COMPUTATIONAL METHODS IN RESEARCH

ASSIGNMENT 6

Exercise 1: Random Matrix (to do with Mathematica)

Write a matrix of dimension DxD (where $D = 1000$) filled with random numbers from a Gaussian distribution. Add this matrix to its transpose. Let us call the resulting matrix "mat". Use ";", so that your matrix is not seen.

Find the eigenvalues $E^{(i)}$ and the eigenvectors of the matrix "mat". Do NOT show them. Show only:

(a) The histogram of the eigenvalues.

(b) Look at the bin size chosen by Mathematica. Let us call that size "bin". Show a plot of the histogram above and on top of it (in black) a curve for the semicircle function:

$$f(E) = \frac{2 * D * bin\sqrt{1 - \frac{E^2}{8D}}}{\pi\sqrt{8D}}$$

(c) For each eigenstate, compute the participation ratio, which is defined as

$$PR^{(i)} = \frac{1}{\sum_k^D |a_k^{(i)}|^4}$$

Make a plot of $PR^{(i)}$ vs $E^{(i)}$ (use PlotRange from 0 to 400 for the y-axis).

Exercise 2: Solve equations (to do with Mathematica)

(a) Solve the equation

$$x^4 - 16x^3 + 61x^2 - 22x - 12 = 0,$$

exactly and numerically

(b) A theorem from algebra says that if

$$p(x) = a_0 + a_1x + a_2x^2 + ... + a_{n-1}x^(n-1) + a_nx^n,$$

the sum of the roots of the equation $p(x) = 0$ is

$$-\frac{a_{n-1}}{a_n}$$

and their product is

$$(-1)^n\frac{a_0}{a_n}$$

Verify this for the equation

$$20x^7 + 32x^6 - 221x^5 - 118x^4 + 725x^3 - 18x^2 - 726x + 252 = 0$$

(c) To solve the equation $5\cos(x) = 4 - x^3$, first, make a plot. Then make sure you find all solutions.

Exercise 3: Plots and Integrals (to do with Mathematica)

Find the area of the bounded region trapped between the graphs of the function $\exp(x)$ and the function $4 - x^2$. Follow these steps:

(a) Plot together the two functions to get an idea of the borders of the area and which of the functions is larger.

(b) Find the points where the two functions meet.

(c) Compute the area.

Exercise 4: Fourier transforms of simple functions

Write a Python program to calculate the coefficients in the discrete Fourier transforms of the following periodic function sampled at $N = 1000$ evenly spaced points, and make plots of their amplitudes similar to the plot shown in Fig. 7.4 of the book.

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

Use the Fourier transform function from the file `dft.py` that we discussed in class.

Exercise 5: Detecting periodicity

In the on-line resources there is a file called `sunspots.txt`, which contains the observed number of sunspots on the Sun for each month since January 1749. The file contains two columns of numbers, the first representing the month and the second being the sunspot number.

1. Write a program 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.
2. 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.

3. 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).

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.

Exercise 6: Fourier transforms of musical instruments

In the on-line resources you will find files called `piano.txt` and `trumpet.txt`, which contain data representing the waveform of a single note, played on, respectively, a piano and a trumpet.

1. Write a program that loads a waveform from one of these files, plots it, then calculates its discrete Fourier transform and plots the magnitudes of the first 10 000 coefficients in a manner similar to Fig. 7.4 of the book. Note that you will have to use a fast Fourier transform for the calculation because there are too many samples in the files to do the transforms the slow way in any reasonable amount of time.

   Apply your program to the piano and trumpet waveforms and discuss briefly what one can conclude about the sound of the piano and trumpet from the plots of Fourier coefficients.
2. Both waveforms were recorded at the industry-standard rate of 44 100 samples per second and both instruments were playing the same musical note when the recordings were made. From your Fourier transform results calculate what note they were playing. (Hint: The musical note middle C has a frequency of 261 Hz.)

Exercise 7: Fourier filtering and smoothing

In the on-line resources you'll find a file called `dow.txt`. 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:

1. Read in the data from `dow.txt` and plot them on a graph.
2. 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.
3. 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%).
4. 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?
5. Modify your program so that it sets all but the first 2% of the coefficients to zero and run it again.

Exercise 8

The function $f(t)$ represents a square-wave with amplitude 1 and frequency 1 Hz:

$$f(t) = \begin{cases} 1 & \text{if } \lfloor 2t \rfloor \text{ is even,} \\ -1 & \text{if } \lfloor 2t \rfloor \text{ is odd,} \end{cases} \tag{1}$$

where $\lfloor x \rfloor$ means x rounded down to the next lowest integer. Let us attempt to smooth this function using a Fourier transform, as we did in the previous exercise. Write a program that creates an array of $N = 1000$ elements containing a thousand equally spaced samples from a single cycle of this square-wave. Calculate the discrete Fourier transform of the array. Now set all but the first ten Fourier coefficients to zero, then invert the Fourier transform again to recover the smoothed signal. Make a plot of the result and on the same axes show the original square-wave as well. You should find that the signal is not simply smoothed—there are artifacts, wiggles, in the results.

