****

数字信号处理实验报告

专业： 电子信息工程

班别： 电信专升本201 班

姓名： 陈黎军

学号： 3200442029

**浙大宁波理工学院实验报告**

**姓名 陈黎军 学号 3200442029 学院** 信息科学与工程学院

**专业 电子信息工程 班级 电信专升本201 课程** 数字信号处理

**实验时间 2021-6-8 实验地点 指导教师**

**实验一**

**实验目的：**

通过各种波形频谱了解谐波结构

**实验内容：**

1、生成锯齿波，幅度为-1到1，并显示频谱

2、比较锯齿波与三角波和方波的谐波结构

3、创建1100hz的方波信号，绘制频谱

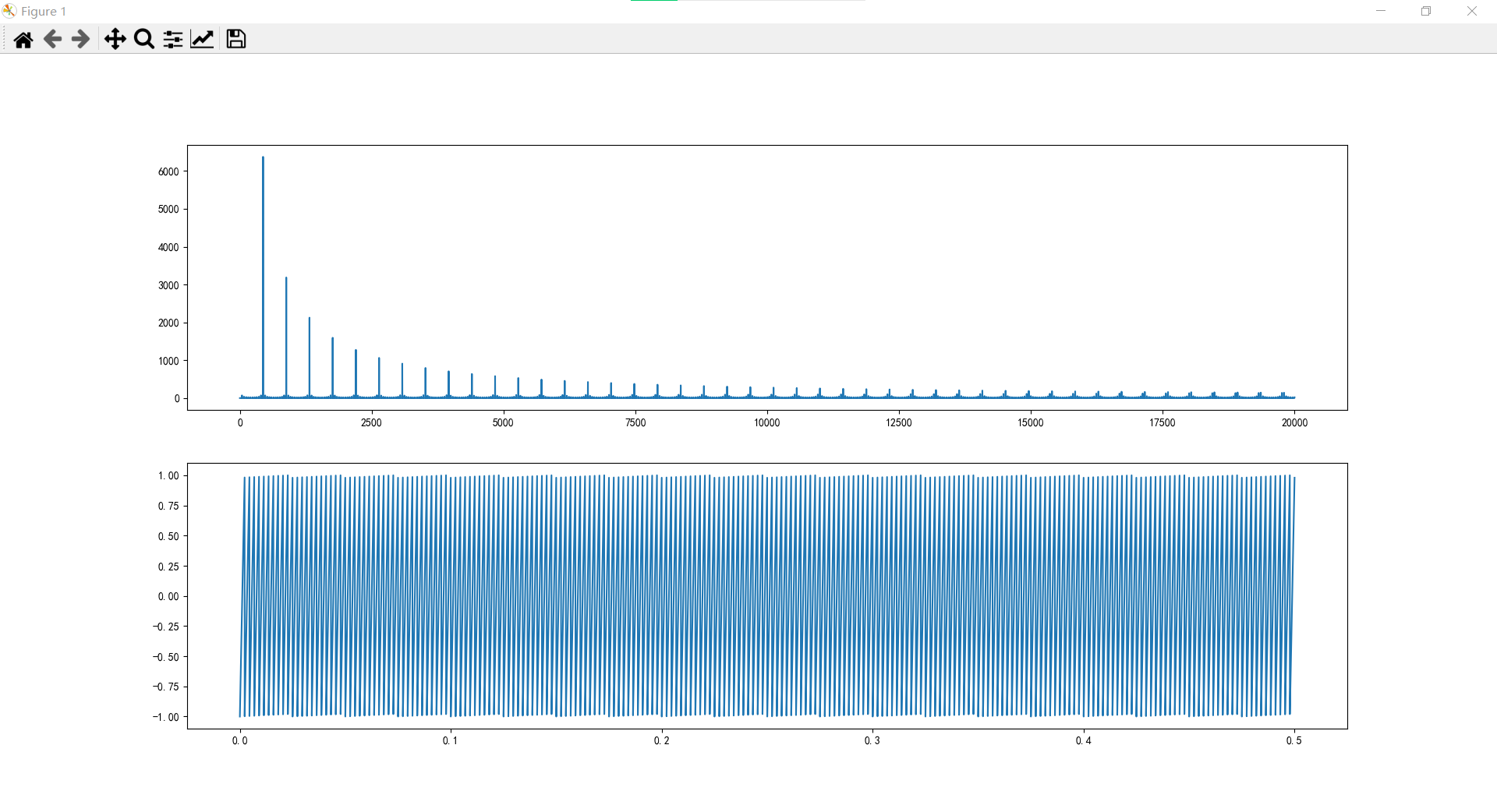
4、创建440hz的三角波，绘制波形图

5、创建spectrum对象并打印频率为零时的振幅和相位。

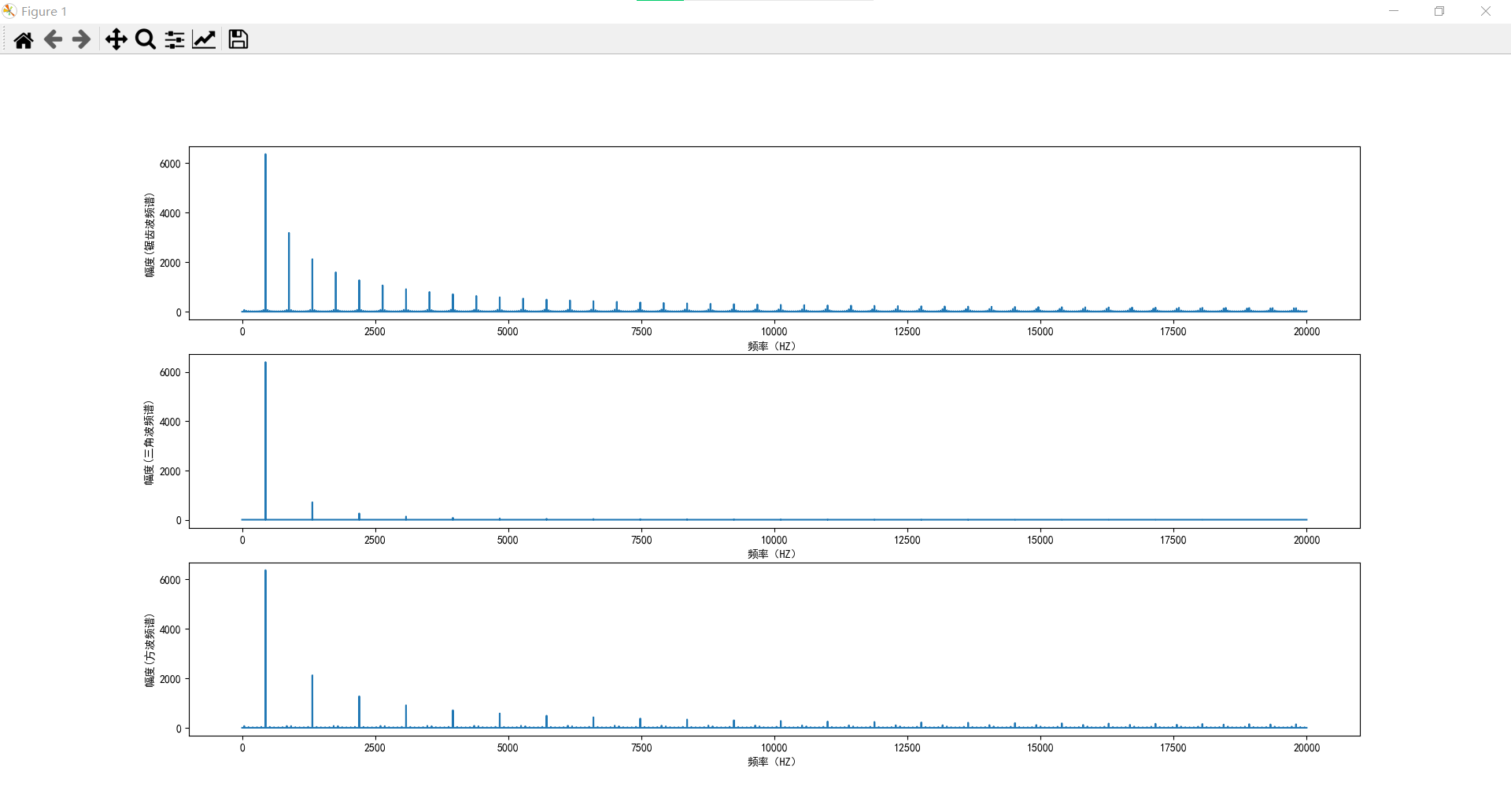
6、设置spectrum,hs[0]=100.观察波形。

