CS170A — Mathematical Models & Methods for Computer Science
HW#3 — Music, Earthquakes, Fake Photos, and Newton
Due: 11:59pm Friday March 14, 2014

D. Stott Parker, Xingyu Ma, Costas Sideris

Please turn in your written answers/outputs in a file called `hw3.pdf` on CourseWeb, and include your source code.

1. **Fourier Music**
The Matlab commands
`[y,MysteryFs] = wavread('mystery.wav'); n = length(y)`
create a vector `y` that contains a sound track (audio sequence) for the first few seconds of a mystery tune. Then as shown in class, if `Fs` is an integer sampling frequency then executing `sound(y, Fs)` will play it at the sampling frequency `Fs`. There are many small questions below; all of them are based on `y`, which is the first of the 2 stereo audio tracks.

Remember the k-th note in the the *equal-tempered scale* has frequency c^k times the frequency of C, where $c = 2^{1/12} = 1.059463\ldots$:

note	frequency (Hz)	frequency/C	interval
Middle C	261.63	$c^0 = 1.000$	
C#	277.18	$c^1 = 1.059$	minor 2nd
D	293.66	$c^2 = 1.122$	major 2nd
D#	311.13	$c^3 = 1.189$	minor 3rd
E	329.63	$c^4 = 1.260$	major 3rd
F	349.23	$c^5 = 1.334$	4th
F#	369.99	$c^6 = 1.414$	diminished 5th
G	392.00	$c^7 = 1.498$	5th
G#	415.30	$c^8 = 1.587$	minor 6th
A	440	$c^9 = 1.682$	major 6th
A#	466.16	$c^{10} = 1.782$	minor 7th
B	493.88	$c^{11} = 1.888$	major 7th
High C	523.25	$c^{12} = 2.000$	octave/8th

The equal-tempered scale can be extended to higher or lower octaves by multiplying the frequencies in it by a power of 2. (For example, the note 'A' occurs at frequencies 220, 440, 880, etc., because 440 = 2 × 220, 880 = 4 × 220, etc. However, it is not ok to call the frequency 660 = 3 × 220 an 'A'; the frequency 660 is an 'E', because 600 = 2 × 330, and 330 is the frequency for 'E'.)

Remember: if a signal is sampled at frequency `Fs`, then the *Nyquist frequency* at the end of this first half is `Fs/2`.

(a) The length of `y` is $n = 398180$. Find the integer factors of n by using the Matlab function `factor()`. Also find the integer factors of $(n - 4964)$. Find out how much cpu time it takes to compute `fft(y(1:n));` and `fft(y(1:(n-4964)));` with Matlab. (To do this, run each of these 100 times and compute the average time required, using `cputime`.) Explain the difference in timing.

(b) Plot the frequency spectrum for `y` by plotting the power of the 'first half' of its Fourier transform, and using `Fs` as the sampling frequency.
Assume for this assignment that the *power* of a complex value z is its complex absolute value squared:
i.e., power(z) $= |z|^2 = z\,\bar{z}$.
Here by *'first half'* of a sequence $s = [s_0, \ldots, s_{n-1}]$ we actually mean $[s_1, \ldots, s_m]$ where $m = \lfloor n/2 \rfloor$. For example, the 'first half' of $[0, 1, 2, 3, 4, 5, 6, 7]$ is $[1, 2, 3, 4]$.
Again: we omit the very first element in the `power` vector (which is the square of the sum of the data).

(c) There are altogether about $(5 \times 12) + 1 = 61$ notes in the equal-tempered scale with frequencies between low C (with frequency near 131) and the very high C with frequency near $131 \times 2^4 = 2096$.
Write a Matlab function `note_power(power)` that distills a power spectrum vector `power` into an 61×1 vector that gives, for each note in this list, the maximum value in `power` for all frequencies that are closest to this note.

For the mystery clip, use your function to print the power value for each of these 61 notes.

2. **Earthquakes**
Attached in the file TongaQuakes.csv is a list of about 5000 earthquakes of magnitude 5.0 in the Tonga/Kermadec Islands region near New Zealand. This region, where the Australian continent and Pacific Ocean meet, is highly active earthquake-wise.

