Lab assignment #10: Random numbers, Monte Carlo integration

Instructor: Nicolas Grisouard (nicolas.grisouard@utoronto.ca)
TA & Marker: Ahmed Rayyan (a.rayyan@mail.utoronto.ca)

Due Friday, 26 November 2021, 5 pm

It appears that there is no need for a second room. Should it be incorrect and the room overflows, N.G. will open up the room he had for the previous labs (grab the link on the old question documents, e.g., Lab #3).

Room 1 (also for office hour) Ahmed Rayyan and Nicolas Grisouard,
https://gather.town/app/x4DAC1wGAgr6jdtS/phy407-ng-room2
PWD: `phy407-ng-room2`

Physics and mathematics background

Spherical to Cartesian coordinates: Assuming spherical coordinates of radius r, inclination θ, azimuth ϕ, with suitable limits, the Cartesian coordinates can be determined by

$$x = r\sin\theta\cos\phi, \tag{1}$$
$$y = r\sin\theta\sin\phi, \tag{2}$$
$$z = r\cos\theta. \tag{3}$$

Limb darkening in plane-parallel, grey atmospheres Limb darkening describes the fact that a (normal) star seems to be darker at the edge than at its centre. This can be clearly observed for the Sun (see Fig. 1). If the direction cosine (i.e., the cosine of angle between the radius vector and photon direction) is denoted by μ, one can approximate the angular dependency of the specific intensity by the expression

$$I(\mu) \approx (0.4 + 0.6\mu)\, I_1, \tag{4}$$

where $I_1 = I(\mu = 1)$ is the specific intensity for the radial direction, $\mu = 1$, i.e., the intensity observed at the centre of a star. For $\mu = 0$, on the other hand, the angle between radial and photon's direction is 90° (this condition is met at the limb of the star), and the corresponding region appears to be fainter, by a factor of roughly 0.4.

Equation (4) is an approximate solution of the equation of radiative transfer in 'plane-parallel symmetry', which assumes that the stellar photosphere is very thin compared to the stellar radius. For example, the solar photosphere is only a few hundred kilometers thick, while the Sun's radius is

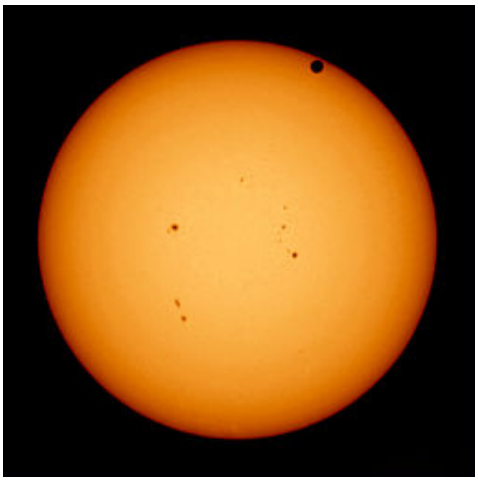

Figure 1: Limb darkening is observed for the Sun, where the edges have a lower luminosity than the centre. Credit: Wikipedia.

around 700,000 km). In addition, we assume that the absorption and emission processes assumed are independent of frequency, i.e., 'grey'.

Note: Equation (4) *has nothing to do with the presence of any temperature stratification, although it can be influenced by it. Instead, it is the consequence of a large number of absorption and emission processes in an atmosphere of finite optical depth, which describes how much absorption occurs when light travels through an absorbing medium.*

Radiation transfer

Computational background

Random number generators There is not one go-to Python package for random number generation. We have seen a few already, and the following pages have a lot more info:

https://numpy.org/doc/stable/reference/random/index.html

https://docs.python.org/3/library/random.html

Loading land data The Earth map for question 1 is stored in a file called `Earth.npz`. This map contains a dictionnary with three entries: a 2D array of integers containing ones for points over land, and zeros for points over water, and two 1D NumPy arrays containing the latitude and longitude coordinates for the 2D array. To read the file and load each entry onto NumPy arrays, you can use

```
import numpy as np
loaded = np.load('Earth.npz')
data = loaded['data']
lon_array = loaded['lon']
lat_array = loaded['lat']
```

Hint: After you load the data, plot stuff to make sure you understand what is in it.

