PHYS 7411 Spring 2015
Computational Physics
Homework 3

Due by 3:00pm in Nicholson 447 on 13 March 2015, Friday the 13th

(Any late assignments will be penalized in the amount of 25% per day late. Any copying of computer programs will be penalized with no credit.)

Exercise 1: The Stefan–Boltzmann constant

The Planck theory of thermal radiation tells us that in the (angular) frequency interval ω to $\omega + \mathrm{d}\omega$, a black body of unit area radiates electromagnetically an amount of thermal energy per second equal to $I(\omega)\,\mathrm{d}\omega$, where

$$I(\omega) = \frac{\hbar}{4\pi^2 c^2}\,\frac{\omega^3}{(\mathrm{e}^{\hbar\omega/k_B T} - 1)}.$$

Here $\hbar$ is Planck's constant over 2π, c is the speed of light, and k_B is Boltzmann's constant.

a) Show that the total energy per unit area radiated by a black body is

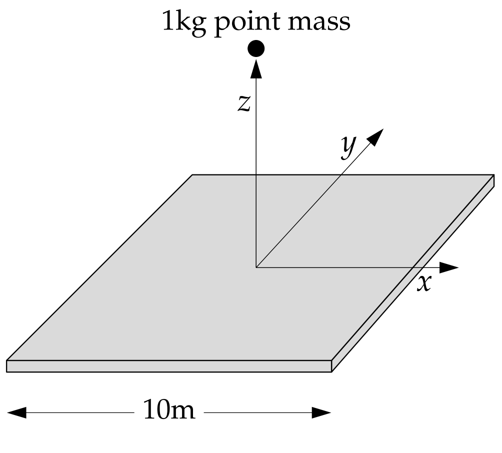
$$W = \frac{k_B^4 T^4}{4\pi^2 c^2 \hbar^3}\int_0^\infty \frac{x^3}{\mathrm{e}^x - 1}\,\mathrm{d}x.$$

b) Write a program to evaluate the integral in this expression. Explain what method you used, and how accurate you think your answer is.

c) Even before Planck gave his theory of thermal radiation around the turn of the 20th century, it was known that the total energy W given off by a black body per unit area per second followed Stefan's law: $W = \sigma T^4$, where σ is the Stefan–Boltzmann constant. Use your value for the integral above to compute a value for the Stefan–Boltzmann constant (in SI units) to three significant figures. Check your result against the known value, which you can find in books or on-line. You should get good agreement.

Exercise 2: Gravitational pull of a uniform sheet

A uniform square sheet of metal is floating motionless in space:



The sheet is 10 m on a side and of negligible thickness, and it has a mass of 10 metric tonnes.

a) Consider the gravitational force due to the plate felt by a point mass of 1 kg a distance z from the center of the square, in the direction perpendicular to the sheet, as shown above. Show that the component of the force along the z-axis is

$$F_z = G\sigma z \iint_{-L/2}^{L/2} \frac{\mathrm{d}x\,\mathrm{d}y}{(x^2+y^2+z^2)^{3/2}}\,,$$

where $G = 6.674 \times 10^{-11}\,\mathrm{m}^3\,\mathrm{kg}^{-1}\,\mathrm{s}^{-2}$ is Newton's gravitational constant and σ is the mass per unit area of the sheet.

b) Write a program to calculate and plot the force as a function of z from $z = 0$ to $z = 10\,\mathrm{m}$. For the double integral use (double) Gaussian quadrature, as in Eq. (5.82), with 100 sample points along each axis.

c) You should see a smooth curve, except at very small values of z, where the force should drop off suddenly to zero. This drop is not a real effect, but an artifact of the way we have done the calculation. Explain briefly where this artifact comes from and suggest a strategy to remove it, or at least to decrease its size.

This calculation can thought of as a model for the gravitational pull of a galaxy. Most of the mass in a spiral galaxy (such as our own Milky Way) lies in a thin plane or disk passing through the galactic center, and the gravitational pull exerted by that plane on bodies outside the galaxy can be calculated by just the methods we have employed here.

Exercise 3: Create a user-defined function `f(x)` that returns the value $1 + \frac{1}{2}\tanh 2x$, then use a central difference to calculate the derivative of the function in the range $-2 \le x \le 2$. Calculate an analytic formula for the derivative and make a graph with your numerical result and the analytic answer on the same plot. It may help to plot the exact answer as lines and the numerical one as dots. (Hint: In Python the tanh function is found in the `math` package, and it's called simply `tanh`.)

Exercise 4: The gamma function

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 $9! = 362\,880$.