**实验结果：**

1. 锯齿信号以及其频谱



2、锯齿波与三角波和方波频谱对比



对比三角波，锯齿波的衰减更慢并且衰减幅度较低，对比方波，锯齿波的衰减相似，衰减幅度相似，但方波没有偶次谐波，三角波也没有。

**程序：**

from thinkdsp import Sinusoid

from thinkdsp import normalize, unbias

from thinkdsp import TriangleSignal

from thinkdsp import decorate

from thinkdsp import SquareSignal

import os

import numpy as np

import matplotlib.pyplot as plt

#三角波

signal = TriangleSignal(amp=0.79)

duration = signal.period\*3

segment = signal.make\_wave(duration, framerate=40000)

#三角波频谱  采样40000hz

spectrum = signal.make\_wave(duration=0.5, framerate=40000)

#方波

square = SquareSignal(amp=0.5).make\_wave(duration=0.5, framerate=40000)

square.make\_spectrum().plot()

#生成锯齿波

class SawtoothSignal(Sinusoid):

    def evaluate(self, ts):

        cycles = self.freq \* ts + self.offset / np.pi / 2

        frac, \_ = np.modf(cycles)

        ys = normalize(unbias(frac), self.amp)

        return ys

#锯齿波频谱

sawtooth = SawtoothSignal().make\_wave(duration=0.5, framerate=40000)

#正常显示中文

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

plt.rcParams['axes.unicode\_minus']=False

plt.subplot(311)

sawtooth.make\_spectrum().plot()

plt.ylabel('幅度(锯齿波频谱)')

plt.xlabel('频率（HZ）')

plt.subplot(312)

spectrum.make\_spectrum().plot()

plt.ylabel('幅度(三角波频谱)')

plt.xlabel('频率（HZ）')

plt.subplot(313)

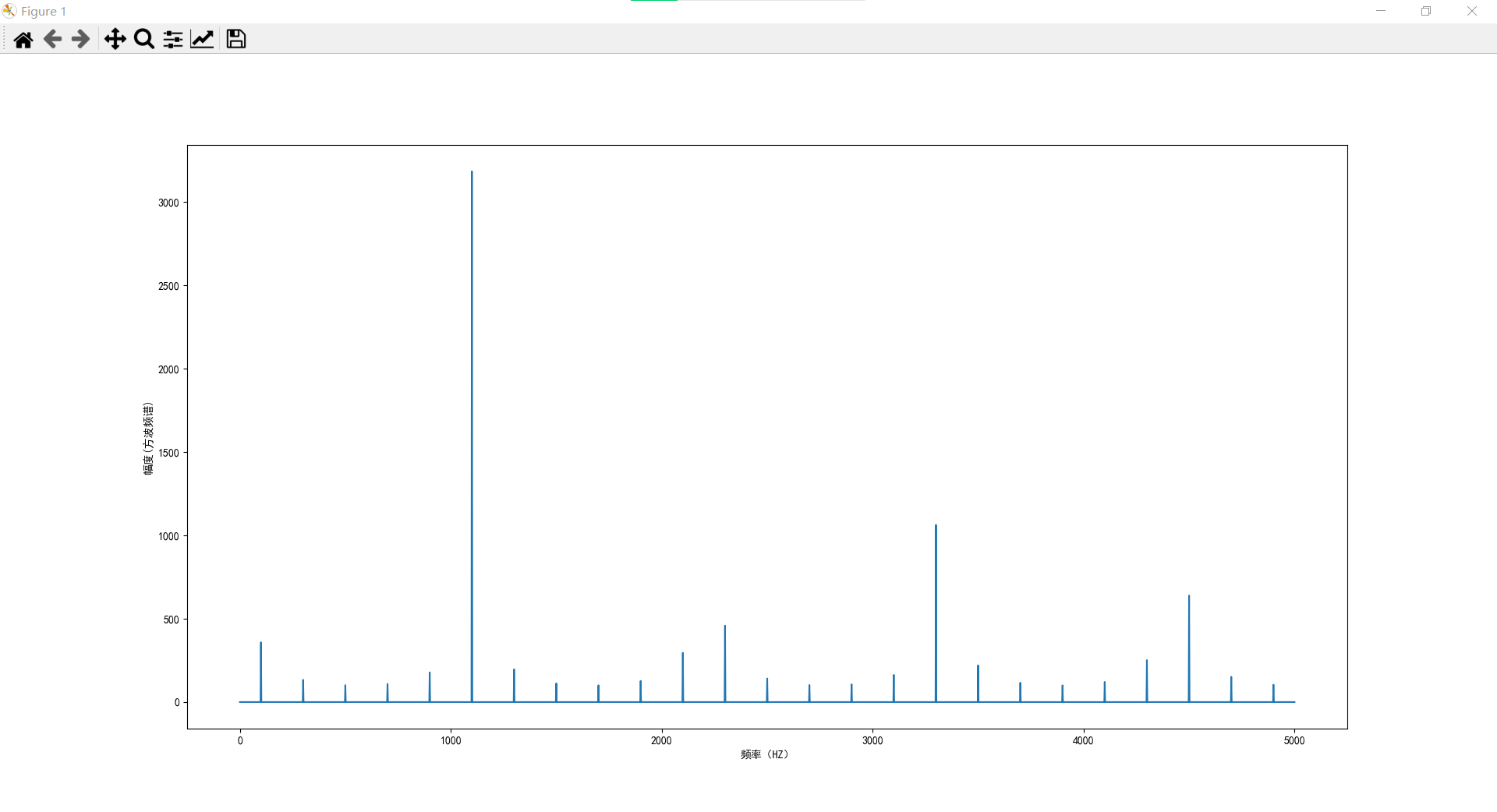
square.make\_spectrum().plot()

