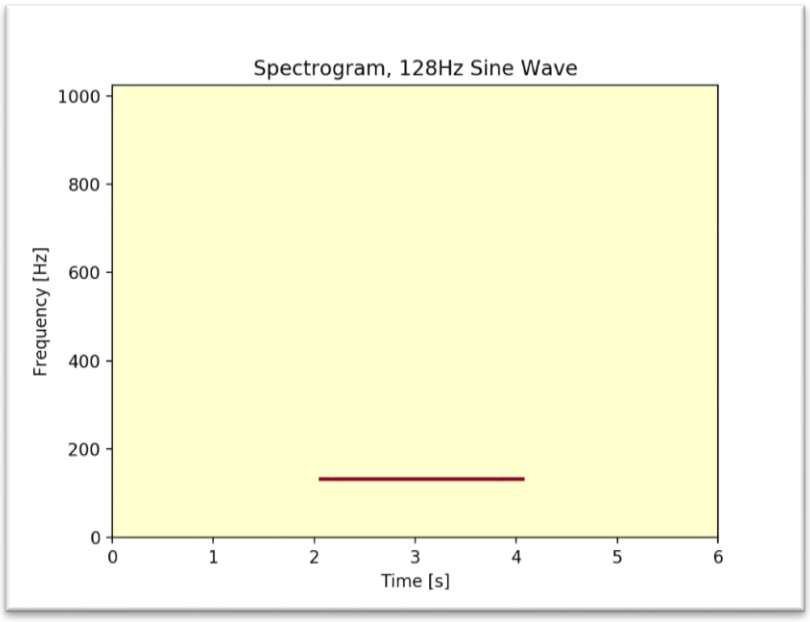
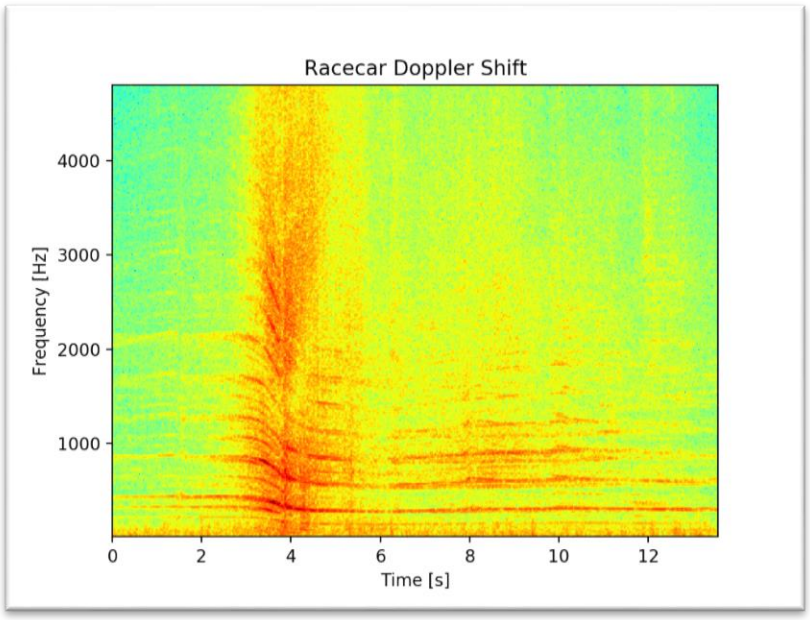
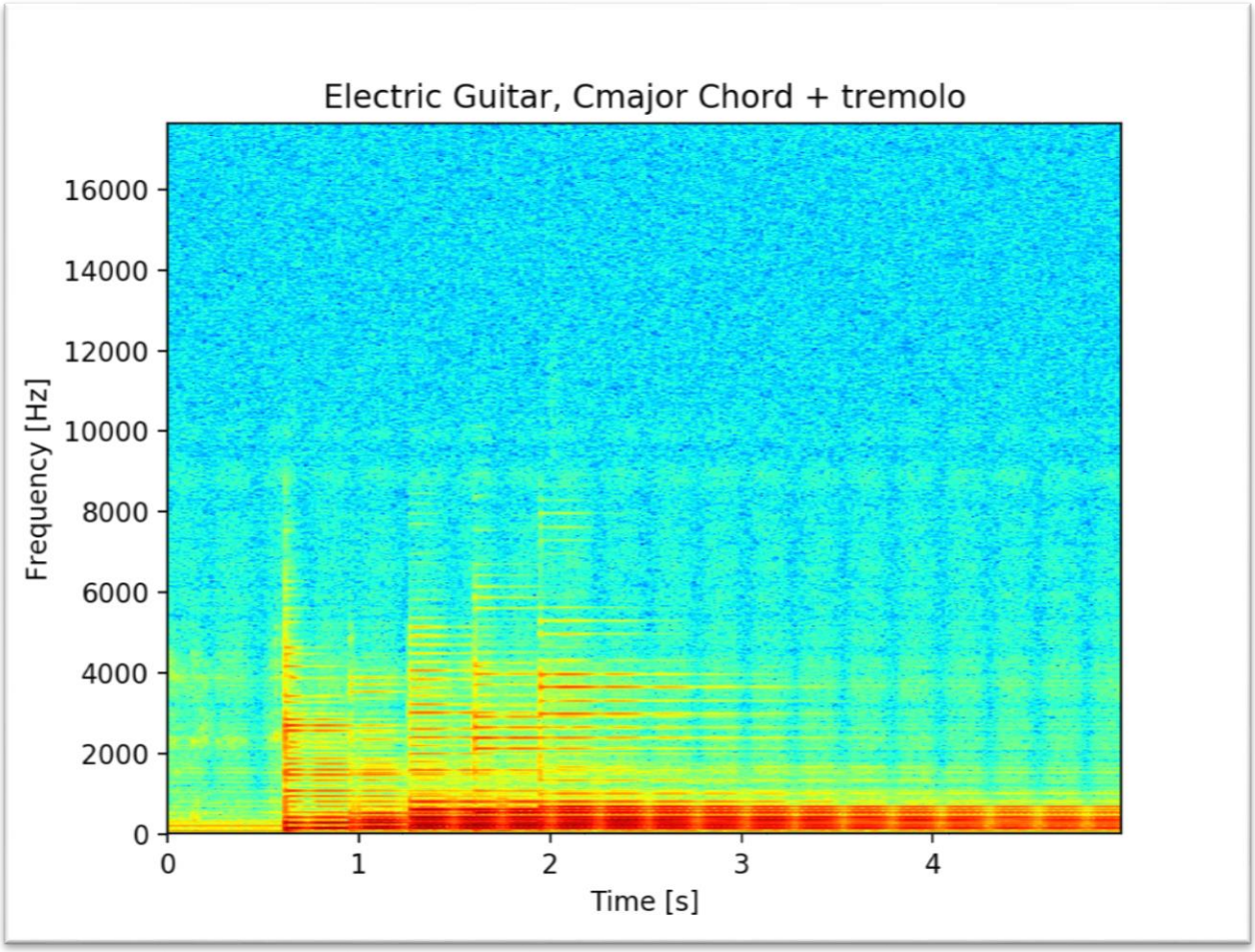
Mohammed Alshehri
ACS597 Signal Analysis
Spectrogram
9/18/2018



We can see a slight time-shift in the reading, which I am assuming is due to the time-bin resolution being somewhat reduced. Increasing the recording length edges us ever so closer to the actual start time for the wave.



Here, I stayed with an overlap of 75%, but I went with an N of 1024 samples, as I felt that gave me a reasonable amount of frequency and time resolution.



This is a spectrogram of a ~5 seconds recording of me playing a C Major chord arpeggio through a tremolo effect. This effect modulates the volume of the guitar signal, which can be visible from the periodic vertical blue lines. What surprised me the most is how little information there is beyond 10kHz.

Appendix: Python Code

"Spectrogram":

```
1.  import numpy as np
2.  import matplotlib.pyplot as plt
3.  from scipy.io import wavfile
4.  from numpy import sin, pi, shape
5.  import sigA
6.  import sounddevice as sd
7.
8.  foldername = "/Users/macbookpro/PycharmProjects/PSUACS/ACS597_SigAnalysis/"
9.  #foldername = "C:/Users/alshehrj/PycharmProjects/PSUACS/ACS597_SigAnalysis/"
10.
