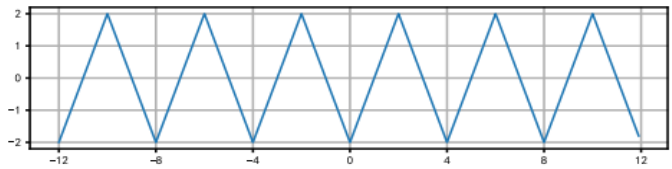
实验3

学号	姓名	分工
21307035	邓栩瀛	完成思考题
21307056	钟欣余	完成实验三中A代码的编写
21307037	金思琪	完成实验三中B代码的编写
21307062	郑越	完成实验三中C代码的编写

1、问题描述

A. 给定下图的周期三角信号$x(t)$，用 Python 分别计算并绘制其复指数形式和三角函数形式的傅里叶级数系数，并用有限项级数$xN(t)$，N = 1, … ,10，逼近$x(t)$。



B. 对 2.2 中的周期方波和上面 3-A 中的周期三角波，分别绘图演示其有限项级数$xN(t)$当项数N = 10、100、1000时对$x(t)$的逼近效果。对比分析两者逼近过程中是否出现吉布斯（Gibbs）现象并解释原因。 C. 已知微分方程

$$\frac{\mathrm{d}y(t)}{\mathrm{d}t} + 2y(t) = x(t)$$

满足初始松弛条件。计算该方程所对应系统的频率响应。将 3-A 中的周期三角波输入该系统，用 Python 求解并绘制其输出响应。

D. 完成思考题：请由式（1）中复指数形式的傅里叶级数推导式（2）中三角函数形式的傅里叶级数。

2、问题分析

实验原理

本次实验版本Python=3.6.5，numpy=1.14.5，scipy=1.1.0，sympy=1.1.1， matplotlib= 3.1.1。 对于本实验Python库约定：

```
# 导入 需要的库
import numpy as np # 科学计算
import matplotlib.pyplot as plt # 画图
import scipy.signal as sg # 导入 scipy 的 signal 库 命名为 sg
from scipy.integrate import quad # 用于求解定积分
import sympy
import warnings
warnings.filterwarnings("ignore") # 去掉常规警告
```

2.1周期信号的傅里叶级数

设$x(t)$为连续时间周期信号，其基波周期为T，基波频率为$\omega 0 = 2\pi/T$。该信号满足 Dirichlet 条件，可以展开成傅里叶级数。傅里叶级数有复指数形式和三角函数形式两种，其中复指数形式为：

$$\begin{cases} x(t) = \sum_{k=-\infty}^{\infty} a_k e^{jk\omega_0 t} \\ a_k = \frac{1}{T}\int_T x(t)e^{-jk\omega_0 t}dt \end{cases} \quad (1)$$

三角函数形式为：

$$\begin{cases} x(t) = \frac{c_0}{2} + \sum_{k=1}^{\infty} c_k \cos(k\omega_0 t) + \sum_{k=1}^{\infty} d_k \sin(k\omega_0 t) \\ c_0 = \frac{2}{T}\int_T x(t)dt \\ c_k = \frac{2}{T}\int_T x(t)\cos(k\omega_0 t)\,dt \\ d_k = \frac{2}{T}\int_T x(t)\sin(k\omega_0 t)\,dt \end{cases} \quad (2)$$

在用 Python 计算傅里叶级数的系数时，本质上是对信号进行积分运算，可采用以下两种方式进行:

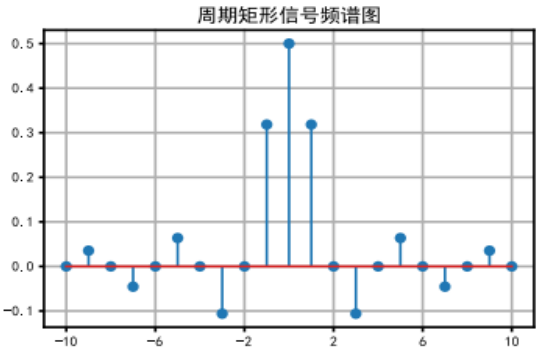
1. 符号积分：scipy. integrate(f, (x, a, b))，f 为符号表达式，x 为积分变量，a 为积分下限，b 为积分上限；
2. 数值积分：scipy. integrate .quad (func, a, b)，func 是被积分的函数名，a 为积分下限，b 为积分上限。

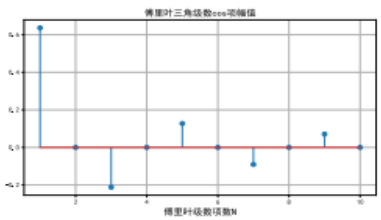
2.2程序实例

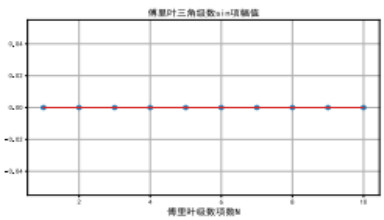
给定基波周期为 4，矩形脉冲带宽为 2，幅度为 1 的连续时间周期信号$x(t)$，用 Python 分别计算并绘制其复指数形式（experiment_3_1.py）和三角函数形式的傅里叶级数系数，并用有限项级数

