****

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： 王雯倩

学号： 3200442003

**浙大宁波理工学院实验报告**

**姓名 王雯倩 学号 3200442003 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 2021-7-5 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、编写一个名为SawglobChirp的类，它扩展Chirp并重写计算，生成频率线性增加(或减少)的锯齿波形。

2、发出锯齿啁啾，从2500赫兹到3000赫兹，然后产生一个持续时间为1s和频率20KHZ的波,绘制频谱图。

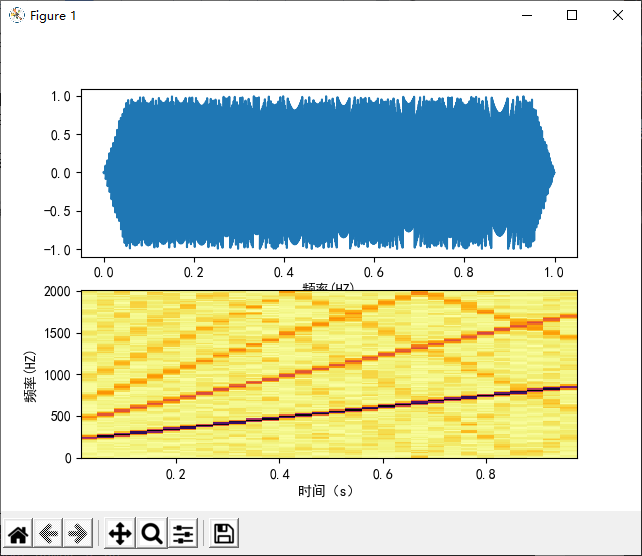
3、录制一个滑奏 “glissando”，并绘制出它的谱图。

4、编写一个名为TromboneGliss的类，它扩展Chirp并提供计算。制作一个模拟长号滑奏的波形，模拟从C3到F3，再到C3。C3为262 Hz，F3为349 Hz。

5、上传github

**实验结果：**

1、频率线性增加的锯齿波的波形与光谱



在一个相对较低的帧速率可以听到他们的背景嘶嘶声。如果你提高帧速率，它们就会消失。

import os

import numpy as np

import matplotlib.pyplot as plt

from thinkdsp import SinSignal

from thinkdsp import Chirp

from thinkdsp import normalize, unbias

PI2=2\*np.pi

class SawtoothChirp(Chirp):

    """Represents a sawtooth signal with varying frequency."""

    def evaluate(self,ts):

        """Helper function that evaluates the signal.

        ts: float array of times

        """

        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)

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

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

plt.subplot(211)

wave.plot()

plt.xlabel('频率(HZ)')

plt.subplot(212)

sp.plot()

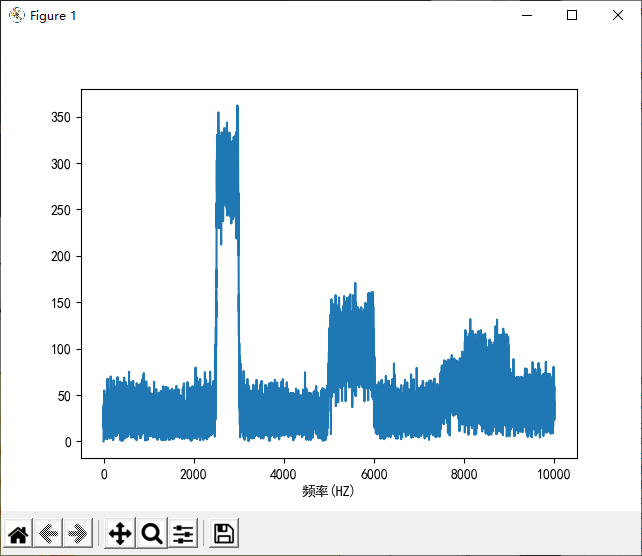
plt.ylabel('频率(HZ)')

plt.xlabel('时间（s）')

wave.write(filename='output3-2.wav')

plt.show()

2、发出锯齿啁啾，从2500赫兹到3000赫兹，然后产生一个持续时间为1s和频率20KHZ的波,绘制频谱图。



import os

import numpy as np

import matplotlib.pyplot as plt

from thinkdsp import SinSignal

from thinkdsp import Chirp

from thinkdsp import normalize,unbias

PI2=2\*np.pi

class SawtoothChirp(Chirp):

    """Represents a sawtooth signal with varying frequency."""

    def evaluate(self,ts):