11. def spectroArray(x_time, fs, sliceLength, sync=0,overlap=0,winType="uniform"):
12.     if overlap >= 1.0:
13.         sys.exit("overlap >= 1")
14.     Gxx = np.zeros((1,sliceLength/2))
15.
16.     i = 0
17.     while True:
18.         n = i * (int(sliceLength*(1-overlap))+sync)
19.         sliceEnd = n + sliceLength - 1
20.
21.         if sliceEnd >= len(x_time)-1:
22.             break
23.
24.         if i == 0:
25.             Gxx[0,]= sigA.ssSpec(x_time[n:sliceEnd], fs,winType)
26.         else:
27.             Gxx = np.concatenate((Gxx,[sigA.ssSpec(x_time[n:sliceEnd], fs,winType)]),ax
    is=0)
28.         i += 1
29.
30.     #### Gxx Avg
31.     GxxAvg = np.mean(Gxx, axis=0)
32.     freqAvg = sigA.freqVec(sliceLength, fs)[:int(sliceLength/2)]
33.     _, delF_Avg, _ = sigA.param(sliceLength, fs, show=False)
34.
35.     print "Stepping out of Gxx"
36.     print np.shape(Gxx)
37.     return GxxAvg, freqAvg, delF_Avg, Gxx
38.
39.
40. def spectrogram(x_time, fs, sliceLength, sync=0, overlap=0,color="jet", dB=True, winTyp
    e="uniform", scale=True):
41.     N = len(x_time)
42.     Nslices = int(N / sliceLength)
43.     T = Nslices * sliceLength / float(fs)
44.     print "T: " + str(T)
45.
46.     _, freqAvg, _, Gxx = spectroArray(x_time=x_time, fs=fs, sliceLength=sliceLength, sy
    nc=sync, overlap=overlap, winType=winType)
47.
48.     GxxRef = 1.0                              # V^2/Hz
49.     Gxx_dB = 10 * np.log10(Gxx / GxxRef)
50.
51.     ext = [0, T, 0, fs / 2]
52.
53.     if dB:
```

```
        plt.imshow(Gxx_dB.T, aspect="auto", origin="lower", cmap=color, extent=ext)
    else:
        plt.imshow(Gxx.T, aspect="auto", origin="lower",cmap=color, extent=ext)
    if scale:
        plt.ylim(ext[1] + 1, ext[3] * 0.8)


def main(args):
    sinTest()
    raceCar()
    recording()


def sinTest():
    fs = 2048.0
    T = 6.0
    N=int(fs*T)
    print N

    times = sigA.timeVec(N,fs)
    delT,delF,_= sigA.param(N,fs)

    f = 128

    x_time = np.zeros(N)
    t = sigA.timeVec(N,fs)

    for i in range(N):
        if i*delT < 2.0:
            x_time[i] += 0
        elif i*delT < 4.0:
            x_time[i] += sin(2*pi*f*times[i])
        else:
            x_time[i] += 0

    plt.figure()
    plt.plot(t,x_time)

    plt.figure()

    sliceLength = 256           # Length of single record
    ov = 0                    # Overlap

    spectrogram(x_time,fs,sliceLength,sync=0,dB=False,color="YlOrRd",overlap=ov,winType
="uniform",scale=False)
    plt.xlabel("Time [s]")
    plt.ylabel("Frequency [Hz]")
            plt.title("Spectrogram, 128Hz Sine Wave")
            plt.show()


       def raceCar():
            filename = "T4_C5_3L_dec4a.wav"
            path = foldername+filename
            fs , data = wavfile.read(path)

            Nwav = len(data)
            print data.dtype
            print data
            if data.dtype != np.float32:
                print "Converting from " + str(data.dtype) + " to float32"
```

```
114.                 data = data.astype(np.float32)
115.                 data = data /32768.0
116.
117.             print fs
118.             print data
119.             print Nwav
120.             print Nwav/float(fs)
121.             print 10 * "-"
122.
123.             #t = sigA.timeVec(Nwav, fs)
124.             print np.shape(data)
125.
126.             ov = 0.75
127.             sliceLength = 64
128.
129.             plt.figure()
130.             spectrogram(data,fs,sliceLength,sync=0,dB=True,overlap=ov,winType="hann")
131.             plt.xlabel("Time [s]")
132.             plt.ylabel("Frequency [Hz]")
133.             plt.title("Racecar Doppler Shift")
134.             plt.show()
135.