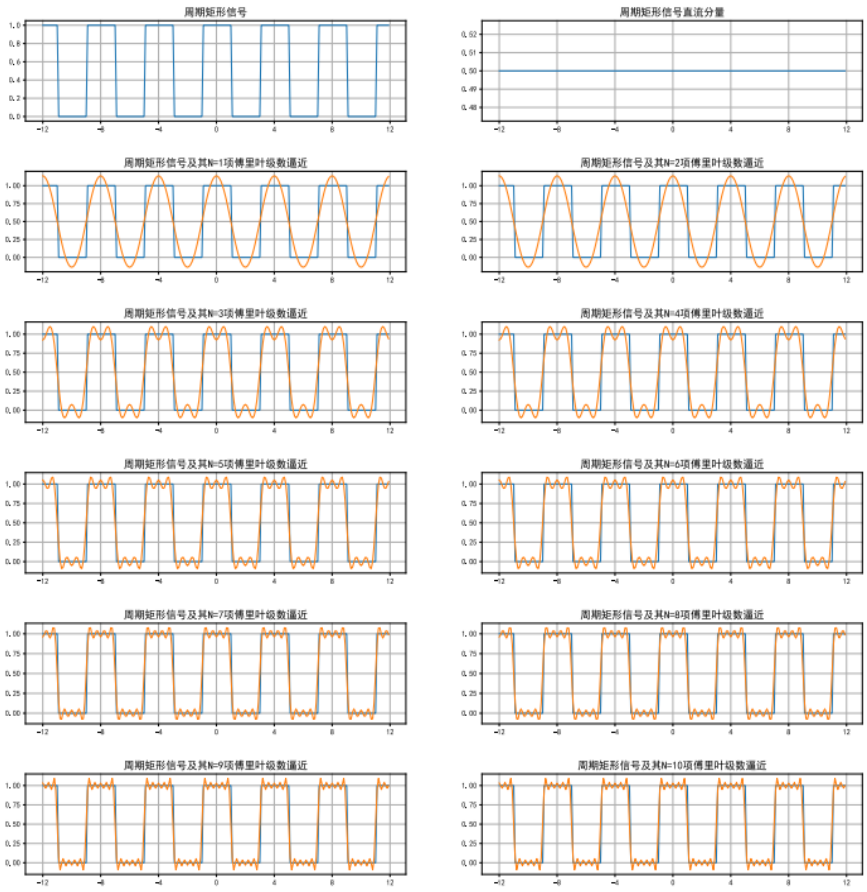
$$x_N(t) = \sum_{k=-N}^{N} a_k e^{jk\omega_0 t} = \frac{c_0}{2} + \sum_{k=1}^{N} c_k \cos(k\omega_0 t) + \sum_{k=1}^{N} d_k \sin(k\omega_0 t)$$

$N = 1, \ldots, 10$，逼近$x(t)$（experiment_3_2.py）。









3、实验代码

A

绘制$x(t)$复指数形式的傅里叶级数系数

```
# 导入 需要的 library 库
import numpy as np  # 科学计算
import matplotlib.pyplot as plt  # 画图
import scipy.signal as sg  # 导入 scipy 的 signal 库 命名为 sg
# import control as ctrl  # 导入 conctrl 库 命名为 ctrl
import sympy  # 符号运算
from scipy.integrate import quad  # 用于求解定积分
import warnings
warnings.filterwarnings("ignore") # 去掉常规警告
T = 4  # 基波周期
tao = 4  # 宽度，即有效区间[-T1,T1],tao=2*T1
omega = 2*np.pi/T  # 基波频率
A = 2  # 幅度
N = 10  # 系数显示至N次谐波
## 傅里叶级数频谱图
Xn = np.zeros(2*N+1) # 傅里叶级数指数形式的幅值
nn = np.arange(-N,N+1) # 傅里叶级数项数-N~N
for n in range(-N,N+1):
    Func = lambda t : (A*abs(t)-2)*np.exp(-1j*n*omega*t)/T
    Xn[n+N] = quad(Func,-tao/2,tao/2)[0]
```

```
plt.figure()
plt.rcParams['font.sans-serif'] = ['SimHei']  # 用来正常显示中文标签
plt.rcParams['axes.unicode_minus'] = False  # 用来正常显示负号
plt.stem(nn,Xn,use_line_collection=True) # 绘制频谱图
_ = plt.title("周期矩形信号频谱图",{'size':14})
_ = plt.xticks(np.arange(-N, N+1, step=T))  # 横坐标间隔
plt.grid(True) # 显示网格
plt.show()  # 显示图像
```

