Physics 411: Homework 6

1. **The temperature of a light bulb:** An incandescent light bulb is a simple device—it contains a resistive filament, usually made of tungsten, heated by the flow of electricity until it becomes hot enough to radiate thermally. Essentially all of the power consumed by such a bulb is radiated as electromagnetic energy, but some of the radiation is not in the visible wavelengths, which means it is useless for lighting purposes.

   Let us define the efficiency of a light bulb to be the fraction of the radiated energy that falls in the visible band. It's a good approximation to assume that the radiation obeys the Planck radiation law, meaning that the power radiated per unit wavelength λ obeys

   $$I(\lambda) = 2\pi A h c^2 \frac{\lambda^{-5}}{e^{hc/\lambda k_B T} - 1},$$

   where A is the surface area of the filament, T is the temperature, h is Planck's constant, c is the speed of light, and k_B is Boltzmann's constant. The visible wavelengths run from $\lambda_1 = 390\,\text{nm}$ to $\lambda_2 = 750\,\text{nm}$, so the total energy radiated in the visible window is $\int_{\lambda_1}^{\lambda_2} I(\lambda)\,d\lambda$ and the total energy at all wavelengths is $\int_0^\infty I(\lambda)\,d\lambda$. Dividing one expression by the other and substituting for $I(\lambda)$ from above, we get an expression for the efficiency η of the light bulb thus:

   $$\eta = \frac{\int_{\lambda_1}^{\lambda_2} \lambda^{-5}/(e^{hc/\lambda k_B T} - 1)\,d\lambda}{\int_0^\infty \lambda^{-5}/(e^{hc/\lambda k_B T} - 1)\,d\lambda},$$

   where the leading constants and the area A have canceled out. Making the substitution $x = hc/\lambda k_B T$, this can also be written as

   $$\eta = \frac{\int_{hc/\lambda_2 k_B T}^{hc/\lambda_1 k_B T} x^3/(e^x - 1)\,dx}{\int_0^\infty x^3/(e^x - 1)\,dx} = \frac{15}{\pi^4} \int_{hc/\lambda_2 k_B T}^{hc/\lambda_1 k_B T} \frac{x^3}{e^x - 1}\,dx,$$

   where we have made use of the known exact value of the integral in the denominator.

   (a) Write a Python function that takes a temperature T as its argument and calculates the value of η for that temperature from the formula above. The integral in the formula cannot be done analytically, but you can do it numerically using any method of your choice. (For instance, Gaussian quadrature with 100 sample points works fine.) Use your function to make a graph of η as a function of temperature between 300 K and 10 000 K. You should see that there is an intermediate temperature where the efficiency is a maximum.

   (b) Calculate the temperature of maximum efficiency of the light bulb to within 1 K using golden ratio search. What efficiency does the bulb achieve at this temperature?

   (c) Is it practical to run a tungsten-filament light bulb at the temperature you found? If not, why not?

☑ **For full credit** turn a printout of your final (golden-ratio search) program, your plot from part (a), and your results and discussion from parts (b) and (c).

2. **Fourier transforms of simple functions:** Calculate the coefficients in the discrete Fourier transforms of the following periodic functions sampled at $N = 1000$ evenly spaced points, then make a plot of their amplitudes, similar to the plot shown in Fig. 7.3:

   (a) A single cycle of a square wave with amplitude 1

   (b) The sawtooth wave $y_n = n$

   (c) The modulated sine wave $y_n = \sin(\pi n/N)\sin(20\pi n/N)$

   ☑ **For full credit** turn in your program from part (a) and printouts of your three plots.

3. **Detecting periodicity:** In the on-line resources there is a file called `sunspots.dat`, which contains the observed number of sunspots on the Sun for each month since January 1749. You previously made a visualization of the data in this file for Homework 3.

   (a) Write a short program (or dig up your old one from Homework 3) that reads the data in the file and makes a graph of sunspots as a function of time. You should see that the number of sunspots has fluctuated on a regular cycle for as long as observations have been recorded. Make an estimate of the length of the cycle in months.

   (b) Modify your program to calculate the Fourier transform of the sunspot data and then make a graph of the magnitude squared $|c_k|^2$ of the Fourier coefficients as a function of k—also called the *power spectrum* of the sunspot signal. You should see that there is a noticeable peak in the power spectrum at a nonzero value of k. The appearance of this peak tells us that there is one frequency in the Fourier series that has a higher amplitude than the others around it—meaning that there is a large sine-wave term with this frequency, which corresponds to the periodic wave you can see in the original data.

