STAT HW1

Yifan Zhou

February 12, 2016

1. Random Number Generators

In class, we talked about many variants of random number generators and explored their properties in terms of their (i) period, (ii) clustering, (iii) efficiency, and (iv) portability. In your computer language and/or algorithm library of choice, choose the random generator that you will be using for the rest of the class. Search the literature for articles that explore the above properties of your generator and summarize your findings in a couple of paragraphs. Make sure to include references and, if you find it necessary, figures.

Python's random module (as well as Numpy.random module) uses Mersenne Twister (MT) generators [Matsumoto and Nishimura, 1998]. According to Matsumoto and Nishimura [1998], the MT generator has a very long period of $2^{19937} - 1$. In terms of clustering, its k- distribution is 623-dimensional equidistribution up to 32-bit. The computational complexity is $O(p^2)$. The computational speed is relative slow and the state space occupation of 2.5 KiB is relative large. According to Wikipedia, this algorithm is the mostly implemented method for pseudo-random number generation, and has been adopted in R, Python, Ruby, PHP, CMU Common Lisp, Embeddable Common Lisp, Steel Bank Common Lisp, Free Pascal, GLib, SageMath, Maple, MATLAB, GAUSS, IDL, Julia, Scilab, Stata, GNU Octave, the GNU Scientific Library, the GNU Multiple Precision Arithmetic Library, and Microsoft Visual C++.

2. Designing surveys

The Kepler mission has been observing a very large number of stars in a small patch in the sky and is making a very reliable measurement of the occurrence rate of planets around solar type

stars (see Batalha, N. M. 2014, Proceedings of the National Academy of Science, 111, 12647; arXiv:1409.1904). For the purposes of this homework problem, we will assume that the occurrence rate of planets with radii between one and two Earth radii has been measured to a very high accuracy and it is equal to 10% (i.e., 10% of solar type stars harbor such planets; the actual rate is consistent with this number but has some considerable uncertainty).
You are designing a survey of solar type stars in a different patch of the sky to find the same type of planets. How many stars would you need to observe in order to have a 90% likelihood that you will find at least 30 planets with radii between one and two Earth radii? What if you want to have a 99% likelihood?
For a sample with a size of N stars, the probability to find at n earth like planets is a binomial distribution, which is

$$P(n) = \binom{N}{n} p^n (1-p)^{N-n} \tag{2.1}$$

Thus the likelihood to find at least 30 planets is

$$P(n \geq 30) = \sum_{n=30}^{N} \binom{N}{n} p^n (1-p)^{N-n} \tag{2.2}$$

For likelihood of 90%, $P(n \geq 30) \geq 90\%$, get

$$N \geq 368 \tag{2.3}$$

For likelihood of 99%, $P(n \geq 30) \geq 99\%$, get

$$N \geq 435 \tag{2.4}$$

3. Blackbody distribution

The energy distribution of the number of photons that follow the blackbody distribution is given by

$$f(\epsilon; T)\mathrm{d}\epsilon = C \frac{\epsilon^2 \mathrm{d}\epsilon}{\exp(\epsilon/kT) - 1} \tag{3.1}$$

where ϵ is the photon energy, C is a normalization constant, T is the temperature of the distribution and k is the Boltzmann constant. (Note that this is the distribution of the number of photons and not of the radiation energy density, which is the expression that you are probably more familiar with). Our goal is to generate an ensemble of photons with energies drawn from this distribution and in a range (ϵ_1, ϵ_2), using the rejection method.

(i). Start by making a change of variables

$$\epsilon' = \frac{\epsilon}{kT} \tag{3.2}$$

in order to remove any parameters in the distribution.

$$f(\epsilon;T)\mathrm{d}\epsilon' kT = C(kT)^3 \frac{\epsilon'^2 \mathrm{d}\epsilon'}{\exp(\epsilon') - 1} \tag{3.3}$$

$$f(\epsilon';T)\mathrm{d}\epsilon' = C' \frac{\epsilon'^2 \mathrm{d}\epsilon'}{\exp(\epsilon') - 1} \tag{3.4}$$

(ii). Find the energy ϵ_0 for which this distribution has a maximum. You will need to do this numerically.