Nearest interpolator Say you have a 2D data array `data`, and the x- and y-arrays that define the grid are the 1D arrays `X` and `Y`, which have the same lengths as the first and second dimensions of `data`.

Now, imagine that you have a point at coordinates `(x, y)`, with `x` and `y` values that do *not* necessarily fall on a point of the grid: they are random coordinates, but must fall within the bounds of `X` and `Y`. If you wanted to assign a value to that random point, equal to that of the nearest point in the `data` array, you would use a nearest interpolator. In Python, you can code it like the following.

```
import numpy as np
from scipy.interpolate import RegularGridInterpolator

X = np.arange(0., 2*np.pi, 2*np.pi/5)  # from 0 to 2pi (excluded), with 5 points
Y = np.linspace(0., np.pi, 3)  # from 0 to pi (included) with 3 points
Yg, Xg = np.meshgrid(Y, X)  # 2D grids to create data wit correct shape
data = np.sin(Xg*Yg)  # creating some data

# create nearest interpolator
interp = RegularGridInterpolator((X, Y), data, method='nearest')

x, y = 1., 3.  # these coordinates do not fall on a grid point above
nearest_value = interp([x, y])
```

You can check that it worked by adding the following diagnostic snippet:

```
print(nearest_value, data[1, 2])  # in this example data[1, 2] is the nearest
plt.scatter(Xg, Yg, c=data)
plt.colorbar()
plt.show()
```

Notes:

- *The statement that the random coordinates "must fall within the bounds of `X` and `Y`" will be non-trivial for the Earth map, because there is a sliver of Earth where a random point could fall slightly outside of the grid! You will need to find a fix. The functions `numpy.concatenate`, `numpy.vstack`, or similar might be of help to add points or rows to NumPy arrays.*
- *Distances on a sphere can be tricky, and what counts as "nearest" on a Cartesian projection of spherical coordinates is not as straightforward as this example, and choosing an incorrect method can lead to errors. However, in this lab, you can make the simplest choice at the expense of making small errors for the sake of not spending too much time fixing a problem that is likely negligible.*

Three functions for simulating photon scattering

```
def get_tau_step():
    """ Calculate how far a photon travels before it gets scattered.
    OUT: optical depth traveled """
    delta_tau = -np.log(np.random.random())
    return delta_tau

def emit_photon(tau_max):
```

```
    """ Emit a photon from the stellar core.
    IN: tau max is max optical depth
    OUT:
      tau: optical depth at which the photon is created
      mu: directional cosine of the photon emitted """
    tau = tau_max
    delta_tau = get_tau_step()
    mu = np.random.random()
    return tau - delta_tau*mu, mu

def scatter_photon(tau):
    """ Scatter a photon.
    IN: tau, optical depth of the atmosphere
    OUT:
      tau: new optical depth
      mu: directional cosine of the photon scattered """
    delta_tau = get_tau_step()
    mu = 2*np.random.random()-1  # sample mu uniformly from -1 to 1
    tau = tau + delta_tau*mu
    return tau, mu
```

Questions

1. **[40%] Random points on the surface of the Earth (based on Newman 10.12)**

   Suppose you wish to calculate the land area on the Earth, but don't have the resources to map every location on the Earth. You could instead choose a random point on the surface of the Earth and determine if that point was land or ocean. If you know the total area of the Earth then you can calculate approximately the land area.

   To calculate this value correctly, you want to choose a location on the planet at random such that all points are equally likely to be chosen.

   Recall that in spherical coordinates (r,θ,ϕ), the element of solid angle is $\sin\theta\,\mathrm{d}\theta\,\mathrm{d}\phi$, and the total solid angle in a whole sphere is 4π. Hence the probability of our point falling in a particular element is

   $$p(\theta,\phi)\,\mathrm{d}\theta\,\mathrm{d}\phi = \frac{\sin\theta\,\mathrm{d}\theta\,\mathrm{d}\phi}{4\pi}. \tag{5}$$

   We can break this up into its θ-and ϕ-parts, thus,

   $$p(\theta,\phi)\,\mathrm{d}\theta\,\mathrm{d}\phi = \frac{\sin\theta\,\mathrm{d}\theta}{2} \times \frac{\mathrm{d}\phi}{2\pi} = p_1(\theta)\,\mathrm{d}\theta \times p_2(\phi)\,\mathrm{d}\phi. \tag{6}$$

   (a) What are the ranges of the variables θ and ϕ? Verify that the two distributions $p_1(\theta)$ and $p_2(\phi)$ are correctly normalized, i.e., that they integrate to 1 over the appropriate ranges. Find formulas for generating angles θ and ϕ drawn from the distributions $p_1(\theta)$ and $p_2(\phi)$.

