****

数字信号处理实验报告

专业：电子信息（专升本）

班别： 电子信息（专升本）201

姓名： 黄子介

学号： 3200442035

**浙大宁波理工学院实验报告**

**姓名 黄子介 学号 3200442035 学院** 信息科学与工程学院

**专业 电子信息 班级 电子信息（专升本）201 课程** 数字信号处理

**实验时间 2021-6-23 实验地点 SF405**   **指导教师 王一刚**

**实验五**

**实验目的：** 非周期信号的波形及频谱绘制

**实验内容：**

3-2：写一个名为SawtoothChirp的类，以扩展Chirp的功能，重写evaluate从而能产生锯齿波，其频率线性的增加或降低

3-3：创建一个锯齿波啁啾，频率从2500HZ扫到3000Hz，然后使用他构建一个波形，时长1s，帧率20kHz。并绘制频谱

3-4：寻找一个“滑奏”的录音并绘制前几秒的频谱

**实验结果：**

**3-2：**

import numpy as np

import matplotlib.pyplot as plt

from thinkdsp import decorate

from thinkdsp import Chirp

from thinkdsp import normalize, unbias

PI2 = 2 \* np.pi

class SawtoothChirp(Chirp):

    def evaluate(self, ts):

        freqs = np.linspace(self.start, self.end, len(ts))

        dts = np.diff(ts, prepend=0)

        dphis = PI2 \* freqs \* dts

        phases = np.cumsum(dphis)

        cycles = phases / PI2

        frac, \_ = np.modf(cycles)

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

        return ys

signal = SawtoothChirp(start=220, end=880)

wave = signal.make\_wave(duration=1, framerate=4000)

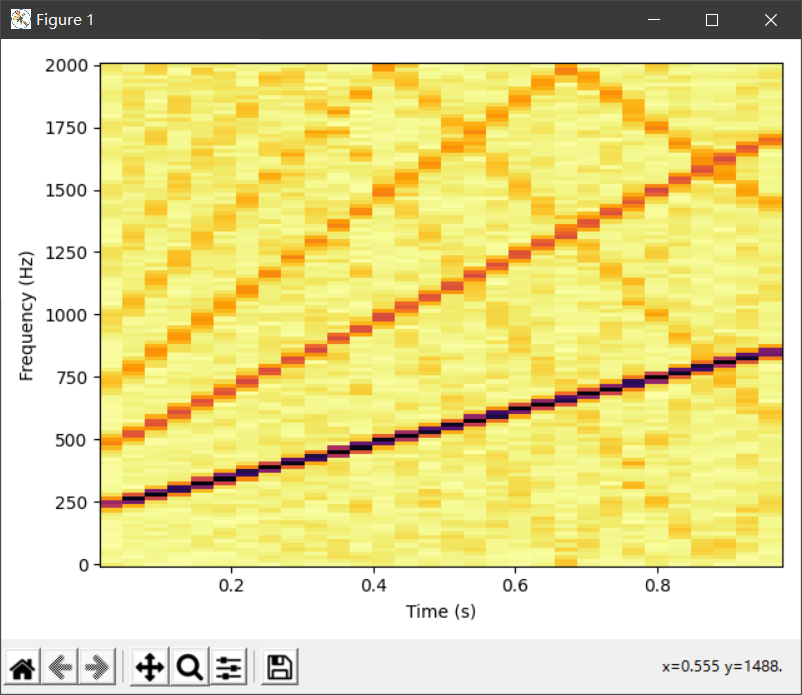
wave.apodize()

sp = wave.make\_spectrogram(256)

sp.plot()

decorate(xlabel='Time (s)', ylabel='Frequency (Hz)')

plt.show()



3-3：

import numpy as np

import matplotlib.pyplot as plt

from thinkdsp import decorate

from thinkdsp import Chirp

from thinkdsp import normalize, unbias

PI2 = 2 \* np.pi

class SawtoothChirp(Chirp):

    def evaluate(self, ts):

        freqs = np.linspace(self.start, self.end, len(ts))

        dts = np.diff(ts, prepend=0)

        dphis = PI2 \* freqs \* dts

        phases = np.cumsum(dphis)

        cycles = phases / PI2

        frac, \_ = np.modf(cycles)

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

        return ys

signal = SawtoothChirp(start=2500, end=3000)

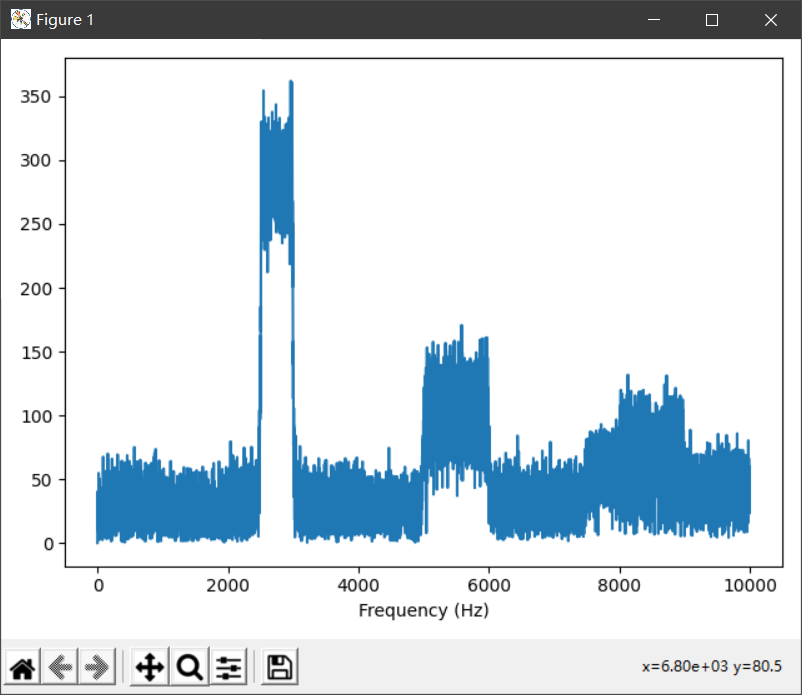
wave = signal.make\_wave(duration=1, framerate=20000)

wave.make\_audio()

wave.make\_spectrum().plot()

decorate(xlabel='Frequency (Hz)')

plt.show()



3-4：

import numpy as np

import matplotlib.pyplot as plt

from thinkdsp import decorate,read\_wave

from thinkdsp import SinSignal

wave=read\_wave('1.wav')

wave.make\_spectrogram(512).plot(high=5000)

decorate(xlabel='Time (s)', ylabel='Frequency (Hz)')

plt.show()

