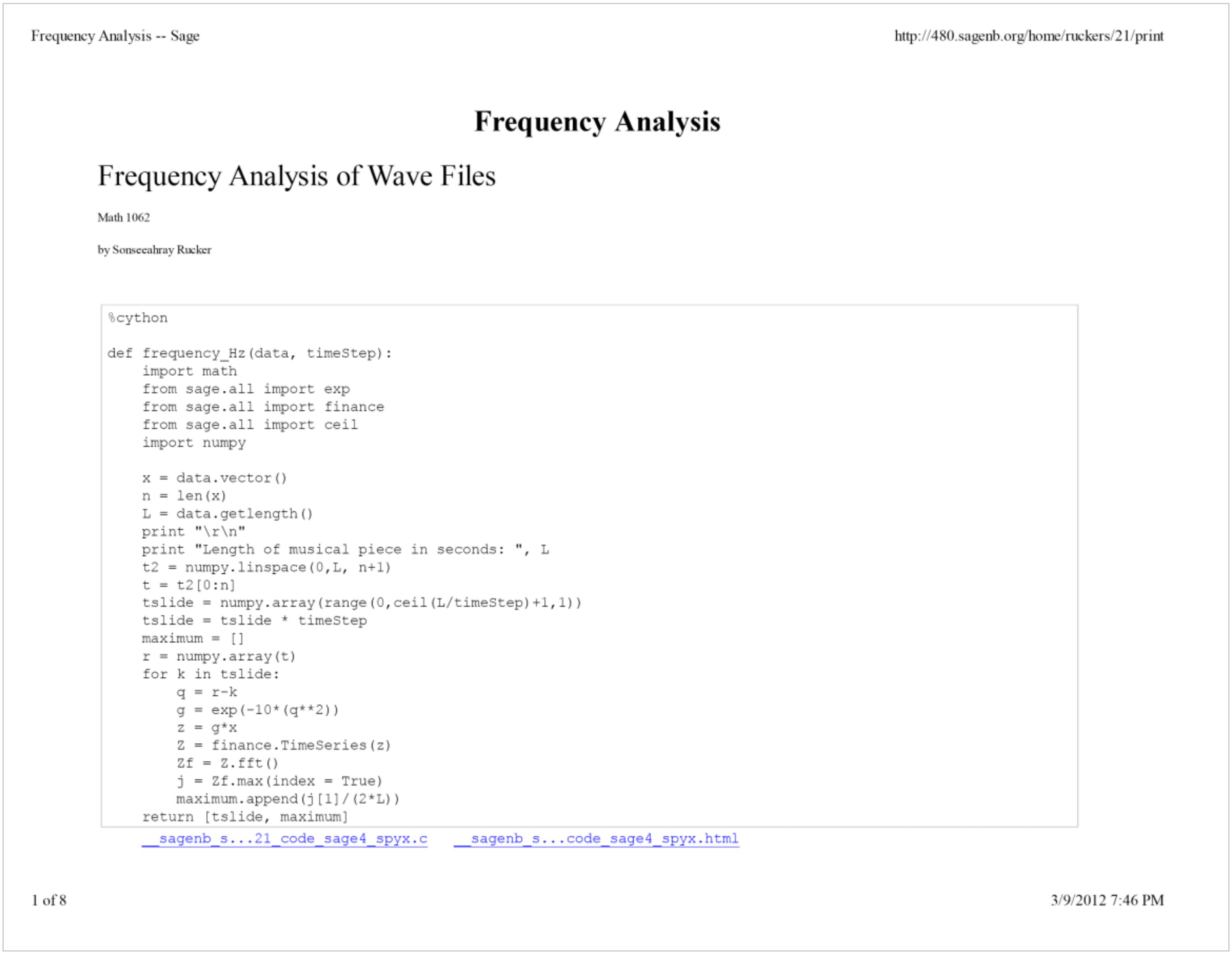


Frequency Analysis

Frequency Analysis of Wave Files

Math 1062

by Sonseeahray Rucker

```
%cython

def frequency_Hz(data, timeStep):
    import math
    from sage.all import exp
    from sage.all import finance
    from sage.all import ceil
    import numpy

    x = data.vector()
    n = len(x)
    L = data.getlength()
    print "\r\n"
    print "Length of musical piece in seconds: ", L
    t2 = numpy.linspace(0,L, n+1)
    t = t2[0:n]
    tslide = numpy.array(range(0,ceil(L/timeStep)+1,1))
    tslide = tslide * timeStep
    maximum = []
    r = numpy.array(t)
    for k in tslide:
        q = r-k
        g = exp(-10*(q**2))
        z = g*x
        Z = finance.TimeSeries(z)
        Zf = Z.fft()
        j = Zf.max(index = True)
        maximum.append(j[1]/(2*L))
    return [tslide, maximum]
```

