PHYS-512: Problem Set 7

Simon Briesenick

November 19, 2022

1. I plotted the triplets on an interactive 3D plot and adjusted the viewing angle manually until I could see an alignment of points into planes. The viewing angle is not perfect, so that the planes are not visible for their entire range. One can see however that there are several families of planes visible here (faint crossing over of *denser* planes around the center of the figure).

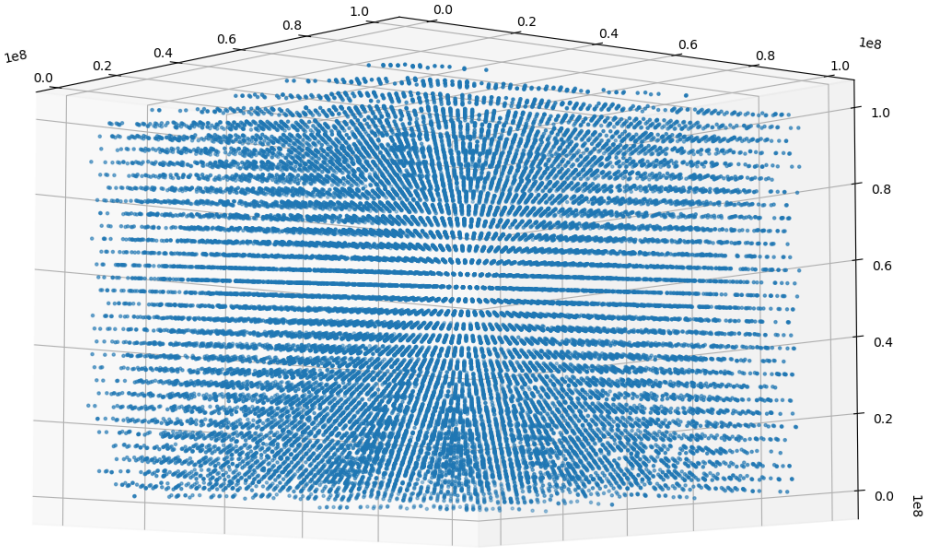

Figure 1: When viewed from this angle, one can observe several planes appear.

2. To implement a rejection method to generate a decaying exponential distribution, one needs a strictly larger (and or equal) distribution that is integrable to obtain the CDF. I.e. if $g(x) = e^{-x}$ and $f_i(x)$ is a test function, then

$$f_i(x) - g(x) \geq 0 \tag{1}$$

needs to hold for all values of x. According to Wolfram Alpha, this is true for the Lorentzian distribution

$$f_{\text{Lorentz}} = \frac{C}{1+x^2}. \tag{2}$$

A Gaussian

$$f_{\text{Gauss}} = Ce^{-ax^2} \tag{3}$$

will only not intersect with $g(x)$ if it is strictly smaller then g. This is clear by comparing the decaying exponential e^{-x} with a Gaussian that can be written as $e^{-x}\,(Ce^{-ax})$. Whatever constants (C, a) we choose, there will always be a value for x, such that the bracketed term will fall below zero, which means that the Gaussian will eventually intersect with the target function. So this is not a suitable bounding function. A power law like distribution

$$f_{\text{Power}} = Cx^{-\alpha} \tag{4}$$

is another suitable candidate, especially if combined with a lateral translation to improve the efficiency of drawing samples. I will start with a simple Lorentzian with $C = 1$. The enveloping function/distribution is drawn from a uniform distribution similar to the in-class example. The inverse of the CDF will essentially sample through all possible values and is weighted with the relative probabilities. I.e. in the inverse most samples from the uniform distribution are mapped to values around the central maximum of the Lorentzian.