plt.ylabel('幅度(方波频谱)')

plt.xlabel('频率（HZ）')

plt.show()

3、创建1100hz的方波信号，绘制频谱



方波二次谐波在3300hz处，三次谐波在5500hz处，因为频谱以5000hz折叠，所以三次谐波应该在4500hz处，从而造成混叠，四次谐波在7700hz处，折叠后出现在2300hz处，观察频谱可发现此处有混叠。

听不出太大区别。

from thinkdsp import Sinusoid

from thinkdsp import normalize, unbias

from thinkdsp import TriangleSignal

from thinkdsp import decorate

from thinkdsp import SquareSignal

import os

import numpy as np

import matplotlib.pyplot as plt

#方波频率1100HZ   采样10000HZ   波长0.5秒

square = SquareSignal(1100).make\_wave(duration=0.5, framerate=10000)

square.write(filename='output2-3.wav')

#显示中文

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

plt.rcParams['axes.unicode\_minus']=False

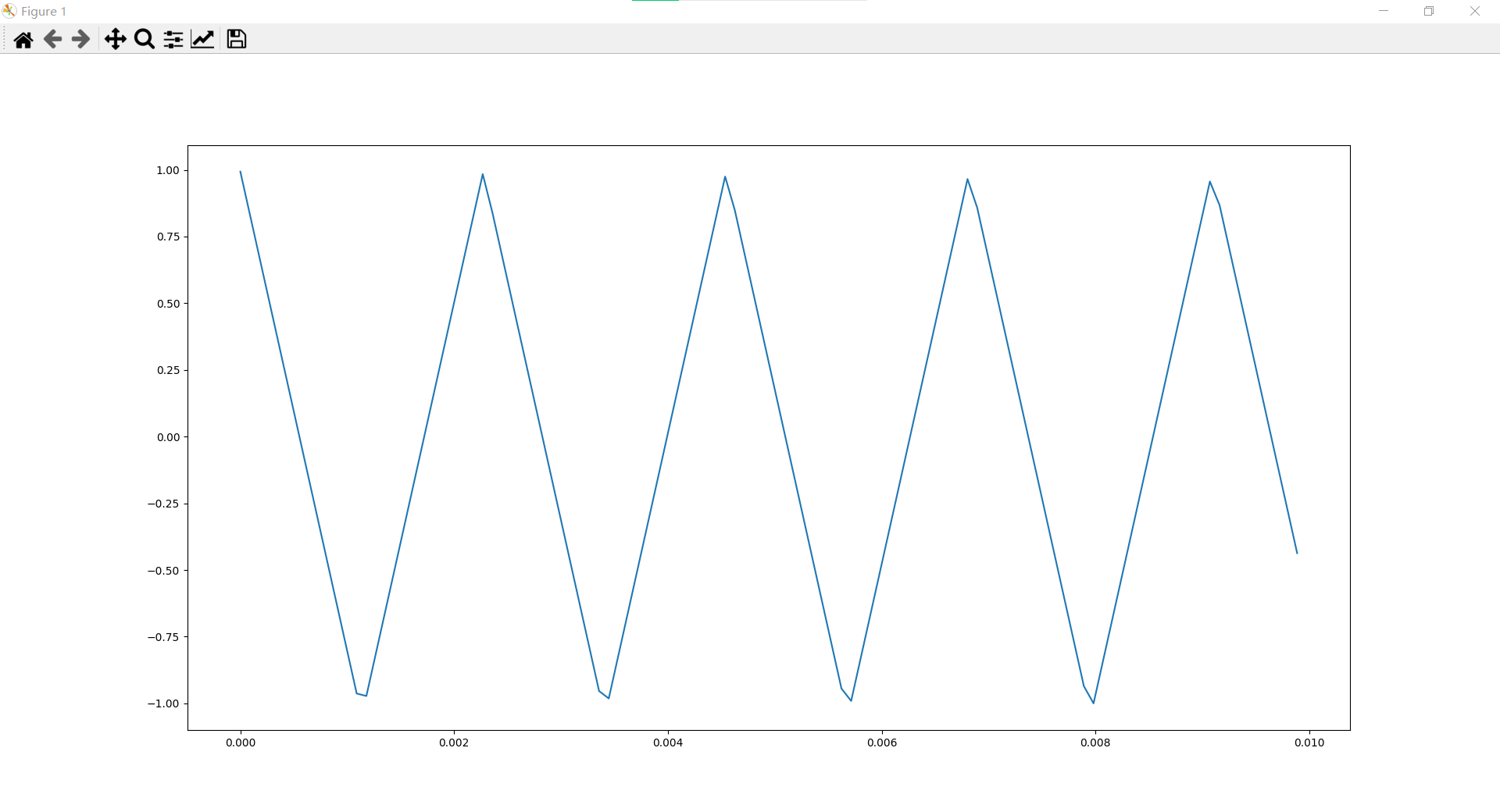
square.make\_spectrum().plot()

plt.ylabel('幅度(方波频谱)')

plt.xlabel('频率（HZ）')

plt.show()