__sagenb_s...21_code_sage4_spyx.c   __sagenb_s...code_sage4_spyx.html

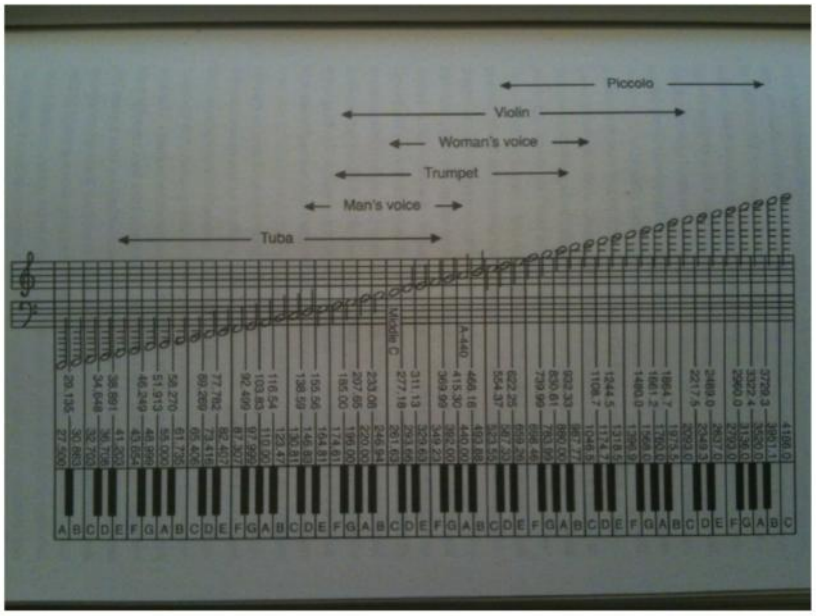
```
#This program analyzes a wave file and returns a plot of the highest frequencies present.  If a wave file with
only one instrument, voice, or signal present is analyzed, this program will return a plot that is equivalent
to the musical score/frequency signature of the piece being analyzed.
#This program is about 10 times slower than Matlab, but it is just as accurate.
#To keep an even time step the final measurement technically oversteps the length of the file, but due to the
nature of the analysis the frequency given at the final time step does correspond to the last frequencies in
the wave file being analyzed.
#This program is designed to work in the notebook and requires music files that are already attached to the
worksheet.
#To change the music file or the graph title parentheses are required around the file/title.


html('<h2>Wave File Analysis for Maximal Frequencies</h2>')

@interact

def Music_Graph(timeStep = input_box(2, label="Time Step (sec)"), Gtitle = input_box('Mary Had a Little Lamb
(Piano)'
, label = "Graph Title"), musicFile = input_box('music1',label= "Music File"), plotSize =
slider(60,120,10,80), auto_update = False):

    html('<h2>Musical Frequency to Note Conversion Chart</h2>')

    html('<img src="chart.png">')
    data = wave(DATA+musicFile)

    u = frequency_Hz(data,timeStep)

    import pylab
    pylab.clf()
    pylab.scatter(u[0],u[1])
    pylab.xlabel('Time (sec)')
    pylab.ylabel('Frequency (Hz)')
    pylab.title(Gtitle)
    pylab.grid(True)
    pylab.savefig('music1.png', dpi=plotSize)
    pylab.clf()
    pylab.xlabel('Time (sec)')
    pylab.ylabel('Frequency (Hz)')
    pylab.title(Gtitle)
    pylab.grid(True)
    pylab.plot(u[0],u[1])
pylab.savefig('music2.png', dpi=plotSize)
pylab.scatter(u[0],u[1])
pylab.savefig('music3.png', dpi=plotSize)
```

Wave File Analysis for Maximal Frequencies

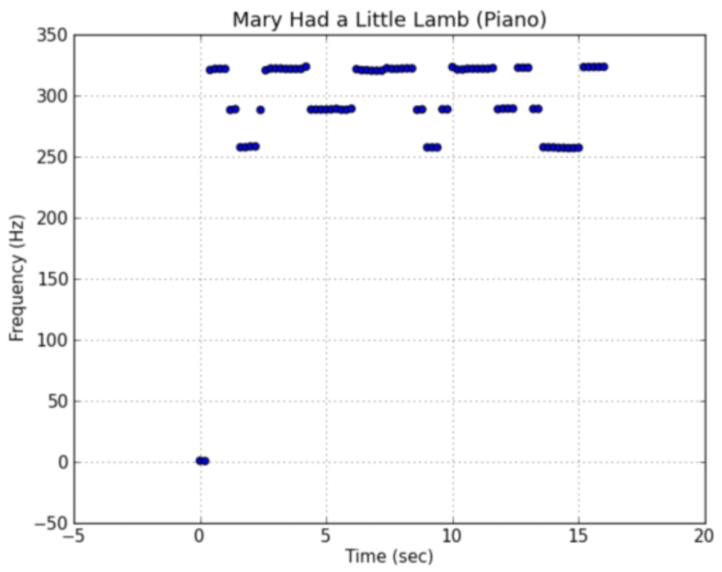
Time Step (sec)	2
Graph Title	Mary Had a Little Lamb (Piano)
Music File	music1
plotSize	80

