**浙大宁波理工学院实验报告**

**姓名 张锡永 学号 3200442033 学院**  信息学院

**专业 电子信息工程 班级 电信专升本201班 课程 数字信号处理**

**实验时间 2021-06-30 实验地点 SL 405**   **指导教师 王一刚**

**实验五**

**实验目的：**

1. chirp的使用
2. 观察chirp信号的频谱

**实验内容：**

**利用python及thinkdsp库编写程序**

**实验结果：**

from thinkdsp import Chirp

from thinkdsp import normalize, unbias

from thinkdsp import decorate

import numpy as np

import matplotlib.pyplot as plt

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()

wave.make\_audio()

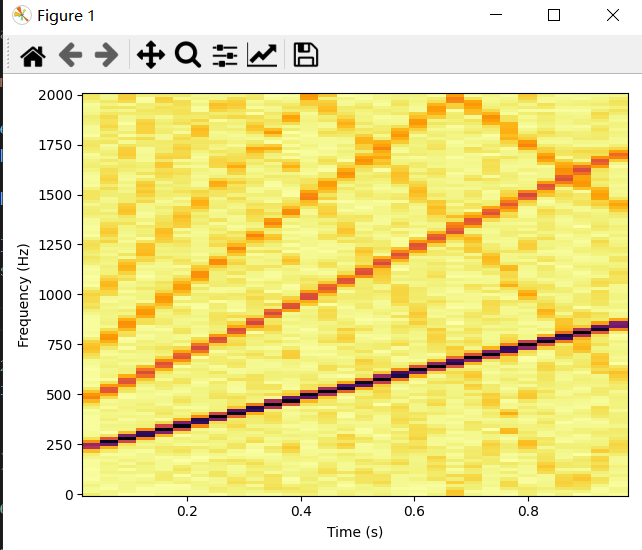
wave.write(filename="3-2.wav")

sp = wave.make\_spectrogram(256)

sp.plot()

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

plt.show()



看到锯齿谐波从折叠频率反弹听到他们作为背景声。而当提高帧速率它们会消失

3-3

from thinkdsp import Chirp

from thinkdsp import normalize, unbias

from thinkdsp import decorate

import numpy as np

import matplotlib.pyplot as plt

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)

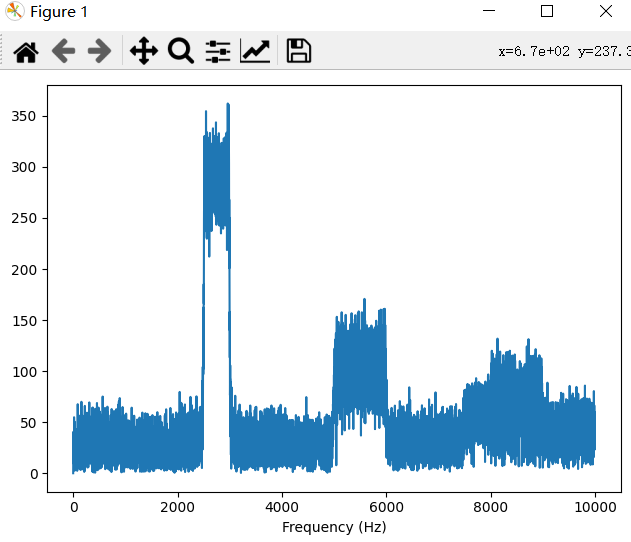
wave.make\_audio()

wave.write(filename="3-3.wav")

wave.make\_spectrum().plot()

decorate(xlabel='Frequency (Hz)')

plt.show()



3-4

from thinkdsp import Chirp

from thinkdsp import normalize, unbias

from thinkdsp import decorate

from thinkdsp import read\_wave

import numpy as np

import matplotlib.pyplot as plt

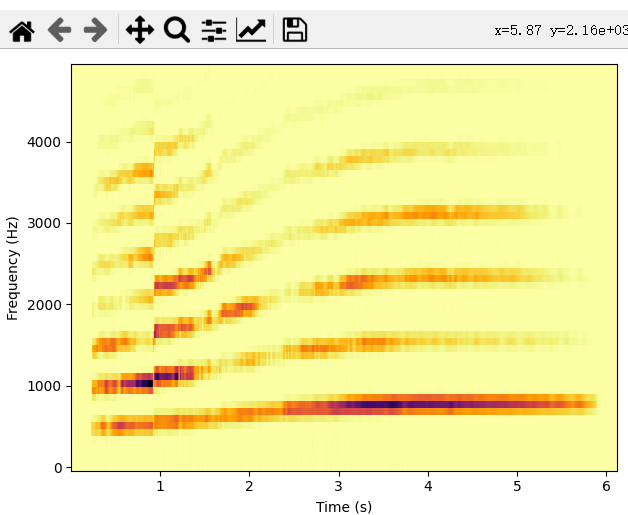
wave = read\_wave('72475\_\_rockwehrmann\_\_glissup02.wav')

wave.make\_audio()

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

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

plt.show()



3-5

from thinkdsp import Chirp

from thinkdsp import normalize, unbias

from thinkdsp import decorate

import numpy as np

import matplotlib.pyplot as plt

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()

wave1.make\_audio()

wave1.write(filename='3-5.1.wav')

signal = TromboneGliss(low, high)

wave2 = signal.make\_wave(duration=1)

wave2.apodize()

wave2.make\_audio()

wave2.write(filename='3-5.2.wav')

wave = wave1 | wave2

wave.make\_audio()

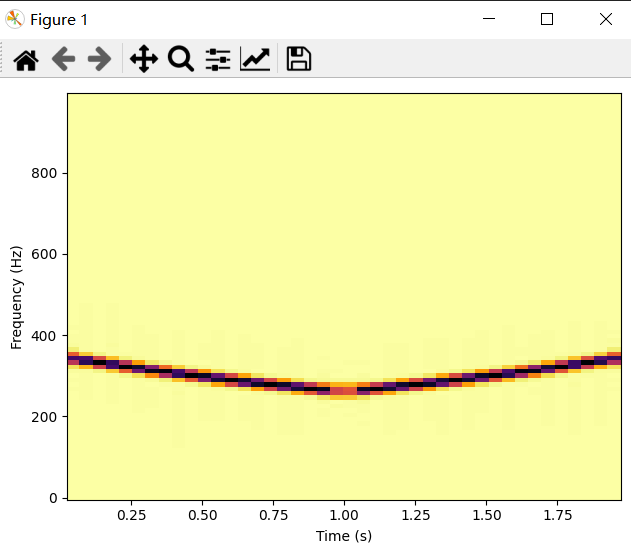
wave.write(filename='3-5.3.wav')

sp = wave.make\_spectrogram(1024)

sp.plot(high=1000)

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

plt.show()



上传github