绘制$x(t)$三角函数形式的傅里叶级数系数，并用有限项级数$xN(t)$，N = 1, ... ,10，逼近$x(t)$。

```
# 导入 需要的 library 库
import numpy as np  # 科学计算
import matplotlib.pyplot as plt  # 画图
import scipy.signal as sg  # 导入 scipy 的 signal 库 命名为 sg
# import control as ctrl  # 导入 conctrl 库 命名为 ctrl
import sympy  # 符号运算
from scipy.integrate import quad  # 用于求解定积分
import warnings
warnings.filterwarnings("ignore") # 去掉常规警告

def Trian(A, tao, T, t):
    # 周期三角函数
    temp_t = abs(t)
    while(temp_t >= T):  # 将时间变量t转移到[0,T]区间上
        temp_t -= T
    if temp_t <= tao / 2:
        return A*temp_t-2
    else:
        return -A*temp_t+6  # 返回区间[0,T]内的函数值

def aNt(N, A, omega, T, tao):
    # 周期三角傅里叶级数有限项级数系数
    an = np.zeros(N+1)  # 系数an，三角形式
    bn = np.zeros(N+1)  # 系数bn，三角形式
    FuncA0 = lambda t : Trian(A, tao, T, t)*2/T # 用于计算a0
    an[0] = quad(FuncA0, -T/2, T/2)[0]  # 计算a0
    for n in range(1, N+1):
        def FuncA(t): return Trian(A, tao, T, t)*np.cos(n*omega*t)*2/T
        def FuncB(t): return Trian(A, tao, T, t)*np.sin(n*omega*t)*2/T
        an[n] = quad(FuncA, -T/2, T/2)[0]
        bn[n] = quad(FuncB, -T/2, T/2)[0]
    return an, bn  # 返回傅里叶系数，从0~N

def plotTrian(A, T, tao):
    # 绘制周期三角信号
    tt = np.arange(-3*T, 3*T, 0.1)  # 采样序列，用于绘制波形
    trian = np.array([Trian(A, tao, T, t) for t in tt])  # 产生周期三角信号
    plt.figure(figsize=(20, 5))  # 通过figsize调整图大小
    plt.rcParams['font.sans-serif'] = ['SimHei']  # 用来正常显示中文标签
    plt.rcParams['axes.unicode_minus'] = False  # 用来正常显示负号
    plt.plot(tt, trian)  # 绘制周期三角信号
```

```
    # 横坐标间隔，使用变量"_"是防止matplotlib打印输出无用信息
    _ = plt.xticks(np.arange(-3*T, 4*T, step=1.0))
    plt.title('周期三角信号', {'size': 18})  # 显示title
    _ = plt.xlabel("Time: 基波周期T=4,宽度τ=4", {'size': 18})
    plt.grid(True)  # 显示网格


def xNt(an, bn, omega, T, N, tao):
    # 周期三角傅里叶级数有限项级数
    A0 = an[0]
    t = np.arange(-3*T, 3*T, 0.1)  # 时间采样序列
    fnt = A0/2  # 直流项，即a0/2
    for n in range(1, N+1):
        fnt = fnt + an[n]*np.cos(n*omega*t) + bn[n]*np.sin(n*omega*t)  # 傅里叶级数
N项逼近值
    return fnt


def plot_xNt(an, bn, omega, T, N, tao):
    tt = np.arange(-3*T, 3*T, 0.1)  # 时间采样序列
    plt.figure(figsize=(20, 20))  # 通过figsize调整图大小
    plt.rcParams['font.sans-serif'] = ['SimHei']  # 用来正常显示中文标签
    plt.rcParams['axes.unicode_minus'] = False  # 用来正常显示负号
    plt.subplots_adjust(wspace=0.2, hspace=1.0)  # 子图间距调整
    plt.subplot((N+2)//2, 2, 1)  # 绘制周期三角信号的子图
    plt.plot(tt, np.array([Trian(A, tao, T, t) for t in tt]))  # 绘制周期三角信号
    _ = plt.xticks(np.arange(-3*T, 4*T, step=T))
    plt.title("周期三角信号", {'size': 14})
    plt.grid(True)  # 显示网格
    plt.subplot((N+2)//2, 2, 2)  # 绘制周期三角信号直流分量的子图
    plt.plot(tt, np.ones_like(tt)*an[0]/2)  # 绘制周期三角信号直流分量
    _ = plt.xticks(np.arange(-3*T, 4*T, step=T))
    plt.title("周期三角信号直流分量", {'size': 14})
    plt.grid(True)  # 显示网格
    for i in range(1, N+1):
        # 显示N=1~10项傅里叶级数逼近
        plt.subplot((N+2)//2, 2, i+2)  # 绘制周期三角信号及其N项傅里叶级数逼近的子图
        plt.plot(tt, np.array([Trian(A, tao, T, t) for t in tt]))  # 绘制周期三角信
号
        plt.plot(tt, xNt(an, bn, omega, T, i, tao))  # N=i项傅里叶级数逼近
        _ = plt.xticks(np.arange(-3*T, 4*T, step=T))  # 横坐标间隔
        plt.title(f"周期三角信号及其N={i}项傅里叶级数逼近", {'size': 14})
        plt.grid(True)  # 显示网格
    plt.show()  # 显示图像

T = 4  # 基波周期
tao = 4  # 宽度，即有效区间[-T1,T1],tao=2*T1
omega = 2*np.pi/T  # 基波频率
A = 2  # 幅度
N = 10  # N项有限项级数逼近，可修改N值显示不同的逼近效果
n = np.arange(0, N+1)  # 0~N傅里叶级数项数序列
an, bn = aNt(N, A, omega, T, tao)  # 0~N傅里叶级数幅度值
# 绘制周期矩形信号傅里叶级数三角形式幅值
fig, axs = plt.subplots(1, 2, figsize=(20, 5))  # 通过figsize调整图大小
```

```
plt.rcParams['font.sans-serif'] = ['SimHei']  # 用来正常显示中文标签
plt.rcParams['axes.unicode_minus'] = False  # 用来正常显示负号
plt.subplots_adjust(wspace=0.2, hspace=0)  # 通过wspace调整子图间距
plt.subplot(121)  # 绘制傅里叶三角级数cos项赋值的子图
plt.stem(n[1:], an[1:], use_line_collection=True)  # 傅里叶三角级数cos项赋值
plt.grid(True)  # 显示网格
plt.title('傅里叶三角级数cos项幅值', {'size': 14})  # 显示title
_ = plt.xlabel("傅里叶级数项数N", {'size': 14})
plt.subplot(122)  # 绘制傅里叶三角级数sin项赋值的子图
plt.stem(n[1:], bn[1:], use_line_collection=True)  # 傅里叶三角级数sin项赋值
plt.title('傅里叶三角级数sin项幅值', {'size': 14})  # 显示title
_ = plt.xlabel("傅里叶级数项数N", {'size': 14})
plt.grid(True)  # 显示网格
plt.show()  # 显示图像
# 有限项级数逼近
plot_xNt(an, bn, omega, T, N, tao)
```