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

N = int(1e6) # number of samples we will draw
M = 1 # scaling constant, such that Mf(x) >= g(x)
target = np.exp #target distribution
custom_rand = []
x = np.random.uniform(low=0, high=np.pi/2, size=N)
trials = np.tan(np.pi*x)

# trials = np.random.standard_cauchy(size=N)
trials = trials[trials >= 0]

u = np.random.rand(size=N))

def lorentz(x):
    return 1/(1+x**2)

target_dist = target(-trials)
scaled_lorentz = M*lorentz(trials)

s = target_dist/scaled_lorentz

keep = trials[u < s]
```

The sampling efficiency for this method is at about 64%. A similar implementation of a "simple" power-law like distribution with $C = 1$ results in an efficiency of only about 37% (see Fig. 3a). To optimize this approach, I've defined the constant `C` in the difference

$$h(x) = C f_{\text{Power}}(x) - g(x) \tag{5}$$

i.e.

$$Cx^{-\alpha} - e^{-x} \tag{6}$$

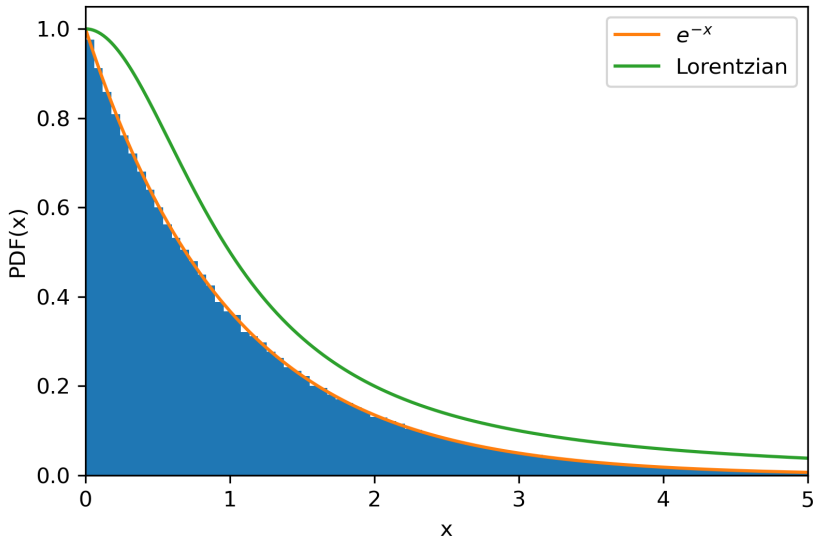

Figure 2: Comparison between the decaying exponential distribution and the Lorentz distribution from which it was sampled by the rejection method.

to be such that $h(x)$ has no real roots (i.e. that the two distributions do not intersect). The roots for $h(x)$ are at

$$x = -\alpha W\left[\frac{-(\frac{1}{C})^{-1/\alpha}}{\alpha}\right] \tag{7}$$

where $W[u]$ is the Lambert W function. It is only defined for inputs $u \geq -\frac{1}{e}$. It follows that I can pick C to be