4、创建440hz的三角波，绘制波形图

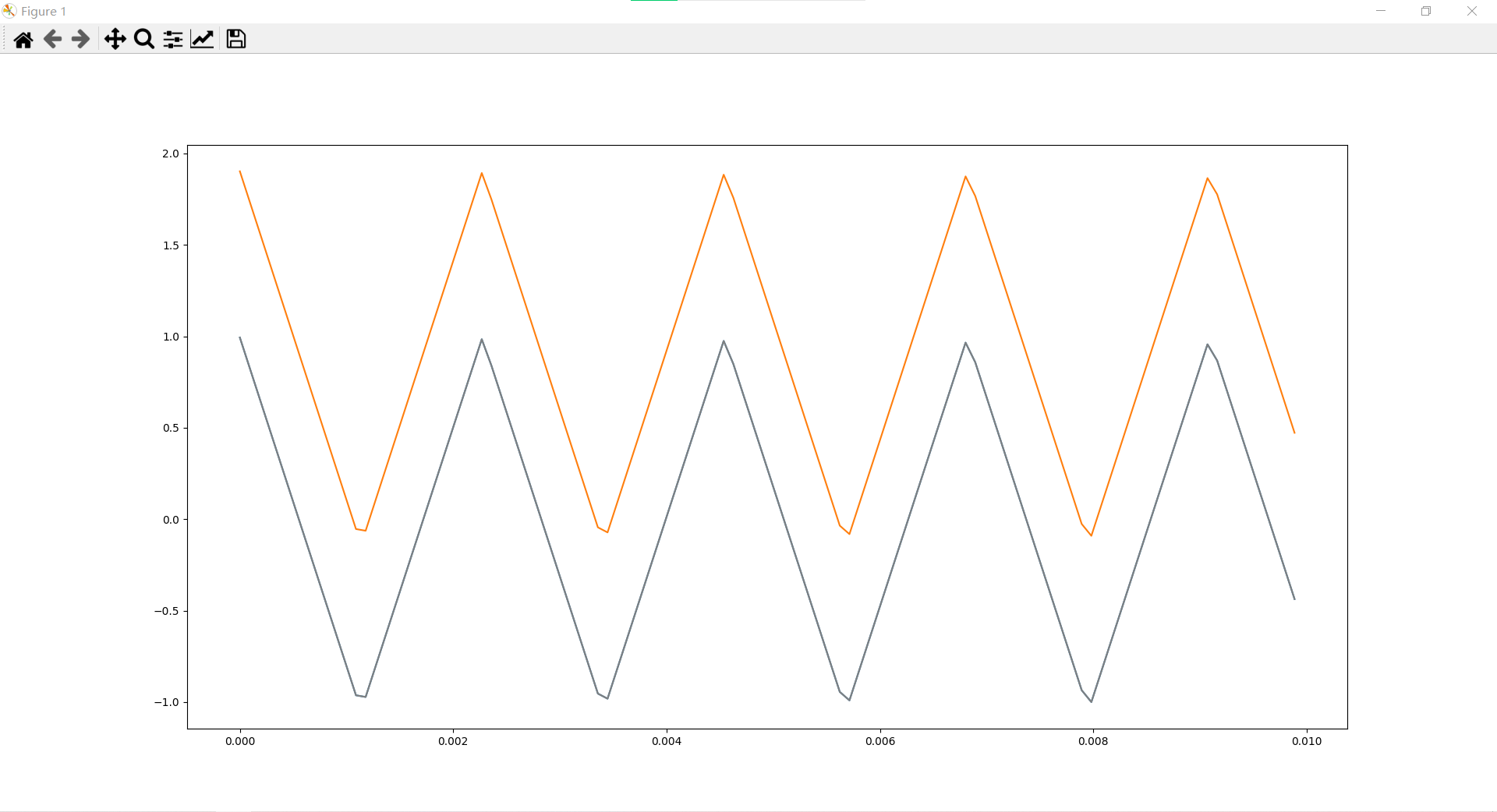


5、创建spectrum对象并打印频率为零时的振幅和相位。



振幅接近零，相位为零

6、设置spectrum,hs[0]=100.观察波形。



在频率为0处的分量设为100，等于叠加了一个0.01s\*100v的直流量

from thinkdsp import Sinusoid

from thinkdsp import normalize, unbias

from thinkdsp import TriangleSignal

from thinkdsp import decorate

from thinkdsp import SquareSignal

import os

import numpy as np

import matplotlib.pyplot as plt

#三角波  频率440HZ  波长0.01秒

triangle = TriangleSignal(440).make\_wave(duration=0.01)

triangle.plot()

#  振幅为2v  相位为0

spectrum = triangle.make\_spectrum()