You can read the file into Matlab with the `csvread()` function. The data was downloaded today from a server at http://www.iris.edu/seismon Earthquake times are reported in UTC (Coordinated Universal Time), which is essentially the same thing as GMT (Greenwich Mean Time).

   (a) Summarize the data in a histogram with 24 bins giving the number of earthquakes that occurred within each of the 24 hours of the day.
   Does this distribution of earthquakes differ significantly from a uniform distribution?

   (b) Generate a histogram with $12 \cdot 24 = 288$ bins of counts of earthquake occurrences for each hour of the day for each month. In other words, generate a histogram each of whose 288 bins is indexed by two integers: (month of the year) and (hour of the day).
   For simplicity, ignore leap years and assume each month's expected count value is the percentage (days in month)/(days in year) of the total. (So for example, January's expected percentage is 31/365 and February's is 28/365; and for January the expected percentage for each of the 24 hours is then (31/365)/24.) Does the observed histogram of earthquake counts differ significantly from the histogram with these expected values?

   (c) The first earthquake of the 5000 was on Feb 3, 1987, about 27 years ago. Using the Matlab `datenum()` function, generate a vector with one entry for each day since 1987. (This should have about 27×365 entries, except that `datenum` also takes leap years into account, so `datenum([2014 3 1])-datenum([2014 2 28])` = 1, but `datenum([2012 3 1])-datenum([2012 2 28])` = 2.) In each entry of this vector, put a '1' if there was any earthquakes on the corresponding day; this should be easy to accomplish using `datenum`. Thus you are measuring earthquake activity with a sampling frequency of $1/\text{day}$. (The Matlab `sunspots` demo may help.)
   Compute the Fourier power spectrum of the resulting vector, omitting the very first Fourier coefficient. There is one spike in this spectrum that has clearly higher power than others. Identify the frequency that has the highest power spike. Express this frequency in units $1/\text{year}$, $1/\text{month}$, and $1/\text{day}$. For example, a frequency of $0.01 \times 1/\text{day}$ is also $1/(100$ days) = (once every 100 days) = (once every 3.3 months) = (once every 0.27 year).

3. **A Simple 'Fake Photo' Detector**
A simple way to check if a $m \times n$ RGB image has been faked (with Photoshop, say) is to use the `reshape` function to convert it into a $(m\,n) \times 3$ matrix, compute the 3 principal components of its 3×3 covariance matrix, project the reshaped image on the second principal component, and then reshape the result back to a $m \times n$ grayscale image. If this grayscale image has bright or dark spots, it is possible that the color distribution in that area (and perhaps also that part of the image) has been altered. Write a program to do this, for any image file.

Determine whether each of the images in the directory `Apollo11Images/` is 'fake' or not, in the sense that the brightness suggests the image has been retouched.

Some history is in e.g. http://en.wikipedia.org/wiki/Examination_of_Apollo_Moon_photographs.

4. **Newtonizing**
The inverse square root $1/\sqrt{x}$ is useful in game software for normalizing a vector. Last year the *Quake* game got attention because its function `Q_rsqrt` used a mysterious algorithm surfaced for computing the inverse square root — the story is at: http://en.wikipedia.org/wiki/Fast_inverse_square_root

   (a) Develop a Newton iteration that computes $y = 1/\sqrt{x}$ when given an input value x. Specifically, Given a value x, show how to use the function `Newtonize()` to obtain a Newton iteration

$$y_{n+1} \;=\; g(y_n) \;=\; y_n \;-\; f(y_n)/f'(y_n)$$

   for an appropriately chosen function f that has a root at $1/\sqrt{x}$. Make your iteration terminate when the successive iteration values agree to 15 decimal digits, i.e., the relative error $|(y_{n+1} - y_n)/y_n|$ is less than 10^{-15}.

   (b) For each starting value y_0 in $[0.1, 0.2, \ldots, 1.0]$, determine how many iterations your method takes before converging (to within 15 decimal digits) to $1/\sqrt{3}$?

   (c) For a given input value x, what is a mathematical expression for the starting value y_0 used by the Quake function `Q_rsqrt` described in the Wikipedia article mentioned above? (Not the C code — a mathematical expression.) If you use this starting value, how many iterations does your program need to compute $1/\sqrt{3}$?