B

方波

```
# 导入 需要的 library 库
import numpy as np  # 科学计算
import matplotlib.pyplot as plt  # 画图
import scipy.signal as sg  # 导入 scipy 的 signal 库 命名为 sg
# import control as ctrl  # 导入 conctrl 库 命名为 ctrl
import sympy  # 符号运算
from scipy.integrate import quad  # 用于求解定积分

import warnings

warnings.filterwarnings("ignore")  # 去掉常规警告

def Rect(A, tao, T, t):
    # 周期矩形函数
    temp_t = abs(t)
    while (temp_t >= T):  # 将时间变量t转移到[0,T]区间上
        temp_t -= T
    return A if temp_t <= tao / 2 or temp_t >= T - tao / 2 else 0  # 返回区间[0,T]
内的函数值


def aNt(N, A, omega, T, tao):
    # 周期矩形傅里叶级数有限项级数系数
    an = np.zeros(N + 1)  # 系数an，三角形式
    bn = np.zeros(N + 1)  # 系数bn，三角形式
    FuncA0 = lambda t: Rect(A, tao, T, t) * 2 / T  # 用于计算a0
    an[0] = quad(FuncA0, -T / 2, T / 2)[0]  # 计算a0
    for n in range(1, N + 1):
        def FuncA(t): return Rect(A, tao, T, t) * np.cos(n * omega * t) * 2 / T
```

```
        def FuncB(t): return Rect(A, tao, T, t) * np.sin(n * omega * t) * 2 / T

        an[n] = quad(FuncA, -T / 2, T / 2, limit=1000)[0]
        bn[n] = quad(FuncB, -T / 2, T / 2, limit=1000)[0]
    return an, bn  # 返回傅里叶系数，从0~N


def plotSquare(A, T, tao):
    # 绘制周期矩形信号
    tt = np.arange(-3 * T, 3 * T, 0.001)  # 采样序列，用于绘制波形
    rect = np.array([Rect(A, tao, T, t) for t in tt])  # 产生周期矩形信号
    plt.figure(figsize=(20, 5))  # 通过figsize调整图大小
    plt.rcParams['font.sans-serif'] = ['SimHei']  # 用来正常显示中文标签
    plt.rcParams['axes.unicode_minus'] = False  # 用来正常显示负号
    plt.plot(tt, rect)  # 绘制周期矩形信号
    # 横坐标间隔，使用变量"_"是防止matplotlib打印输出无用信息
    _ = plt.xticks(np.arange(-3 * T, 4 * T, step=1.0))
    plt.title('周期矩形信号', {'size': 18})  # 显示title
    _ = plt.xlabel("Time: 基波周期T=4,宽度τ=2", {'size': 18})
    plt.grid(True)  # 显示网格


def xNt(an, bn, omega, T, N, tao):
    # 周期矩形傅里叶级数有限项级数
    A0 = an[0]
    t = np.arange(-3 * T, 3 * T, 0.001)  # 时间采样序列
    fnt = A0 / 2  # 直流项，即a0/2
    for n in range(1, N + 1):
        fnt = fnt + an[n] * np.cos(n * omega * t) + bn[n] * np.sin(n * omega * t)
# 傅里叶级数N项逼近值
    return fnt


def plot_xNt(an, bn, omega, T, N, tao):
    tt = np.arange(-3 * T, 3 * T, 0.001)  # 时间采样序列
    plt.figure(figsize=(20, 20))  # 通过figsize调整图大小
    plt.rcParams['font.sans-serif'] = ['SimHei']  # 用来正常显示中文标签
    plt.rcParams['axes.unicode_minus'] = False  # 用来正常显示负号
    plt.subplots_adjust(wspace=0.2, hspace=1.0)  # 子图间距调整
    for i in range(N - 5, N + 1):
        plt.subplot((6 + 2) // 2, 2, i - N+6)  # 绘制周期矩形信号及其N项傅里叶级数逼
近的子图
        plt.plot(tt, np.array([Rect(A, tao, T, t) for t in tt]))  # 绘制周期矩形信
号
        plt.plot(tt, xNt(an, bn, omega, T, i, tao))  # N=i项傅里叶级数逼近
        _ = plt.xticks(np.arange(-3 * T, 4 * T, step=T))  # 横坐标间隔
        plt.title(f"周期矩形信号及其N={i}项傅里叶级数逼近", {'size': 14})
        plt.grid(True)  # 显示网格
    plt.show()  # 显示图像


T = 4  # 基波周期
tao = 2  # 宽度，即有效区间[-T1,T1],tao=2*T1
omega = 2 * np.pi / T  # 基波频率
```

```
A = 1  # 幅度
N = 1000  # N项有限项级数逼近，可修改N值显示不同的逼近效果
n = np.arange(0, N + 1)  # 0~N傅里叶级数项数序列
an, bn = aNt(N, A, omega, T, tao)  # 0~N傅里叶级数幅度值
# 有限项级数逼近
plot_xNt(an, bn, omega, T, N, tao)
```