```
# power = \alpha
M = (power/np.e)**power + 1e-6
```

With this modification, I've tried several different values for α and compared their efficiencies. Without shifting the distribution by any amount, I could reach efficiencies as large as 70%. This comes pretty close to the limit for a power law distribution: The highest efficiency overall I could reach was with a modified Lorentzian with reduced width. In this case, the efficiency reaches almost 80% (see Fig. 5).

3. The ratio-of-uniforms method works by drawing (u, v) from two uniform distributions $U(0, a)$ and $U(b, c)$, respectively. The constants are defined as

$$a = \sup_x \sqrt{g(x)} \qquad b = -\sup_x x\sqrt{g(x)} \qquad c = \sup_x x\sqrt{g(x)}$$

where $g(x)$ is the normalized exponentially decaying distribution we want (like above). It follows that the constants are

$$a = 1 \qquad b = -\frac{2}{e} \qquad c = \frac{2}{e}.$$

The function value of the ratio of the uniforms is then compared to the square of u

$$u^2 < g\left(\frac{u}{v}\right) \tag{8}$$

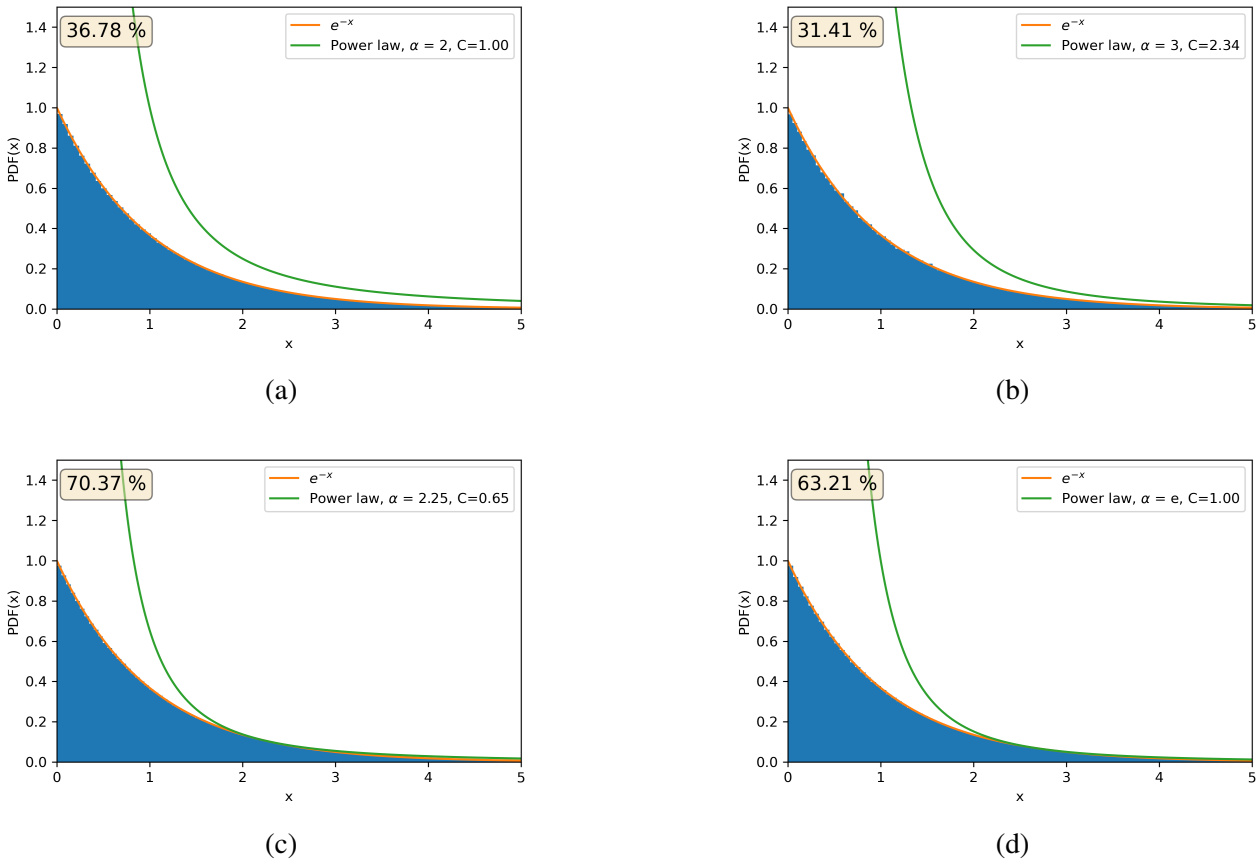

Figure 3: Examples for a power law distribution, $f(x) = Cx^{-\alpha}$.

and if the RHS is larger then the LHS the ratio is "accepted" as a valid draw from the target distribution. The implementation is shown below and the result in Fig. 6. It is noteworthy that the efficiency of this method is defined as

$$\frac{1}{2a(b-c)} = \frac{e}{8} \approx 0.34 \tag{9}$$

which is also what I have found here.

```
import numpy as np
from matplotlib import pyplot as plt
from scipy.stats import expon

# ratio of uniforms method

N = int(1e7)

u = np.random.uniform(low=0, high=1, size = N)
v = np.random.uniform(low=-2/np.e, high = 2/np.e, size = N)

