APPLIED PHYSICS 155

EXAM 2. Due: 9 April 2018, 10:00AM (via UVLe)

Instructions:

1. Use one ipynb file per problem. Label the files problemX.ipynb, where X is the problem number and submit all five in one compressed file.
2. Work alone - discussing the problem/solution with anyone is to be avoided until after the deadline of submission. Any code you turn in must be your own.
3. Submit your solution via UVLe. Emailed, hardcopy, and other non-UVLe submissions will receive a grade of zero. You may (and are encouraged to) resubmit your solution as often as practicable prior to the deadline. Late/no submissions will receive a grade of zero.

Problem 2.1: Quantum uncertainty in the harmonic oscillator (20 points)

In units where all the constants are 1, the wavefunction of the nth energy level of the one-dimensional quantum harmonic oscillator—i.e., a spinless point particle in a quadratic potential well—is given by

$$\psi_n(x) = \frac{1}{\sqrt{2^n n! \sqrt{\pi}}} \, \mathrm{e}^{-x^2/2} \, H_n(x),$$

for $n = 0 \ldots \infty$, where $H_n(x)$ is the nth Hermite polynomial. Hermite polynomials satisfy a relation somewhat similar to that for the Fibonacci numbers, although more complex:

$$H_{n+1}(x) = 2xH_n(x) - 2nH_{n-1}(x).$$

The first two Hermite polynomials are $H_0(x) = 1$ and $H_1(x) = 2x$.

a) Write a user-defined function `H(n,x)` that calculates $H_n(x)$ for given x and any integer $n \geq 0$. Use your function to make a plot that shows the harmonic oscillator wavefunctions for $n = 0$, 1, 2, and 3, all on the same graph, in the range $x = -4$ to $x = 4$. Hint: There is a function `factorial` in the `math` package that calculates the factorial of an integer.

b) Make a separate plot of the wavefunction for $n = 30$ from $x = -10$ to $x = 10$. Hint: If your program takes too long to run in this case, then you're doing the calculation wrong—the program should take only a second or so to run.

c) The quantum uncertainty of a particle in the nth level of a quantum harmonic oscillator can be quantified by its root-mean-square position $\sqrt{\langle x^2 \rangle}$, where

$$\langle x^2 \rangle = \int_{-\infty}^{\infty} x^2 |\psi_n(x)|^2 \, \mathrm{d}x.$$

Write a program that evaluates this integral using Gaussian quadrature on 100 points and then calculates the uncertainty (i.e., the root-mean-square position of the particle) for a given value of n. Use your program to calculate the uncertainty for $n = 5$. You should get an answer in the vicinity of $\sqrt{\langle x^2 \rangle} = 2.3$.

Problem 2.2: The gamma function (20 points)

A commonly occurring function in physics calculations is the gamma function $\Gamma(a)$, which is defined by the integral

$$\Gamma(a) = \int_0^\infty x^{a-1} \mathrm{e}^{-x} \, \mathrm{d}x.$$

There is no closed-form expression for the gamma function, but one can calculate its value for given a by performing the integral above numerically. You have to be careful how you do it, however, if you wish to get an accurate answer.

a) Write a program to make a graph of the value of the integrand $x^{a-1}\mathrm{e}^{-x}$ as a function of x from $x = 0$ to $x = 5$, with three separate curves for $a = 2$, 3, and 4, all on the same axes. You should find that the integrand starts at zero, rises to a maximum, and then decays again for each curve.

b) Show analytically that the maximum falls at $x = a - 1$.

c) Most of the area under the integrand falls near the maximum, so to get an accurate value of the gamma function we need to do a good job of this part of the integral. We can change the integral from 0 to ∞ to one over a finite range from 0 to 1 using the change of variables in Eq. (5.67), but this tends to squash the peak towards the edge of the $[0, 1]$ range and does a poor job of evaluating the integral accurately. We can do a better job by making a different change of variables that puts the peak in the middle of the integration range, around $\frac{1}{2}$. We will use the change of variables given in Eq. (5.69), which we repeat here for convenience:

$$z = \frac{x}{c + x}.$$

For what value of x does this change of variables give $z = \frac{1}{2}$? Hence what is the appropriate choice of the parameter c that puts the peak of the integrand for the gamma function at $z = \frac{1}{2}$?