Artifacts similar to these arise when Fourier coefficients are discarded in audio and visual compression schemes like those described in Section 7.3.1 of the book and are the primary source of imperfections in digitally compressed sound and images.

Exercise 9 - 10: Image deconvolution

You've probably seen it on TV, in one of those crime drama shows. They have a blurry photo of a crime scene and they click a few buttons on the computer and magically the photo becomes sharp and clear, so you can make out someone's face, or some lettering on a sign. Surely (like almost everything else on such TV shows) this is just science fiction? Actually, no. It's not. It's real and in this exercise you'll write a program that does it.

When a photo is blurred each point on the photo gets smeared out according to some "smearing distribution," which is technically called a *point spread function*. We can represent this smearing mathematically as follows. For simplicity let's assume we're working with a black and white photograph, so that the picture can be represented by a single function $a(x, y)$ which tells you the brightness at each point (x, y). And let us denote the point spread function by $f(x, y)$. This means that a single bright dot at the origin ends up appearing as $f(x, y)$ instead. If $f(x, y)$ is a broad function then the picture is badly blurred. If it is a narrow peak then the picture is relatively sharp.

In general the brightness $b(x, y)$ of the blurred photo at point (x, y) is given by

$$b(x, y) = \int_0^K \int_0^L a(x', y') f(x - x', y - y') \, \mathrm{d}x' \, \mathrm{d}y',$$

where $K \times L$ is the dimension of the picture. This equation is called the *convolution* of the picture with the point spread function.

Working with two-dimensional functions can get complicated, so to get the idea of how the math works, let's switch temporarily to a one-dimensional equivalent of our problem. Once we work out the details in 1D we'll return to the 2D version. The one-dimensional version of the convolution above would be

$$b(x) = \int_0^L a(x') f(x - x') \, \mathrm{d}x'.$$

The function $b(x)$ can be represented by a Fourier series as in Eq. (7.5):

$$b(x) = \sum_{k=-\infty}^{\infty} \tilde{b}_k \exp\left(\mathrm{i}\frac{2\pi k x}{L}\right),$$

where

$$\tilde{b}_k = \frac{1}{L}\int_0^L b(x) \exp\left(-\mathrm{i}\frac{2\pi k x}{L}\right) \mathrm{d}x$$

are the Fourier coefficients. Substituting for $b(x)$ in this equation gives

$$\begin{aligned}\tilde{b}_k &= \frac{1}{L}\int_0^L \int_0^L a(x') f(x-x') \exp\left(-\mathrm{i}\frac{2\pi k x}{L}\right) \mathrm{d}x' \,\mathrm{d}x \\ &= \frac{1}{L}\int_0^L \int_0^L a(x') f(x-x') \exp\left(-\mathrm{i}\frac{2\pi k (x-x')}{L}\right) \exp\left(-\mathrm{i}\frac{2\pi k x'}{L}\right) \mathrm{d}x' \,\mathrm{d}x.\end{aligned}$$

Now let us change variables to $X = x - x'$, and we get

$$\tilde{b}_k = \frac{1}{L}\int_0^L a(x') \exp\left(-\mathrm{i}\frac{2\pi k x'}{L}\right) \int_{-x'}^{L-x'} f(X) \exp\left(-\mathrm{i}\frac{2\pi k X}{L}\right) \mathrm{d}X \,\mathrm{d}x'.$$

If we make $f(x)$ a periodic function in the standard fashion by repeating it infinitely many times to the left and right of the interval from 0 to L, then the second integral above can be written as

$$\begin{aligned}\int_{-x'}^{L-x'} f(X) \exp\left(-\mathrm{i}\frac{2\pi k X}{L}\right) \mathrm{d}X &= \int_{-x'}^{0} f(X) \exp\left(-\mathrm{i}\frac{2\pi k X}{L}\right) \mathrm{d}X \\ &\qquad + \int_0^{L-x'} f(X) \exp\left(-\mathrm{i}\frac{2\pi k X}{L}\right) \mathrm{d}X \\ &= \exp\left(\mathrm{i}\frac{2\pi k L}{L}\right) \int_{L-x'}^{L} f(X) \exp\left(-\mathrm{i}\frac{2\pi k X}{L}\right) \mathrm{d}X + \int_0^{L-x'} f(X) \exp\left(-\mathrm{i}\frac{2\pi k X}{L}\right) \mathrm{d}X \\ &= \int_0^L f(X) \exp\left(-\mathrm{i}\frac{2\pi k X}{L}\right) \mathrm{d}X,\end{aligned}$$

which is simply L times the Fourier transform $\tilde{f}_k$ of $f(x)$. Substituting this result back into our equation for $\tilde{b}_k$ we then get

$$\tilde{b}_k = \int_0^L a(x') \exp\left(-\mathrm{i}\frac{2\pi k x'}{L}\right) \tilde{f}_k \,\mathrm{d}x' = L\,\tilde{a}_k \tilde{f}_k.$$

In other words, apart from the factor of L, the Fourier transform of the blurred photo is the product of the Fourier transforms of the unblurred photo and the point spread function.