三角

```
# 导入 需要的 library 库
import numpy as np  # 科学计算
import matplotlib.pyplot as plt  # 画图
import scipy.signal as sg  # 导入 scipy 的 signal 库 命名为 sg
# import control as ctrl  # 导入 conctrl 库 命名为 ctrl
import sympy  # 符号运算
from scipy.integrate import quad  # 用于求解定积分

import warnings
warnings.filterwarnings("ignore") # 去掉常规警告


def Trian(A, tao, T, t):
    # 周期三角函数
    temp_t = abs(t)
    while(temp_t >= T):  # 将时间变量t转移到[0,T]区间上
        temp_t -= T
    if temp_t <= tao / 2:
        return A*temp_t-2
    else:
        return -A*temp_t+6  # 返回区间[0,T]内的函数值


def aNt(N, A, omega, T, tao):
    # 周期三角傅里叶级数有限项级数系数
    an = np.zeros(N+1)  # 系数an，三角形式
    bn = np.zeros(N+1)  # 系数bn，三角形式
    FuncA0 = lambda t : Trian(A, tao, T, t)*2/T # 用于计算a0
    an[0] = quad(FuncA0, -T/2, T/2)[0]  # 计算a0
    for n in range(1, N+1):
        def FuncA(t): return Trian(A, tao, T, t)*np.cos(n*omega*t)*2/T
        def FuncB(t): return Trian(A, tao, T, t)*np.sin(n*omega*t)*2/T
        an[n] = quad(FuncA, -T/2, T/2,limit=1000)[0]
        bn[n] = quad(FuncB, -T/2, T/2,limit=1000)[0]
    return an, bn  # 返回傅里叶系数，从0~N


def plotTrian(A, T, tao):
    # 绘制周期三角信号
    tt = np.arange(-3*T, 3*T, 0.001)  # 采样序列，用于绘制波形
    trian = np.array([Trian(A, tao, T, t) for t in tt])  # 产生周期三角信号
    plt.figure(figsize=(20, 5))  # 通过figsize调整图大小
    plt.rcParams['font.sans-serif'] = ['SimHei']  # 用来正常显示中文标签
```

```
    plt.rcParams['axes.unicode_minus'] = False  # 用来正常显示负号
    plt.plot(tt, trian)  # 绘制周期三角信号
    # 横坐标间隔，使用变量"_"是防止matplotlib打印输出无用信息
    _ = plt.xticks(np.arange(-3*T, 4*T, step=1.0))
    plt.title('周期三角信号', {'size': 18})  # 显示title
    _ = plt.xlabel("Time: 基波周期T=4,宽度τ=4", {'size': 18})
    plt.grid(True)  # 显示网格


def xNt(an, bn, omega, T, N, tao):
    # 周期三角傅里叶级数有限项级数
    A0 = an[0]
    t = np.arange(-3*T, 3*T, 0.001)  # 时间采样序列
    fnt = A0/2  # 直流项，即a0/2
    for n in range(1, N+1):
        fnt = fnt + an[n]*np.cos(n*omega*t) + bn[n]*np.sin(n*omega*t)  # 傅里叶级数
N项逼近值
    return fnt


def plot_xNt(an, bn, omega, T, N, tao):
    tt = np.arange(-3*T, 3*T, 0.001)  # 时间采样序列
    plt.figure(figsize=(20, 20))  # 通过figsize调整图大小
    plt.rcParams['font.sans-serif'] = ['SimHei']  # 用来正常显示中文标签
    plt.rcParams['axes.unicode_minus'] = False  # 用来正常显示负号
    plt.subplots_adjust(wspace=0.2, hspace=1.0)  # 子图间距调整
    for i in range(N-5, N+1):
        plt.subplot((6+2)//2, 2, i-N+6)  # 绘制周期三角信傅里叶级数逼近的子图
        plt.plot(tt, np.array([Trian(A, tao, T, t) for t in tt]))  # 绘制周期三角信
号
        plt.plot(tt, xNt(an, bn, omega, T, i, tao))  # N=i项傅里叶级数逼近
        _ = plt.xticks(np.arange(-3*T, 4*T, step=T))  # 横坐标间隔
        plt.title(f"周期三角信号及其N={i}项傅里叶级数逼近", {'size': 14})
        plt.grid(True)  # 显示网格
    plt.show()  # 显示图像
T = 4  # 基波周期
tao = 4  # 宽度，即有效区间[-T1,T1],tao=2*T1
omega = 2*np.pi/T  # 基波频率
A = 2  # 幅度
N = 1000  # N项有限项级数逼近，可修改N值显示不同的逼近效果
n = np.arange(0, N+1)  # 0~N傅里叶级数项数序列
an, bn = aNt(N, A, omega, T, tao)  # 0~N傅里叶级数幅度值

# 有限项级数逼近
plot_xNt(an, bn, omega, T, N, tao)
```

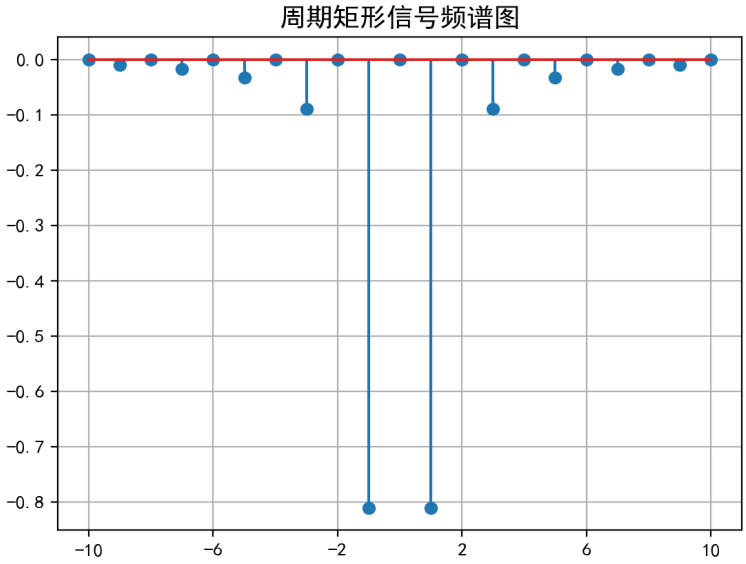
C

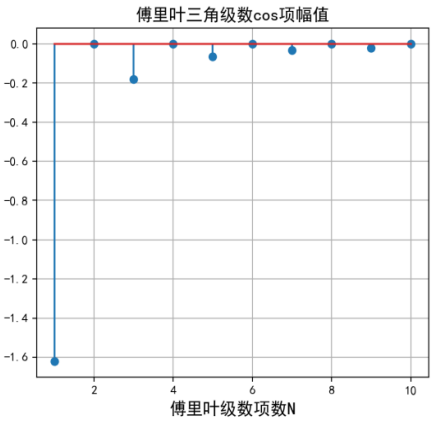
将 3-A 中的周期三角波输入该系统，用 Python 求解并绘制其输出响应。

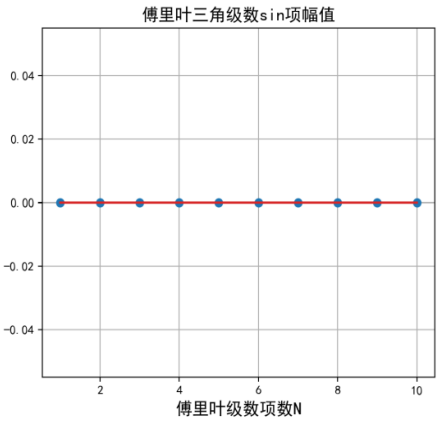
```
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt
def input_signal(t):
    if t > -8 and t <= 12:
        return input_signal(t - 4)
    elif t >= -12 and t <= -10:
        return 2 * t + 22
    elif t > -10 and t <= -8:
        return -2 * t - 18
    else:
        return 0

def triangle_wave(t, T):
    t = t % T
    if t < T / 2:
        return 4 * t / T - 1
    else:
        return 3 - 4 * t / T

def system(y, t):
    dydt = -2 * y + input_signal(t)
    return dydt

def solve(T):
    t = np.linspace(-12, 12, 4000)
    y0 = 0
    y = odeint(system, y0, t)
    return t, y[:, 0]

t, x = np.linspace(-12, 12, 4000), np.zeros(4000)
for i in range(4000):
    x[i] = input_signal(t[i])

t, y = solve(6)

plt.plot(t, x, label='Input')
plt.plot(t, y, label='Output')
plt.legend()
plt.grid()
plt.show()
```

