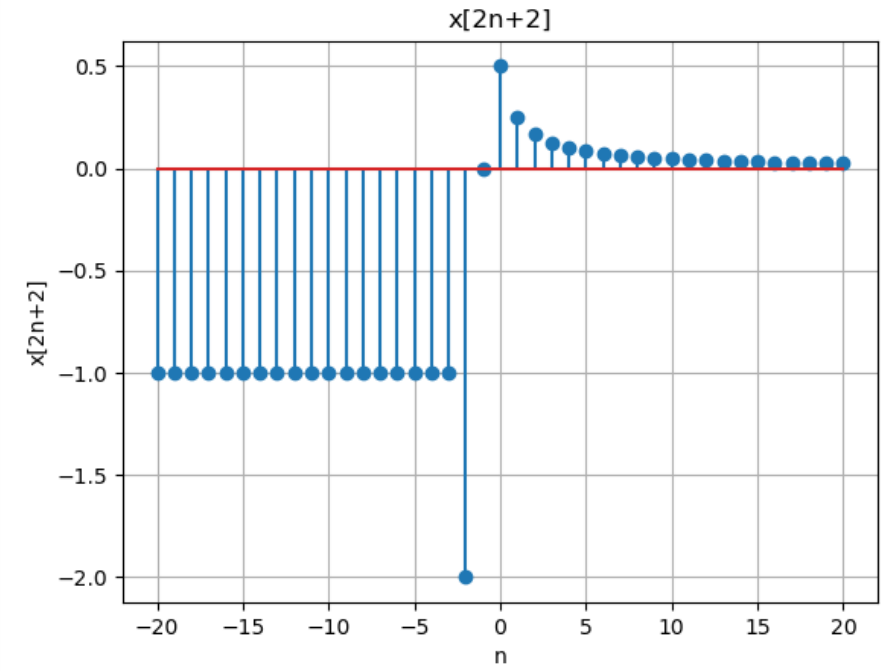


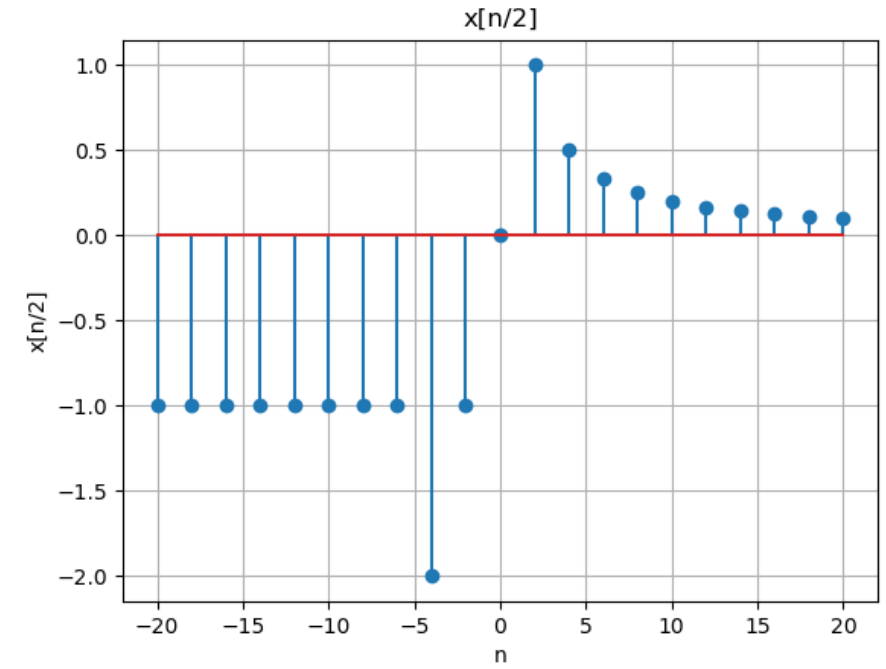
3、设

$$x[n]=\begin{cases}-1 & n<-2\\ n & -2\le n\le 1\\ \frac{1}{n} & n>1\end{cases} \tag{3}$$

绘制$x[-n],x[2n+2],x[n/2](-20\le n\le 20)$







5. 结论

实验1:

- 信号的平移、翻转和尺度变换是信号处理中非常基础和重要的操作，通过这些操作可以改变信号的时域特性，进而实现各种信号处理任务。
- 平移信号可以通过对时间变量进行加减常数的方式实现，其中加减的常数值决定了平移的距离和方向。
- 翻转信号可以通过对时间变量进行取负的方式实现，可以实现前后颠倒、上下翻转等效果。
- 尺度变换可以通过对时间变量进行缩放或拉伸的方式实现，其中缩放因子控制了信号在时间轴上的展开速度和收缩程度。这些操作都可以用 Python 的 Numpy 库和 Matplotlib 库来实现。在进行这些操作时需要注意时间变量的范围和采样点数，以免出现信号丢失、失真等问题。
- 通过对信号进行平移、翻转和尺度变换，可以实现对信号的形状、幅值、频率等特性的调整和改变，从而实现各种信号处理任务，例如滤波、降噪、特征提取等。

实验2:

- $x(-2t)$的波形是将信号在时间轴上翻转压缩。
- $x(t/2+1)$的波形是将信号在时间轴上拓展2倍，并向左平移1个单位，实现了对信号的时间拉伸和时间提前效果。
- $5x(t)$的波形是将信号在幅值上放大5倍，即变为$5x(t)$的波形，实现了对信号幅值的增强效果。
- 平移、翻转和尺度变换是信号处理中常用的基本操作，通过这些操作可以改变信号的时域特性，进而实现各种信号处理任务。在实际应用中，我们可以根据需要选择合适的变换方式，实现对信号的调整和改变。

实验3：

- 离散信号的平移可以通过改变序列的索引来实现。
- 离散信号的翻转可以通过将序列的索引取负来实现，具体来说，翻转后的序列的第n个元素为原序列的倒数第n个元素。
- 离散信号的尺度变换可以通过改变序列的索引间隔来实现，将序列的索引乘以a可以实现将信号在x轴上压缩a倍的效果，而将序列的索引除以a可以实现将信号在x轴上拉伸a倍的效果。

6. 收获与感想

钟欣余

完成信号处理的代码实现的过程中，我不仅学到了信号处理中的一些基本概念和操作，例如信号的生成、平移、翻转和尺度变换，还学会了使用Python中的NumPy和Matplotlib库来生成和可视化信号，并进行信号处理操作，对信号处理的理解有所加深，收获满满。

邓栩瀛

通过完成信号处理的代码实现，我深入理解了信号处理的基本概念和操作，也学会了如何使用Python编写代码来生成和处理信号，以及如何进行数据可视化和结果分析，这些技能对我未来的学习和实践都将非常有帮助。在实际编写代码的过程中，我也发现了一些问题，比如需要定义合适的时间范围和采样点数等，这些都需要仔细思考并做出调整。

金思琪

在实验过程中，通过编写代码来实现离散信号的尺度变换，我学习并巩固了信号处理的相关操作，学习并掌握了使用 Python 绘制信号波形的方法，包括使用 Numpy 库定义信号函数和时间范围、使用 Matplotlib 库绘制波形等，对我的信号与系统学习有很大帮助。

郑越

通过实际的操作和尝试，我学会了如何使用代码来实现信号处理中的一些常见操作，例如平移、翻转和尺度变换，并且能够自己编写代码来生成和处理不同类型的信号。更深入地理解了信号的基本概念和特性，同时也巩固了 Python 编程和信号处理的相关知识和技能。