136.
137.         def recording():
138.             fs = 44100
139.             T = 5
140.             N = fs*T
141.             print N
142.
143.             recArray = sd.rec(frames=N,samplerate=fs,channels=1,blocking=True)
144.
145.             x_time = np.reshape(recArray, (len(recArray),))
146.
147.             t = sigA.timeVec(N,fs)
148.
149.             plt.figure()
150.             plt.plot(t,x_time)
151.
152.             ov = 0.75
153.             sliceLength = 2056
154.
155.             GxxAvg = sigA.ssSpec(x_time=x_time,fs=fs)
156.             FreqAvg = sigA.freqVec(N,fs)
157.
158.             plt.figure()
159.             plt.plot(FreqAvg[:len(GxxAvg)],GxxAvg)
160.
161.             scaled = np.int16(x_time/np.max(np.abs(x_time)) * 32767)
162.             wavfile.write('test.wav', 44100, scaled)
163.
164.             print np.shape(x_time)
165.
166.             plt.figure()
167.             spectrogram(x_time=x_time, fs=fs, sliceLength=sliceLength, sync=0, dB=True,
    overlap=ov, winType="hann",scale=True)
168.             #spectroArray(x_time, fs, sliceLength, sync=0, overlap=ov, winType="hann")
169.
170.             plt.title("Electric Guitar, Cmajor Chord + tremolo")
171.             plt.xlabel("Time [s]")
172.             plt.ylabel("Frequency [Hz]")
173.             plt.show()
```

```
174.
175.
176.       if __name__ == '__main__':
177.           import sys
178.           sys.exit(main(sys.argv))
```

“sigA” Library Methods Used in “Spectrogram”:

```
1.  import numpy as np
2.  import matplotlib.pyplot as plt
3.  from numpy import pi, sin, cos, tan, exp
4.
5.  def param(N,fs,show=True):
6.      delT = 1/float(fs)
7.      delF = float(fs)/float(N)
8.      T = float(N)/float(fs)
9.      if show:
10.         print "N: " + str(N)
11.         print u"\N{GREEK CAPITAL LETTER DELTA}" + "t: " + str(delT)
12.         print u"\N{GREEK CAPITAL LETTER DELTA}" + "f: " + str(delF)
13.         print "fs: " + str(fs)
14.         print "T: " + str(T)
15.         print 10 * "-"
16.     return delT, delF, T
17.
18.
19. def timeVec(N, fs):
20.     delT,_,_ = param(N,fs,False)
21.     t = np.arange(0, N) * delT
22.     return t
23.
24.
25. def freqVec(N, fs):
26.     _, delF, _ = param(N, fs, False)
27.     f = np.arange(0, N) * delF
28.     return f
29.
30. def linSpec(x_time,fs,winType="uniform"):
31.     N = len(x_time)
32.     delT, _, _ = param(N, fs, False)
33.     x_time = x_time*window(winType,N)
34.     return np.fft.fft(x_time)*delT
35.
36. def dsSpec(x_time,fs,winType="uniform"):
37.     N = len(x_time)
38.     _, delF, _ = param(N, fs, False)
39.     lsp = linSpec(x_time,fs,winType)
40.     Sxx = (abs(lsp)**2)*delF
41.     return Sxx
42.
```

```
def ssSpec(x_time,fs,winType="uniform"):
    Sxx = dsSpec(x_time,fs,winType)
    N = len(Sxx)

    Gxx = Sxx[0:(N/2)+1]
    odd = bool(len(Sxx)%2)

    for i in range(len(Gxx)):
        if (i != 0) or (i==(len(Gxx)-1) and odd):
            Gxx[i] = (Gxx[i])*2
    return Gxx

def window(type, N):
    n = np.asfarray(np.arange(N))
    vec = n/N
    win = np.ones(N)
    if type == "uniform":
        win = np.ones(N)
    elif type == "hann":
        win = 1-cos(2*pi*vec)
    elif type == "flat top":
        win = 1-(1.93*cos(2*pi*vec))+(1.29*cos(4*pi*vec))\
              -(0.388*cos(6*pi*vec))+(0.322*cos(8*pi*vec))
    return win
```