$$\left.\frac{\mathrm{d}f(\epsilon';T)}{\mathrm{d}\epsilon'}\right|_{\epsilon'=\epsilon'_0} = C' \frac{2\epsilon'_0\left[\exp(\epsilon'_0) - 1\right] - \epsilon'^2_0 \exp(\epsilon')}{\left[\exp(\epsilon') - 1\right]^2} = 0 \tag{3.5}$$

$$\implies 2\exp(\epsilon'_0) - \exp(\epsilon'_0)\epsilon'_0 - 2 = 0 \tag{3.6}$$

from numerical calculation $\epsilon'_0 = 1.59362$

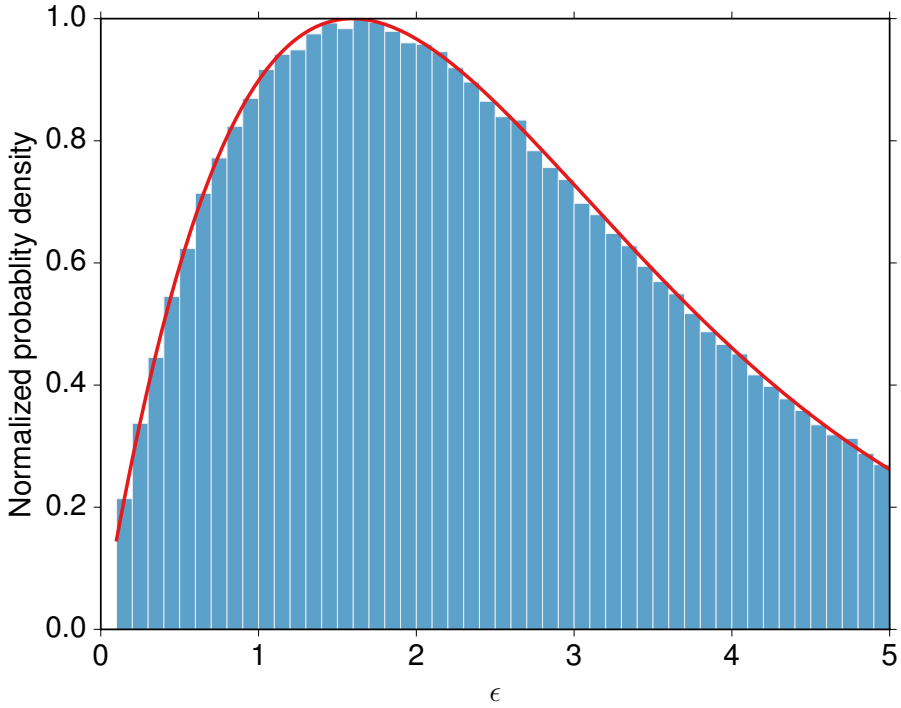
(iii). Normalize your blackbody distribution such that its maximum value is equal to unity, i.e., consider the distribution

$$\begin{aligned} f'(\epsilon)\mathrm{d}\epsilon &= \frac{C}{f(\epsilon'_0)} \frac{\epsilon'^2 \mathrm{d}\epsilon'}{\exp(\epsilon') - 1} \\ &= \frac{\exp(\epsilon'_0) - 1}{\exp(\epsilon') - 1} \left(\frac{\epsilon'}{\epsilon'_0}\right)^2 \end{aligned} \tag{3.7}$$

(iv). Use your random number generator to draw a random number between $\epsilon'_1 = 0.1$ and $\epsilon'_2 = 5$. This will be the energy ϵ' of your photon.

(v). Use your random number generator to draw a second random number between 0 and 1. If this number is larger than $f'(\epsilon')$ then reject this photon, otherwise accept it and repeat the last two steps.

(vi). Plot the distribution you just generated and compare it to the blackbody distribution to verify your result. What fraction of your initial photon energies was acceptable (i.e., how efficient was your rejection algorithm)?.



For §(iv) to (vi),
the efficiency is 67.8%.
with following code

```
import numpy as np
import matplotlib.pyplot as plt

def BlackBody(epsi):
    return epsi**2/(np.exp(epsi) -1)

epsi0 = 1.59362 # result of numerical calculation

def normedBlackBody(epsi):
    """normalize the planck function so that the maximum function
    value is one"""
    return BlackBody(epsi) / BlackBody(epsi0)

e_min = 0.1
e_max = 5.0
N_photon = int(1e6)
```

```
# generate photon with random energy between e_min and e_max
photons = np.random.uniform(e_min, e_max, N_photon)

photons_BB = normedBlackBody(photons)
# second random var to judge whether reject or keep photon
randBB = np.random.uniform(0, 1, N_photon)
photons_eff = photons[randBB <= photons_BB]

## make the plot
fig = plt.figure()
ax = fig.add_subplot(111)
hist, bins = np.histogram(photons_eff, bins=np.linspace(e_min, e_max, 50))
dbin = bins[1] - bins[0] # bin width
# matplotbar bar plot default align to the left edge
ax.bar(bins[:-1], hist/hist.max(), width=dbin, alpha=0.8)
BB_samp = np.linspace(e_min, e_max, 1000)
ax.plot(BB_samp, normedBlackBody(BB_samp), lw=2)
ax.set_xlabel('$\epsilon$')
ax.set_ylabel('Normalized probablity density')
```

References

Makoto Matsumoto and Takuji Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul.*, 8(1):3–30, January 1998. ISSN 1049-3301. doi: 10.1145/272991.272995. URL http://doi.acm.org/10.1145/272991.272995.