   SUBMIT YOUR WRITTEN ANSWERS AND BRIEF DERIVATIONS

   (b) Write a function that generates a random θ and ϕ using the formulas you worked out. Use your program to generate $N = 5{,}000$ such random points and plot those points either on a two-dimensional (latitude vs longitude) plot or a 3D Cartesian plot.

SUBMIT THE PLOT

(c) The calculation of land fraction can be numerically calculated without too much difficulty. You can use the formula for the area of a sphere, i.e., 4π if you set radius to 1. Calculate, by numerical integration, the land fraction using the provided Earth map.

To do so, the file `Earth.npy` contains a binary bitpacked dataset of the Earth that (approximately) identifies water and land points.

SUBMIT YOUR WRITTEN ANSWER

(d) Using your random location generator, generate N random locations on the planet and calculate the fraction of the planet that is covered by land (as opposed to ocean or sea ice). One way of doing this is to randomly sample a location on your map and determine whether it was land or not, and calculate the ratio of land points you identified compared to the total number of samples.

Calculate, using random sampling, the land fraction using the same Earth map as above. Give your estimates with $N = 50, 500, 5000, 50000$. For the latter value, plot a map where all points over land are in one colour, and all points over water are in another.

SUBMIT YOUR CODE FOR THE WHOLE QUESTION, WRITTEN ANSWERS, AND PLOT

2. **[30%] Limb darkening of a star**

In order to avoid almost all subtleties of radiative transfer and corresponding approximate solutions, in this exercise, we will perform a Monte Carlo simulation to confirm the result of Eq. (4). The simulation will emit a photon from the stellar core and follow it until it leaves the atmosphere.

(a) The concept of radiation transfer is relatively simple: a photon is emitted, it travels a distance, and it scatters. To determine the final scattering angle of the photon (the angle at which it leaves the photosphere), we draw samples of optical depth and scattering angle from probability distributions. Thus, Monte Carlo methods are a good way to simulate this. Figure 2 shows a possible path of a photon, where a plane-parallel atmosphere is assumed. The scattering angle θ is the angle between the radial and photon direction, and for this problem, it is more convenient to express it as $\cos\theta$, which we will call μ.

Using the three given functions in the computational background as building blocks, write a function that, taking $\tau_{\max}$ as its input, emits a photon from the stellar core and follows its path until it leaves the atmosphere. At each step, update the optical depth of the photon and follow the photons until they have left the atmosphere (i.e., $\tau < 0$). Have your program print out the μ at last scattering and how many times the photon scattered. If a photon scatters back into the core, have your program release a new photon at the inner boundary.

NOTHING TO SUBMIT (YET)

(b) Using your function, test it for 10^5 photons with an optical depth of $\tau_{\max} = 10$, and save the final μ values of each photon. Make a histogram plot of $N(\mu)/N_1$, with $N_1 = N(\mu = 1)$, as a function of μ using `n_bins=20` as the number of bins.

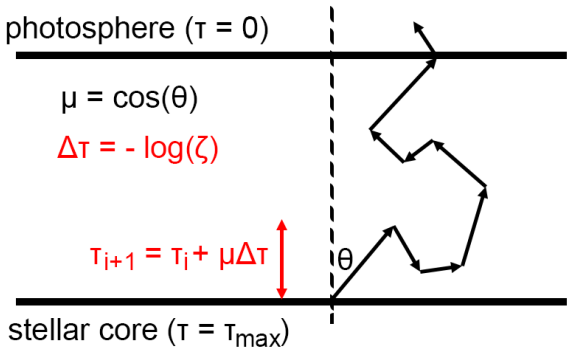

Figure 2: Random walk of a photon from the stellar core to the photosphere. The photon is emitted from the stellar core with some angle θ and scatters at some new optical depth τ_{i+1} for step i, and continues onward until one of the following happens: $\tau < 0$ where the photon has left the photosphere, or $\tau \geq \tau_{max}$ where the photon has scattered back into the stellar core. In the latter case, we emit a new photon to replace it. Since the probability that a photon travels an optical depth of τ without an interaction is $e^{-\tau}$, the optical depth is sampled from $\tau = -\log\zeta$, where ζ is a random number uniformaly sampled from 0 to 1.