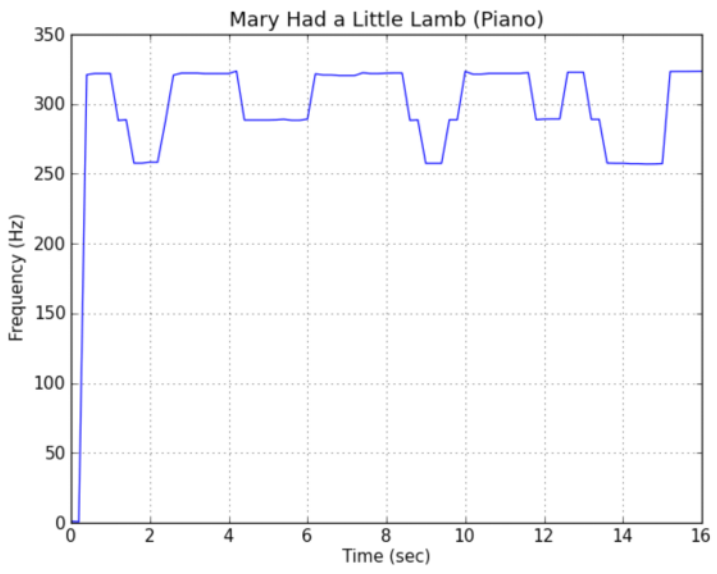
Update

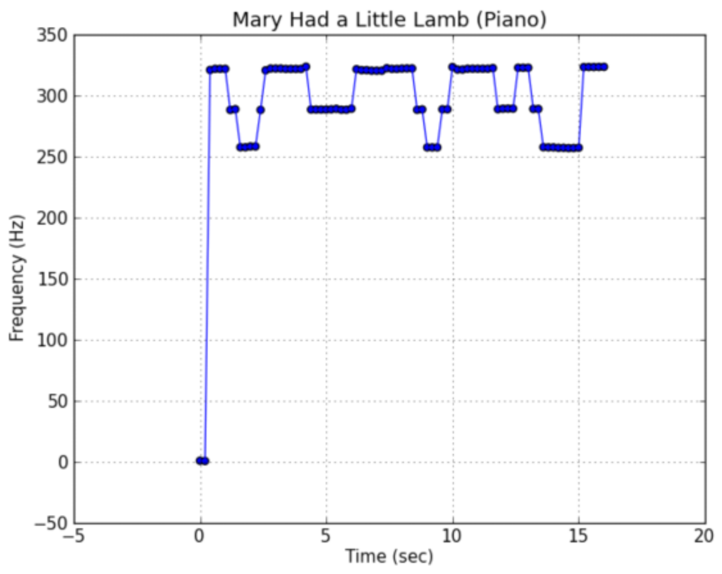
Musical Frequency to Note Conversion Chart



```
Length of musical piece in seconds:  15.9056689342
```

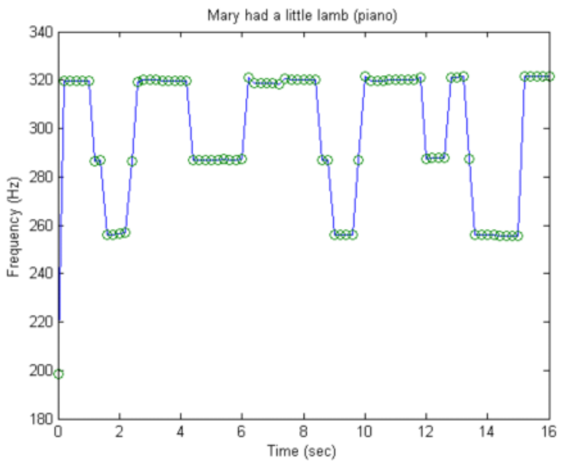

Mary Had a Little Lamb (Piano)
Frequency (Hz)
Time (sec)
350
300
250
200
150
100
50
0
−50
−5
0
5
10
15
20


Mary Had a Little Lamb (Piano)
Frequency (Hz)
Time (sec)
0
50
100
150
200
250
300
350
0
2
4
6
8
10
12
14
16



Comparing the above graphs of the frequency analysis done by Sage to the graph below of Matlab's solution, we can see that although my implementation in Sage is slow it is just as accurate as using Matlab.

```
html('<img src="piano1.png">')
```


Mary had a little lamb (piano)
Frequency (Hz)
Time (sec)
340
320
300
280
260
240
220
200
180
0
2
4
6
8
10
12
14
16