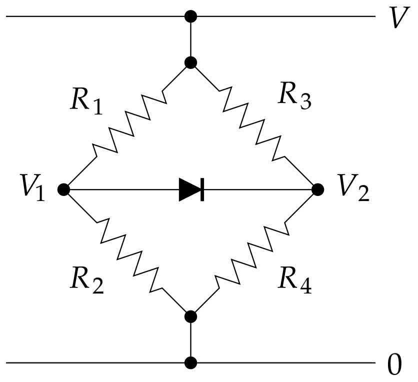
d) Before we can calculate the gamma function, there is another detail we need to attend to. The integrand $x^{a-1}\mathrm{e}^{-x}$ can be difficult to evaluate because the factor x^{a-1} can become very large and the factor e^{-x} very small, causing numerical overflow or underflow, or both, for some values of x. Write $x^{a-1} = \mathrm{e}^{(a-1)\ln x}$ to derive an alternative expression for the integrand that does not suffer from these problems (or at least not so much). Explain why your new expression is better than the old one.

e) Now, using the change of variables above and the value of c you have chosen, write a user-defined function `gamma(a)` to calculate the gamma function for arbitrary argument a. Use whatever integration method you feel is appropriate. Test your function by using it to calculate and print the value of $\Gamma(\frac{3}{2})$, which is known to be equal to $\frac{1}{2}\sqrt{\pi} \simeq 0.886$.

f) For integer values of a it can be shown that $\Gamma(a)$ is equal to the factorial of $a-1$. Use your Python function to calculate $\Gamma(3)$, $\Gamma(6)$, and $\Gamma(10)$. You should get answers closely equal to $2! = 2$, $5! = 120$, and $9! = 362\,880$.

Problem 2.3: Nonlinear circuits

Consider the following simple circuit, a variation on the classic Wheatstone bridge:



The resistors obey the normal Ohm law, but the diode obeys the diode equation:

$$I = I_0(\mathrm{e}^{V/V_T} - 1),$$

where V is the voltage across the diode and I_0 and V_T are constants.

a) The Kirchhoff current law says that the total net current flowing into or out of every point in a circuit must be zero. Applying the law to voltage V_1 in the circuit above we get

$$\frac{V_1 - V_+}{R_1} + \frac{V_1}{R_2} + I_0\big[\mathrm{e}^{(V_1-V_2)/V_T} - 1\big] = 0.$$

Derive the corresponding equation for voltage V_2.

b) Solve the two nonlinear equations for the voltages V_1 and V_2 with the conditions

$$\begin{aligned} V_+ &= 5\,\mathrm{V}, \\ R_1 &= 1\,\mathrm{k}\Omega, & R_2 &= 4\,\mathrm{k}\Omega, & R_3 &= 3\,\mathrm{k}\Omega, & R_4 &= 2\,\mathrm{k}\Omega, \\ I_0 &= 3\,\mathrm{nA}, & V_T &= 0.05\,\mathrm{V}. \end{aligned}$$

You can use either the relaxation method or Newton's method to solve the equations. If you use Newton's method you can solve $\nabla\mathbf{f}\cdot\Delta\mathbf{x} = \mathbf{f}(\mathbf{x})$ for $\Delta\mathbf{x}$ using the function `solve()` from `numpy.linalg` if you want to, but in this case the matrix is only a 2×2 matrix, so it's easy to calculate the inverse directly too.

c) The electronic engineer's rule of thumb for diodes is that the voltage across a (forward biased) diode is always about 0.6 volts. Confirm that your results agree with this rule.

Problem 2.4: Fourier filtering (20 points)

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. Explain briefly where these come from.

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

Problem 2.5: Image deconvolution (20 points)

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):

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

where

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

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

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

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

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

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

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

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

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

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.

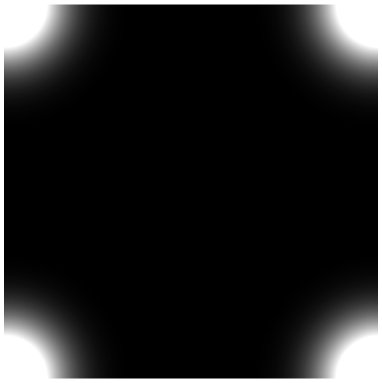
The main complication with deblurring in practice is that we don't usually know the point spread function. Typically we have to experiment with different ones until we find something that works. For many cameras it's a reasonable approximation to assume the point spread function is Gaussian:

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

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.

a) On the web site [`http://www-personal.umich.edu/~mejn/cp/data/blur.txt`] 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.)

b) Write another program that creates an array, of the same size as the photo, containing a grid of samples from 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:



c) 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}.

d) 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.