ratio = v/u
f = expon.pdf(ratio)
x = ratio[u < np.sqrt(f)]
```

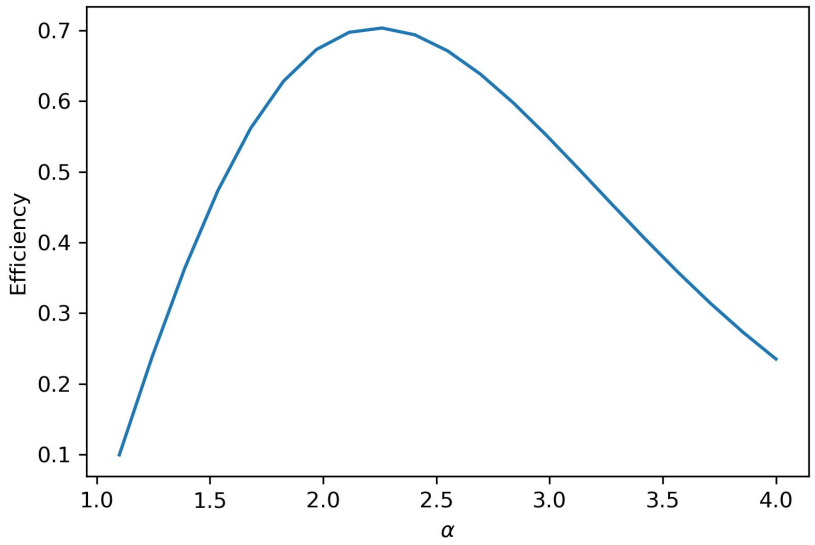

Figure 4: Efficiency in the rejection method for a power-law distribution with α as variable between 1 and 4.

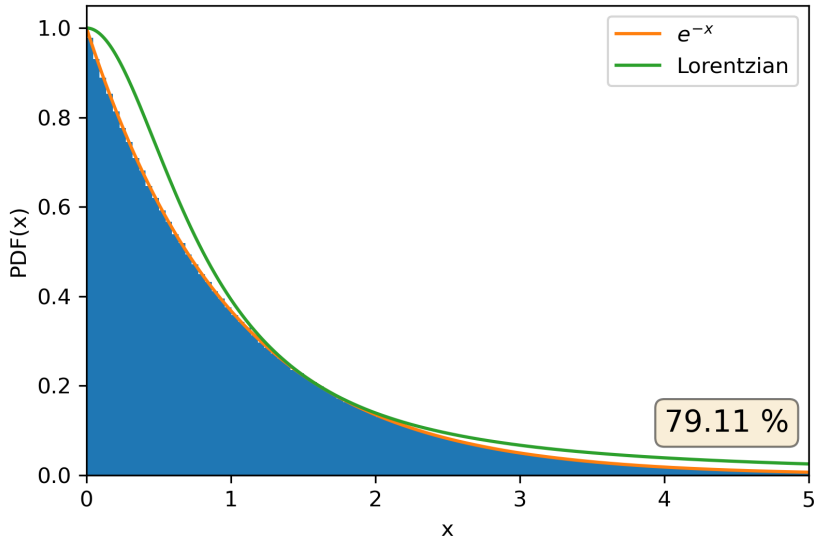

Figure 5: Rejection sampling with decaying exponential with reduced with.

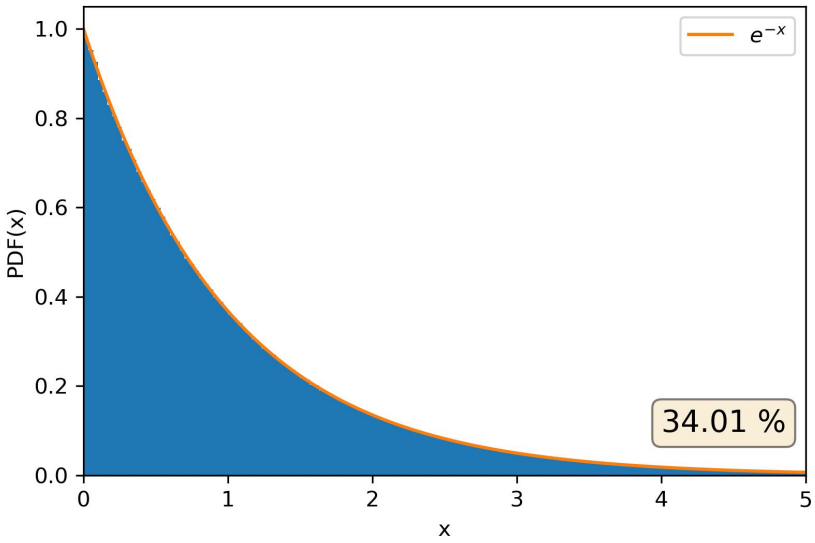

Figure 6: Ratio of uniforms method