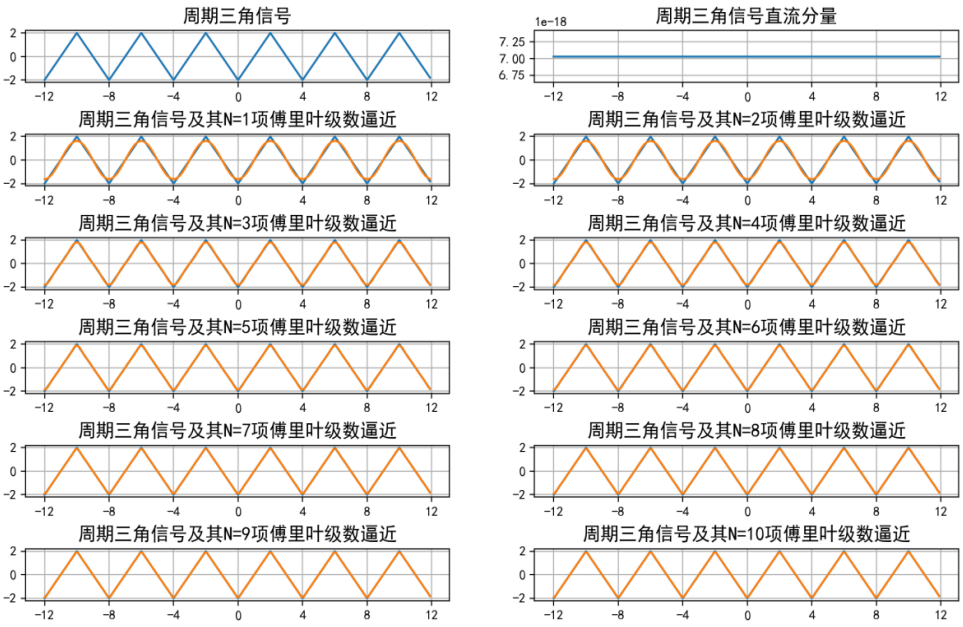
4、实验结果

A


周期矩形信号频谱图


傅里叶三角级数cos项幅值
傅里叶级数项数N

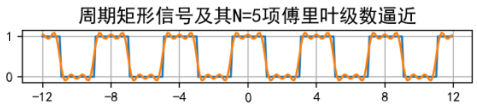

傅里叶三角级数sin项幅值
傅里叶级数项数N

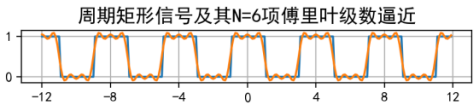

周期三角信号
周期三角信号直流分量
周期三角信号及其N=1项傅里叶级数逼近
周期三角信号及其N=2项傅里叶级数逼近
周期三角信号及其N=3项傅里叶级数逼近
周期三角信号及其N=4项傅里叶级数逼近
周期三角信号及其N=5项傅里叶级数逼近
周期三角信号及其N=6项傅里叶级数逼近
周期三角信号及其N=7项傅里叶级数逼近
周期三角信号及其N=8项傅里叶级数逼近
周期三角信号及其N=9项傅里叶级数逼近
周期三角信号及其N=10项傅里叶级数逼近