Exercise 5: Electric field of a charge distribution

Suppose we have a distribution of charges and we want to calculate the resulting electric field. One way to do this is to first calculate the electric potential ϕ and then take its gradient. For a point charge q at the origin, the electric potential at a distance r from the origin is $\phi = q/4\pi\epsilon_0 r$ and the electric field is $\mathbf{E} = -\nabla\phi$.

a) You have two charges, of $\pm 1\,\mathrm{C}$, 10 cm apart. Calculate the resulting electric potential on a $1\,\mathrm{m} \times 1\,\mathrm{m}$ square plane surrounding the charges and passing through them. Calculate the potential at 1 cm spaced points in a grid and make a visualization on the screen of the potential using a density plot.

b) Now calculate the partial derivatives of the potential with respect to x and y and hence find the electric field in the xy plane. Make a visualization of the field also. This is a little trickier than visualizing the potential, because the electric field has both magnitude and direction. One way to do it might be to make two density plots, one for the magnitude, and one for the direction, the latter using the "`hsv`" color scheme in `pylab`, which is a rainbow scheme that passes through all the colors but starts and ends with the same shade of red, which makes it suitable for representing things like directions or angles that go around the full circle and end up where they started. A more sophisticated visualization might use the arrow object from the `visual` package, drawing a grid of arrows with direction and length chosen to represent the field.

c) Now suppose you have a continuous distribution of charge over an $L \times L$ square. The charge density in Cm^{-2} is

$$\sigma(x, y) = q_0 \sin\frac{2\pi x}{L} \sin\frac{2\pi y}{L}.$$

Calculate and visualize the resulting electric field at 1 cm-spaced points in 1 square meter of the xy plane for the case where $L = 10\,\mathrm{cm}$, the charge distribution is centered in the middle of the visualized area, and $q_0 = 100\,\mathrm{Cm}^{-2}$. You will have to perform a double integral over x and y, then differentiate the potential with respect to position to get the electric field. Choose whatever integration method seems appropriate for the integrals.

Exercise 6:

a) Modify the program `gausselim.py` in Example 6.1 to incorporate partial pivoting (or you can write your own program from scratch if you prefer). Run your program and demonstrate that it gives the same answers as the original program when applied to Eq. (6.1)

b) Modify the program to solve the equations in (6.17) and show that it can find the solution to these as well, even though Gaussian elimination without pivoting fails.

Exercise 7: The QR algorithm

In this exercise you'll write a program to calculate the eigenvalues and eigenvectors of a real symmetric matrix using the QR algorithm. The first challenge is to write a program that finds the QR decomposition of a matrix. Then we'll use that decomposition to find the eigenvalues.

As described above, the QR decomposition expresses a real square matrix $\mathbf{A}$ in the form $\mathbf{A} = \mathbf{QR}$, where $\mathbf{Q}$ is an orthogonal matrix and $\mathbf{R}$ is an upper-triangular matrix. Given an $N \times N$ matrix $\mathbf{A}$ we can compute the QR decomposition as follows.

Let us think of the matrix as a set of N column vectors $\mathbf{a}_0 \ldots \mathbf{a}_{N-1}$ thus:

$$\mathbf{A} = \begin{pmatrix} | & | & | & \cdots \\ \mathbf{a}_0 & \mathbf{a}_1 & \mathbf{a}_2 & \cdots \\ | & | & | & \cdots \end{pmatrix},$$

where we have numbered the vectors in Python fashion, starting from zero, which will be convenient when writing the program. We now define two new sets of vectors $\mathbf{u}_0 \ldots \mathbf{u}_{N-1}$ and $\mathbf{q}_0 \ldots \mathbf{q}_{N-1}$ as follows:

$$\begin{aligned} \mathbf{u}_0 &= \mathbf{a}_0, & \mathbf{q}_0 &= \frac{\mathbf{u}_0}{|\mathbf{u}_0|}, \\ \mathbf{u}_1 &= \mathbf{a}_1 - (\mathbf{q}_0 \cdot \mathbf{a}_1)\mathbf{q}_0, & \mathbf{q}_1 &= \frac{\mathbf{u}_1}{|\mathbf{u}_1|}, \\ \mathbf{u}_2 &= \mathbf{a}_2 - (\mathbf{q}_0 \cdot \mathbf{a}_2)\mathbf{q}_0 - (\mathbf{q}_1 \cdot \mathbf{a}_2)\mathbf{q}_1, & \mathbf{q}_2 &= \frac{\mathbf{u}_2}{|\mathbf{u}_2|}, \end{aligned}$$

and so forth. The general formulas for calculating $\mathbf{u}_i$ and $\mathbf{q}_i$ are

$$\mathbf{u}_i = \mathbf{a}_i - \sum_{j=0}^{i-1} (\mathbf{q}_j \cdot \mathbf{a}_i)\mathbf{q}_j, \qquad \mathbf{q}_i = \frac{\mathbf{u}_i}{|\mathbf{u}_i|}.$$

a) Show, by induction or otherwise, that the vectors $\mathbf{q}_i$ are orthonormal, i.e., that they satisfy

$$\mathbf{q}_i \cdot \mathbf{q}_j = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{if } i \neq j. \end{cases}$$