Now it is clear how we deblur our picture. We take the blurred picture and Fourier transform it to get $\tilde{b}_k = L\,\tilde{a}_k \tilde{f}_k$. We also take the point spread function and Fourier transform it to get $\tilde{f}_k$. Then we divide one by the other:

$$\frac{\tilde{b}_k}{L\tilde{f}_k} = \tilde{a}_k$$

which gives us the Fourier transform of the *unblurred* picture. Then, finally, we do an inverse Fourier transform on $\tilde{a}_k$ to get back the unblurred picture. This process of recovering the unblurred picture from the blurred one, of reversing the convolution process, is called *deconvolution*.

Real pictures are two-dimensional, but the mathematics follows through exactly the same. For a picture of dimensions $K \times L$ we find that the two-dimensional Fourier transforms are related by

$$\tilde{b}_{kl} = KL\tilde{a}_{kl}\tilde{f}_{kl}\,,$$

and again we just divide the blurred Fourier transform by the Fourier transform of the point spread function to get the Fourier transform of the unblurred picture.

In the digital realm of computers, pictures are not pure functions $f(x,y)$ but rather grids of samples, and our Fourier transforms are discrete transforms not continuous ones. But the math works out the same again.

The main complication with deblurring in practice is that we don't usually know the point spread function. Typically we have to experiment with different ones until we find something that works. For many cameras it's a reasonable approximation to assume the point spread function is Gaussian:

$$f(x,y) = \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right),$$

where σ is the width of the Gaussian. Even with this assumption, however, we still don't know the value of σ and we may have to experiment to find a value that works well. In the following exercise, for simplicity, we'll assume we know the value of σ.

1. On the web site you will find a file called `blur.txt` that contains a grid of values representing brightness on a black-and-white photo—a badly out-of-focus one that has been deliberately blurred using a Gaussian point spread function of width $\sigma = 25$. Write a program that reads the grid of values into a two-dimensional array of real numbers and then draws the values on the screen of the computer as a density plot. You should see the photo appear. If you get something wrong it might be upside-down. Work with the details of your program until you get it appearing correctly. (Hint: The picture has the sky, which is bright, at the top and the ground, which is dark, at the bottom.)
2. Write another program that creates an array, of the same size as the photo, containing a grid of samples drawn from the Gaussian $f(x,y)$ above with $\sigma = 25$. Make a density plot of these values on the screen too, so that you get a visualization of your point spread function. Remember that the point spread function is periodic (along both axes), which means that the values for negative x and y are repeated at the end of the interval. Since the Gaussian is centered on the origin, this means there should be bright patches in each of the four corners of your picture, something like this:

3. Combine your two programs and add Fourier transforms using the functions `rfft2` and `irfft2` from `numpy.fft`, to make a program that does the following:

   i) Reads in the blurred photo
   ii) Calculates the point spread function
   iii) Fourier transforms both
   iv) Divides one by the other
   v) Performs an inverse transform to get the unblurred photo
   vi) Displays the unblurred photo on the screen

   When you are done, you should be able to make out the scene in the photo, although probably it will still not be perfectly sharp.

   Hint: One thing you'll need to deal with is what happens when the Fourier transform of the point spread function is zero, or close to zero. In that case if you divide by it you'll get an error (because you can't divide by zero) or just a very large number (because you're dividing by something small). A workable compromise is that if a value in the Fourier transform of the point spread function is smaller than a certain amount ϵ you don't divide by it—just leave that coefficient alone. The value of ϵ is not very critical but a reasonable value seems to be 10^{-3}.

4. Bearing in mind this last point about zeros in the Fourier transform, what is it that limits our ability to deblur a photo? Why can we not perfectly unblur any photo and make it completely sharp?

We have seen this process in action here for a normal snapshot, but it is also used in many physics applications where one takes photos. For instance, it is used in astronomy to enhance photos taken by telescopes. It was famously used with images from the Hubble Space Telescope after it was realized that the telescope's main mirror had a serious manufacturing flaw and was returning blurry photos—scientists managed to partially correct the blurring using Fourier transform techniques.

Exercise 10: Monte Carlo (use either Python or Mathematica)

Birthday Problem.

We want to find out the probability that out of 30 people two share a birthday.

Person 1
Person 2: prob=364/365 of no overlap with the first;
Person 3: prob=363/365 of no overlap with 1 and 2;
Person 4: prob=362/365 of no overlap with 1, 2 and 3;
...
Person 30: prob=336/365 of no overlap with any person above;

The probability of having no shared birthdays is then

$$(364/365) * (363/365) * (362/365)...(336/365) = 0.293684$$

So the probability of having at least one pair of people having the same birthday is 71%.

Let us find this probability with the Monte Carlo Approach:
1) Pick 30 random numbers in the range [1,365].
2) Check to see if any of the thirty are equal.
3) Go back to step 1 and repeat 10000 times.
4) Report the fraction of trial that have matching birthdays and use it to compare with the result above.