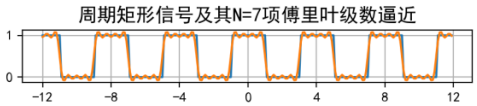
B

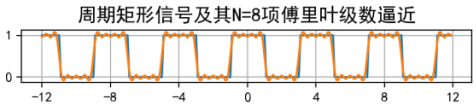
方波

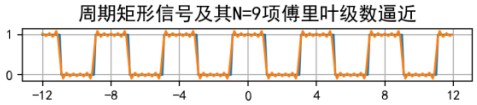
N = 10

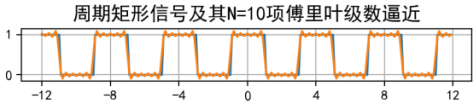

周期矩形信号及其N=5项傅里叶级数逼近


周期矩形信号及其N=6项傅里叶级数逼近

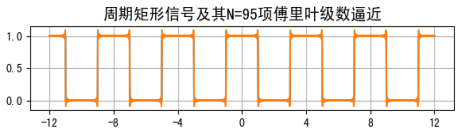

周期矩形信号及其N=7项傅里叶级数逼近

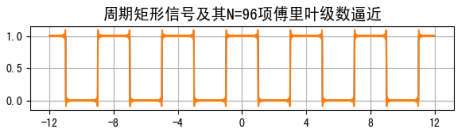

周期矩形信号及其N=8项傅里叶级数逼近

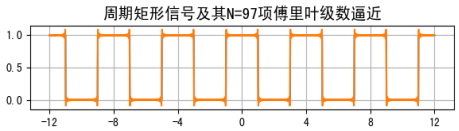

周期矩形信号及其N=9项傅里叶级数逼近

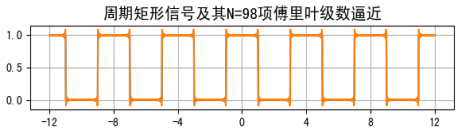

周期矩形信号及其N=10项傅里叶级数逼近

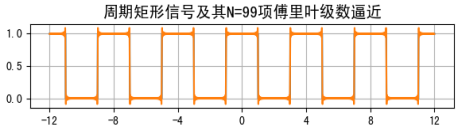
N = 100

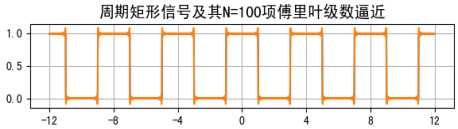

周期矩形信号及其N=95项傅里叶级数逼近


周期矩形信号及其N=96项傅里叶级数逼近

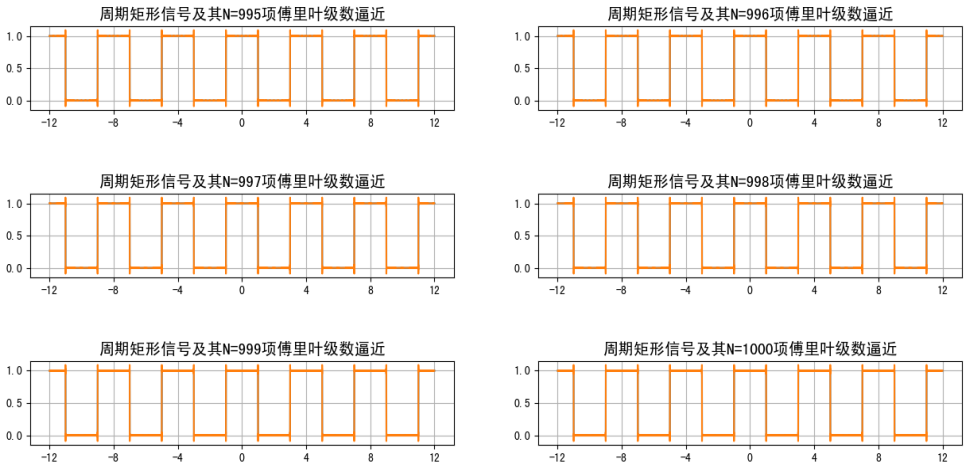

周期矩形信号及其N=97项傅里叶级数逼近


周期矩形信号及其N=98项傅里叶级数逼近


周期矩形信号及其N=99项傅里叶级数逼近


周期矩形信号及其N=100项傅里叶级数逼近

N = 1000



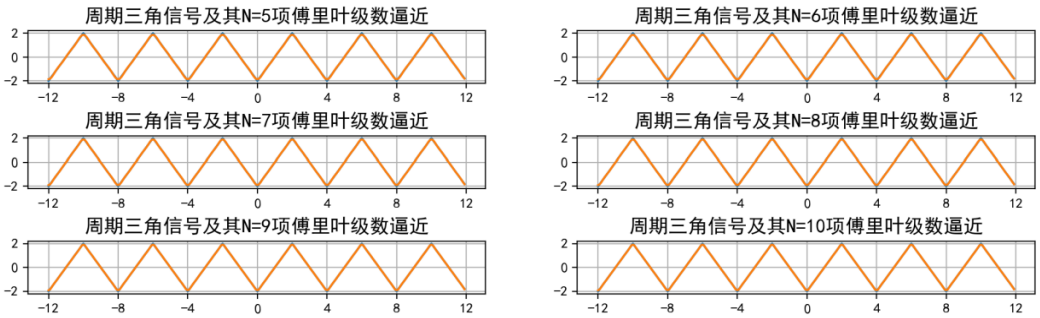
当N = 10时，我们得到以下逼近结果： 可以看出，在级数中出现了明显的吉布斯现象。这是因为截断傅里叶级数导致了级数中的振铃效应，从而使逼近函数在跳跃点处呈现过冲现象。

当N = 100时，我们得到以下逼近结果： 可以看出，随着项数的增加，吉布斯现象逐渐减弱，逼近效果也逐渐变好。

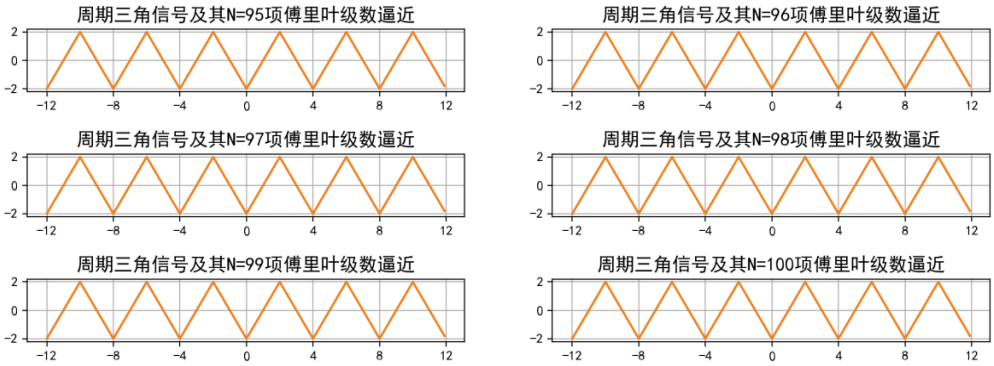
当N = 1000时，我们得到以下逼近结果： 可以看出，随着项数的进一步增加，逼近效果进一步变好，但是改善效果逐渐减小。