Now, rearranging the definitions of the vectors, we have

$$\begin{aligned} \mathbf{a}_0 &= |\mathbf{u}_0|\,\mathbf{q}_0, \\ \mathbf{a}_1 &= |\mathbf{u}_1|\,\mathbf{q}_1 + (\mathbf{q}_0 \cdot \mathbf{a}_1)\mathbf{q}_0, \\ \mathbf{a}_2 &= |\mathbf{u}_2|\,\mathbf{q}_2 + (\mathbf{q}_0 \cdot \mathbf{a}_2)\mathbf{q}_0 + (\mathbf{q}_1 \cdot \mathbf{a}_2)\mathbf{q}_1, \end{aligned}$$

and so on. Or we can group the vectors $\mathbf{q}_i$ together as the columns of a matrix and write all of these equations as a single matrix equation

$$\mathbf{A} = \begin{pmatrix} | & | & | & \cdots \\ \mathbf{a}_0 & \mathbf{a}_1 & \mathbf{a}_2 & \cdots \\ | & | & | & \cdots \end{pmatrix} = \begin{pmatrix} | & | & | & \cdots \\ \mathbf{q}_0 & \mathbf{q}_1 & \mathbf{q}_2 & \cdots \\ | & | & | & \cdots \end{pmatrix} \begin{pmatrix} |\mathbf{u}_0| & \mathbf{q}_0 \cdot \mathbf{a}_1 & \mathbf{q}_0 \cdot \mathbf{a}_2 & \cdots \\ 0 & |\mathbf{u}_1| & \mathbf{q}_1 \cdot \mathbf{a}_2 & \cdots \\ 0 & 0 & |\mathbf{u}_2| & \cdots \end{pmatrix}.$$

(If this looks complicated it's worth multiplying out the matrices on the right to verify for yourself that you get the correct expressions for the $\mathbf{a}_i$.)

Notice now that the first matrix on the right-hand side of this equation, the matrix with columns $\mathbf{q}_i$, is orthogonal, because the vectors $\mathbf{q}_i$ are orthonormal, and the second matrix is upper triangular. In other words, we have found the QR decomposition $\mathbf{A} = \mathbf{QR}$. The matrices $\mathbf{Q}$ and $\mathbf{R}$ are

$$\mathbf{Q} = \begin{pmatrix} | & | & | & \cdots \\ \mathbf{q}_0 & \mathbf{q}_1 & \mathbf{q}_2 & \cdots \\ | & | & | & \cdots \end{pmatrix}, \qquad \mathbf{R} = \begin{pmatrix} |\mathbf{u}_0| & \mathbf{q}_0 \cdot \mathbf{a}_1 & \mathbf{q}_0 \cdot \mathbf{a}_2 & \cdots \\ 0 & |\mathbf{u}_1| & \mathbf{q}_1 \cdot \mathbf{a}_2 & \cdots \\ 0 & 0 & |\mathbf{u}_2| & \cdots \end{pmatrix}.$$

b) Write a Python function that takes as its argument a real square matrix $\mathbf{A}$ and returns the two matrices $\mathbf{Q}$ and $\mathbf{R}$ that form its QR decomposition. As a test case, try out your function on the matrix

$$\mathbf{A} = \begin{pmatrix} 1 & 4 & 8 & 4 \\ 4 & 2 & 3 & 7 \\ 8 & 3 & 6 & 9 \\ 4 & 7 & 9 & 2 \end{pmatrix}.$$

Check the results by multiplying $\mathbf{Q}$ and $\mathbf{R}$ together to recover the original matrix $\mathbf{A}$ again.

c) Using your function, write a complete program to calculate the eigenvalues and eigenvectors of a real symmetric matrix using the QR algorithm. Continue the calculation until the magnitude of every off-diagonal element of the matrix is smaller than 10^{-6}. Test your program on the example matrix above. You should find that the eigenvalues are 1, 21, -3, and -8.