print(spectrum.hs[0])  #频率分量0

#  振幅为2v  垂直偏移1v

spectrum.hs[0] = 100   #在频率为0处加上100分量  等于叠加一个100\*（duration=0.01）==1v的直流信号

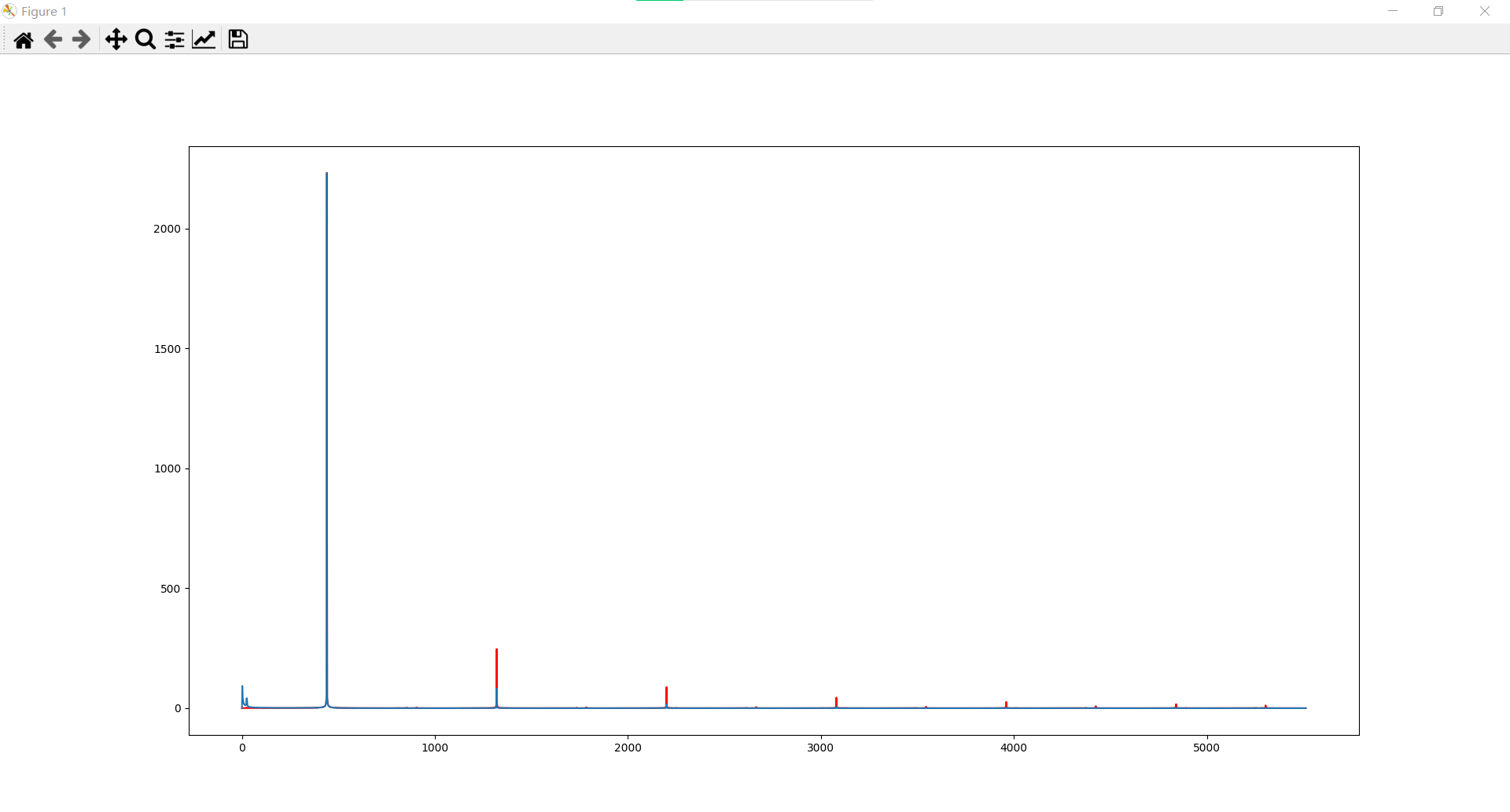
triangle.plot(color='gray')

spectrum.make\_wave().plot()

plt.show()

7、写一个以spectrum为参数的函数，将hs的各个元素除以对应频率fs。

用三角波验证修改前的spectrum与修改后的spectrum波形差异。



红色为修改前，蓝色为修改后，发现谐波分量明显减小。

from thinkdsp import Sinusoid

from thinkdsp import normalize, unbias

from thinkdsp import TriangleSignal

from thinkdsp import decorate

from thinkdsp import SquareSignal

import os

import numpy as np

import matplotlib.pyplot as plt

def filter\_spectrum(spectrum):

    spectrum.hs[1:] /= spectrum.fs[1:]

    spectrum.hs[0] = 0

#三角波  频率440HZ  波长0.5秒

wave = TriangleSignal(freq=440).make\_wave(duration=0.5)

spectrum = wave.make\_spectrum()

spectrum.plot(high=10000, color='red')

filter\_spectrum(spectrum)

spectrum.scale(440)

spectrum.plot(high=10000)

filtered = spectrum.make\_wave()

filtered.make\_audio()

filtered.write(filename='output2-5.wav')

plt.show()