        """Helper function that evaluates the signal.

        ts: float array of times

        """

        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)

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

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

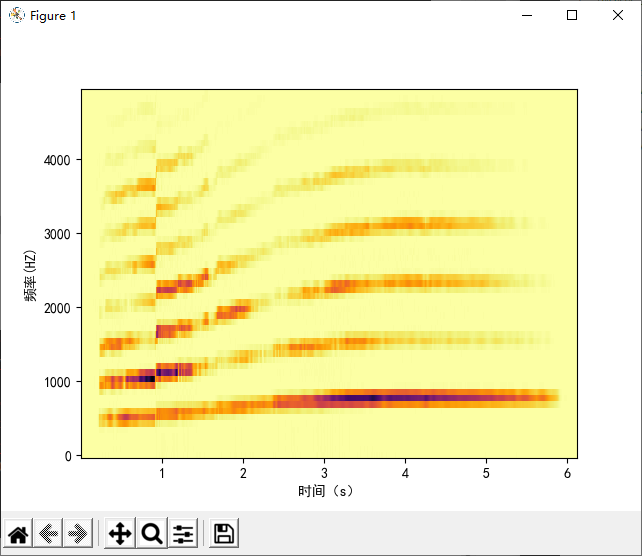
wave.make\_spectrum().plot()

plt.xlabel('频率(HZ)')

wave.write(filename='output3-3.wav')

plt.show()

3、录制一个滑奏 “glissando”，并绘制出它的谱图。



import os

import numpy as np

import matplotlib.pyplot as plt

from thinkdsp import read\_wave

wave=read\_wave('72475\_\_rockwehrmann\_\_glissup02.wav')

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

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

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

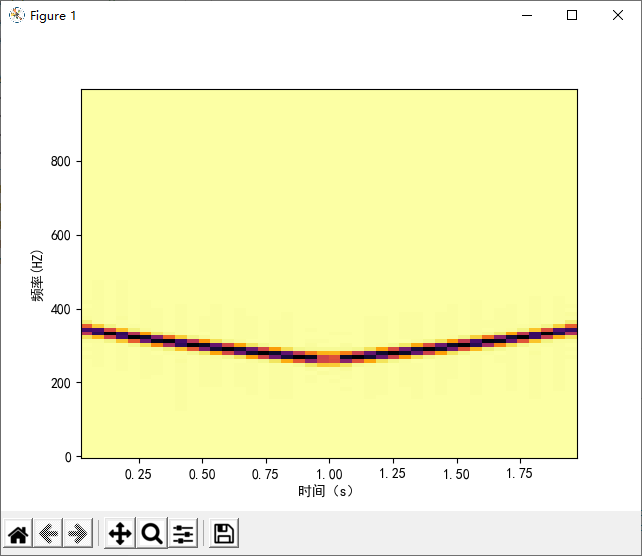
plt.ylabel('频率(HZ)')

plt.xlabel('时间（s）')

wave.write(filename='output3-4.wav')

plt.show()

4、编写一个名为TromboneGliss的类，它扩展Chirp并提供计算。制作一个模拟长号滑奏的波形，模拟从C3到F3，再到C3。C3为262 Hz，F3为349 Hz。



import os

import numpy as np

import matplotlib.pyplot as plt

from thinkdsp import SinSignal

from thinkdsp import Chirp

from thinkdsp import normalize, unbias

PI2 = 2 \* np.pi

class TromboneGliss(Chirp):

    """Represents a trombone-like signal with varying frequency."""

    def evaluate(self, ts):

        """Evaluates the signal at the given times.

        ts: float array of times

        returns: float wave array

        """

        l1,l2=1.0/self.start,1.0/self.end

        lengths = np.linspace(l1, l2, len(ts))

        freqs = 1 / lengths

        dts = np.diff(ts, prepend=0)

        dphis = PI2 \* freqs \* dts

        phases = np.cumsum(dphis)

        ys = self.amp \* np.cos(phases)

        return ys

low=262

high=349

signal=TromboneGliss(high,low)

wave1=signal.make\_wave(duration=1)

wave1.apodize()

signal=TromboneGliss(low,high)

wave2=signal.make\_wave(duration=1)

wave2.apodize()

wave=wave1|wave2

sp=wave.make\_spectrogram(1024)

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

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

sp.plot(high=1000)

plt.ylabel('频率(HZ)')

plt.xlabel('时间（s）')

wave.write(filename='output3-5.wav')

plt.show()

5、上传github