In our simulation, we have calculated the number of photons leaving the atmosphere with respect to a surface perpendicular to the radial direction. This number is proportional to the specific intensity weighted with the projection angle μ, since the specific intensity $I(\mu)$ is defined with respect to unit projected area, namely,

$$N(\mu)\,\mathrm{d}\mu \propto I(\mu)\mu\,\mathrm{d}\mu. \tag{7}$$

Therefore, we can take specific intensity $I(\mu) \propto N(\mu)/\mu$, which is obtained from the number of photons divided by the appropriate average of μ, i.e., centred at the middle of the corresponding bin.

Write a function to calculate $I(\mu)/I_1$ and compare it with the analytic expression from Equation (4). Use 10^5 photons and $\tau_{\max} = 10$ as before, and derive the limb-darkening coefficients from a linear regression to your results.

You may use `scipy.optimize.curve_fit` to determine the coefficients. State the coefficients (either on your plot or in your report) and plot both the analytic fit and the best fit from your results. Relate your results back to limb-darkening (e.g., where is specific intensity greatest and how does affect what an observer would see?).

SUBMIT YOUR CODE, PLOTS, AND EXPLANATORY NOTES.

(c) Limiting case of $\tau_{\max} \ll 1$. To convince yourself that limb-darkening is the consequence of a multitude of scattering effects, reduce $\tau_{\max}$ to a very small value ($\tau = 10^{-4}$). Which angular distribution do you expect for the specific intensity? Does your simulation verify your expectation? Make the same plots as in the previous part and use them to explain your results.

SUBMIT YOUR CODE (YOU MAY COMBINE WITH THE PREVIOUS CODE), PLOTS, AND

EXPLANATORY NOTES.

3. **[30%] Importance sampling**

(a) The integral

$$\int_0^1 \frac{x^{-1/2}}{1+e^x}\,dx \tag{8}$$

(eqn. 10.34 of the textbook) is an integral whose integrand has a divergence. Section 10.2.3 discusses using importance sampling to evaluate this integral. To learn about the regular mean value method and the importance sampling methods, evaluate the integral using both methods with 10,000 sample points, then repeat this calculation 100 times for each method. For the importance sampling approach, use the weighting function $w(x) = x^{-1/2}$, which removes the singularity. You will need to determine the transformation needed to non-uniformly sample your region. The probability distribution you will be drawing from is

$$p(x) = \frac{1}{2\sqrt{x}}. \tag{9}$$

Pages 458-459 of the text explain how to find the transformation for a given probability distribution, to help you understand where this $p(x)$ comes from. Make a histogram of the resulting integral values using 10 bins from 0.80 to 0.88. For an array of values (say it's called `I`), you do this with the command:

```
import matplotlib.pyplot as plt
# ...
plt.hist(I, 10, range=[0.8, 0.88])
plt.show()
```

Comment on what your histograms from the mean value method and the importance sampling method tell you about each method.

HAND CODE, PLOTS, EXPLANATORY NOTES.

(b) Importance sampling also helps evaluate rapidly varying integrands even if they aren't singular. For example, consider the integral

$$\int_0^{10} e^{-2|x-5|}\,dx. \tag{10}$$

This is a function that is sharply peaked near $x = 5$, and we can easily find its exact value. If we calculate it using points uniformly distributed on the interval $0 \le x \le 10$ we will be using a lot of points that don't contribute to the expectation value. To sample more points near $x = 5$ in order to increase the precision of the sampling method, choose an importance function that is Gaussian of the form

$$w(x) = \frac{1}{\sqrt{2\pi}} e^{-(x-5)^2/2} \tag{11}$$

that will lead you to draw from the normal distribution

$$p(x) = \frac{1}{\sqrt{2\pi}} e^{-(x-5)^2/2} \tag{12}$$

with mean of 5 and standard deviation of 1. You can use `numpy.random.normal` to do this.

Repeat part 3a using the mean value and importance sampling methods for this integral, and comment on the results.

HAND CODE, PLOTS, EXPLANATORY NOTES.