   (c) Find the approximate value of k to which the peak corresponds. What is the period of the sine wave with this value of k? You should find that the period corresponds roughly to the length of the cycle that you estimated in part (a) above.

   This kind of Fourier analysis is a sensitive method for detecting periodicity in signals. Even in cases where it is not clear to the eye that there is a periodic component to a signal, it may still be possible to find one using a Fourier transform.

   ☑ **For full credit** turn your program and plot from part (b) and your results from part (c).

4. **Fourier filtering and smoothing:** In the on-line resources you'll find a file called `dow.dat`. It contains the daily closing value for each business day from late 2006 until the end of 2010 of the Dow Jones Industrial Average, which is a measure of average prices on the US stock market.

   Write a program to do the following:

   (a) Read in the data from `dow.dat` and plot them on a graph.

(b) Calculate the coefficients of the discrete Fourier transform of the data using the function `rfft` from `numpy.fft`, which produces an array of $\frac{1}{2}N + 1$ complex numbers.

(c) Now set all but the first 10% of the elements of this array to zero (i.e., set the last 90% to zero but keep the values of the first 10%).

(d) Calculate the inverse Fourier transform of the resulting array, zeros and all, using the function `irfft`, and plot it on the same graph as the original data. You may need to vary the colors of the two curves to make sure they both show up on the graph. Comment on what you see. What is happening when you set the Fourier coefficients to zero?

(e) Modify your program so that it sets all but the first 2% of the coefficients to zero and run it again.

☑ **For full credit** turn a printout of your program and a plot showing all three curves on the same axes.

5. **Extra credit:** This problem is optional, for those who want an extra challenge (and extra credit). It is pretty difficult, but you can learn a lot by completing it.

(a) Write your own program to compute the fast Fourier transform of a given set of N samples, in the case where N is a power of two, based on the formulas given in Section 7.5.1. As a test of your program, you can use it to calculate the Fourier transform of the data in the file `pitch.dat`, which can be found on the web site, and which is shown in Fig. 7.2 (and its transform is shown in Fig. 7.3).

You will have to calculate the coefficients $E_k^{(m,j)}$ from Eq. (7.40) for all levels m, which means that first you will have to plan how the coefficients will be stored. Since, as we have seen, there are exactly N of them at every level, one way to do it would be to create a two-dimensional complex array of size $N \times (1 + \log_2 N)$, so that it has N complex numbers for each level from zero to $\log_2 N$. Then within level m you have 2^m individual transforms denoted by $j = 0 \ldots 2^m - 1$, each with $N/2^m$ coefficients indexed by k. A simple way to arrange the coefficients would be to put all the $k = 0$ coefficients in a block one after another, then all the $k = 1$ coefficients, and so forth. Then $E_k^{(m,j)}$ would be stored in the $j + 2^m k$ element of the array.

This method has the advantage of being quite simple to program, but the disadvantage of using up a lot of memory space. The array contains $N \log_2 N$ complex numbers, and a complex number typically takes sixteen bytes of memory to store. So if you had to do a large Fourier transform of, say, $N = 10^8$ numbers, it would take $16N \log_2 N \simeq 42$ gigabytes of memory, which is much more than most computers have.

An alternative approach is to notice that we do not really need to store all of the coefficients. At any one point in the calculation we only need the coefficients at the current level and the previous level (from which the current level is calculated). If one is clever one can write a program that uses only two arrays, one for the current level and one for the previous level, each consisting of N complex numbers. Then our transform of 10^8 numbers would require less than four gigabytes, which is fine on most computers.

(There is a third way of storing the coefficients that is even more efficient. If you store the coefficients in the correct order, then you can arrange things so that every time you compute a coefficient for the next level, it gets stored in the same place as the old coefficient from the previous level from which it was calculated, and which you no longer need. With this way of doing things you only need one array of N complex numbers—we say the transform is done "in place." Unfortunately, this in-place Fourier transform is much harder to work out and harder to program. If you are feeling particularly ambitious you might want to give it a try, but it's not for the faint-hearted.)

(b) Use your program to duplicate the plot shown in Fig. 7.3. Confirm that your plot looks the same as the one in the figure.

☑ **To receive a substantial amount of extra credit** turn a printout of your program and your final plot.