三角

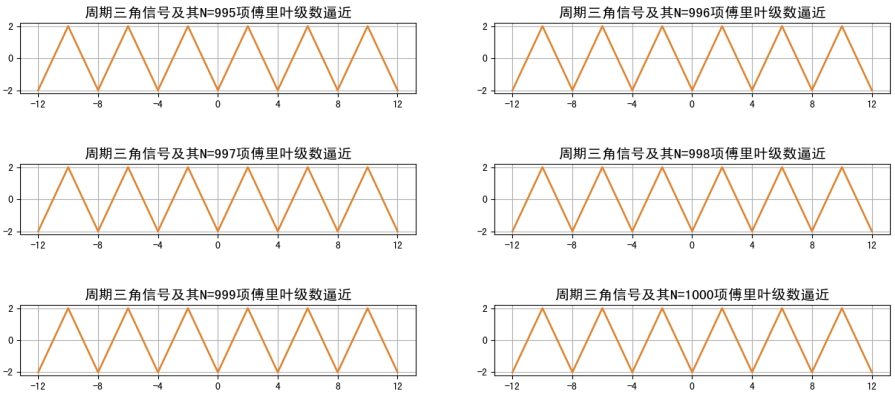
N = 10



N = 100

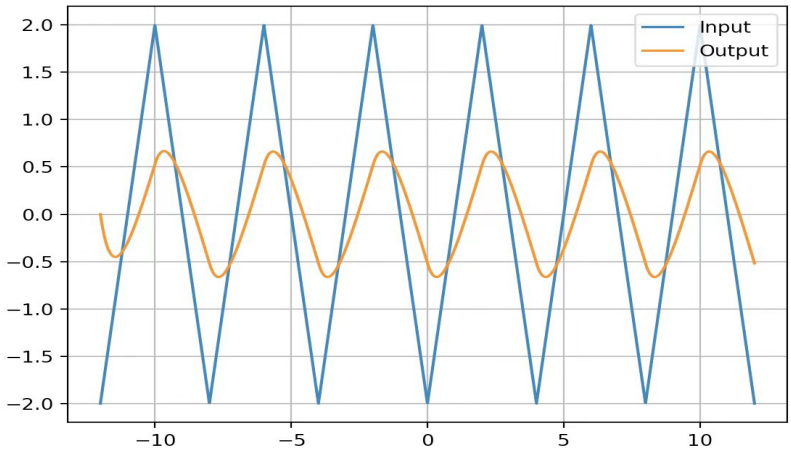


N = 1000



当$x(t)$是一个三角函数时，无论N取多少，都不会出现吉布斯现象。当N越大时，逼近效果会越来越好，但计算复杂度也会越来越高。在实际应用中，需要根据具体情况选择合适的N值，以达到逼近精度和计算效率的平衡。

C



5、思考题

$$x(t)=\sum_{k=-\infty}^{\infty} a_k e^{jk\omega_0 t} = a_0+\sum_{k=1}^{\infty}(a_k e^{jk\omega_0 t}+a_{-k}e^{-jk\omega_0 t})$$

$$= a_0+\sum_{k=1}^{\infty}[(a_k+a_{-k})\cos(k\omega_0 t)+(a_k-a_{-k})j\sin(k\omega_0 t)]$$

如果 $x(t)$ 为实信号

令 $a_k = c_k' + jd_k'$，$a_{-k} = c_{-k}' + jd_{-k}'$

则 $a_k + a_{-k}$ 为实数，$a_k - a_{-k}$ 为虚

所以 $d_k' + d_{-k}' = 0$，$c_k' - c_{-k}' = 0$，$d_k' - d_{-k}' = 0$

得 $d_k' = d_{-k}' = 0$，$c_k' = c_{-k}'$

所以 $a_k = a_{-k}^*$

令 $a_k = \frac{c_k - jd_k}{2}$，则有 $a_{-k} = \frac{c_k + jd_k}{2}$，$a_0 = \frac{c_0}{2}$

$$a_k = \frac{c_k - jd_k}{2} = \frac{1}{T}\int_T x(t)e^{-jk\omega_0 t}dt$$

$$= \frac{1}{T}\int_T x(t)[\cos(k\omega_0 t) - j\sin(k\omega_0 t)]dt$$

$$= \frac{1}{T}\int_T x(t)\cos(k\omega_0 t)dt - \frac{1}{T}\int_T x(t)j\sin(k\omega_0 t)dt$$

$$\therefore\ c_0 = \frac{2}{T}\int_T x(t)dt$$

$$c_k = \frac{2}{T}\int_T x(t)\cos(k\omega_0 t)dt$$

$$d_k = \frac{2}{T}\int_T x(t)\sin(k\omega_0 t)dt$$

6、收获与感想

邓栩瀛：从复指数形式的傅里叶级数推导三角函数形式的傅里叶级数的过程涉及到复数和三角函数之间的转换，通过这种变换可以让我们在信号处理中更加容易地理解问题，从而求解相应的问题。通过这次实验，我更深入地理解傅里叶级数的特性和应用。傅里叶级数是一种非常重要的数学工具，在信号处理中有着广泛的应用。傅里叶级数可以将一个周期性信号分解为一组基本频率的正弦和余弦函数的和，而通过这种分解，我们能够更加容易地理解信号的频率组成，可以了解到信号中哪些频率成分的主次关系，从而设计和优化相关的信号处理算法。

钟欣余：完成实验的过程中，我深刻体会到傅里叶级数是一种非常重要的数学工具，可以将任意周期函数表示为一组三角函数或复指数的线性组合，这对于理解和分析周期信号非常有帮助。通过分析实验结果，可以看出傅里叶级数的逼近效果随着级数的增加而变得更加精确，但计算复杂度也随之增加。因此，在实际应用中需要根据具体情况选择合适的级数来进行逼近。

金思琪：在本次实验中，我更深刻地体会到了Python在数学问题求解中的实用性和方便性。同时，通过实验结果，我对“傅里叶级数的逼近效果随着级数的增加而变得更加精确，但计算复杂度也随之增加”有了更形象的理解。因此在实际应用中，我们需要根据具体情况选择合适的级数来进行逼近。此外，这次实验还加深了我对吉布斯现象的印象——傅里叶级数在逼近函数时可能产生振铃现象，我们需要注意避免吉布斯现象对结果的影响。

郑越：在本次实验中，我学会了使用Python的第三方库，如NumPy和SciPy来求解微分方程和计算频率响应。其次，我学会了如何计算一个系统的频率响应。首先需要将微分方程转化为拉普拉斯域形式，然后通过计算系统的传递函数来得到系统的频率响应。频率响应可以帮助我们了解系统对不同频率信号的响应特性。然后是明白了松弛条件对微分方程解的影响。松弛条件是指在初始时刻系统对输入信号的响应为零。在实验中，可以看到在有松弛条件的情况下，系统的响应与无松弛条件的情况有所不同，这进一步说明了松弛条件在系统分析中的重要性。