Assignment 2

1. As part of a software for modelling defects on fabricated silicon wafers you need to simulate a uniform distribution inside a circle. (You can choose the radius to be $R = 1$.) In each of the cases below, generate 2000 points $(x,\ y)$ and plot them. For comparison, you can also generate more points, for example 20 000 points; in doing so, you see why such geometric distributions sometimes need to be checked with a small number of points.
   a) (w = 2) First implement this using **the rejection method** by thinking of the circle being enveloped by a square. Generate 2000 points $(x,\ y)$ and plot them.
   b) (w = 2) You want to make the simulation computationally more effective and not to reject any points, so you **generate random points inside the circle by using polar coordinates**, $x = r\cos(2\pi\theta)$, $y = r\sin(2\pi\theta)$, and drawing r and θ from uniform distributions. **Generate 2000 points $(\boldsymbol{x},\ \boldsymbol{y})$ and plot them. Explain the outcome and the reason for it.**
   c) Simulation of the circular uniform distribution by **the inverse distribution method**: Generate $(x,\ y)$ from polar coordinates again drawing θ from uniform distribution but applying a proper transformation for the radius $r \in (0,\ R)$: It is straightforward to see that the correct pdf is of the form $p(r) = C2\pi r$, where C is a constant to be determined from the condition $F(r = R) = 1$. F is the cumulative distribution for r**.**
      (i) (w = 2) **Determine the proper transformation for generating $\boldsymbol{r}$. Show the derivation of this transformation in your answer.**
      (ii) (w = 2) **Implement an algorithm and generate and plot 2000 points $(\boldsymbol{x},\ \boldsymbol{y})$ using this method.**
   d) (w = 1) **Compare the outcomes of b) and c). Explain the difference**. (Hint: See the explanation of the method and the examples after that in Lecture 2.)

2. This drill hopefully shows why you're better off using log binning and scaling for many processes generating strongly skewed distributions, especially when data is scarce.

   Power-law distributions are common in nature and society. Here, we simulate the distribution that results when brittle material (e.g. rock) fragments in two dimensions such that the amount of energy (e.g. in impact) used is barely sufficient to cause the whole material volume to fracture. The mass distribution (PMF) of fragments, that is, the number of fragments # of mass x, is of the logarithmic (power-law) form $\#(x) \propto x^{-3/2}\ (> 0)$. Stochastic discrete processes are often treated as continuous. Accordingly, we take the pdf to be $f(x) = C\,x^{-3/2}$.
   a) (w = 2) Determine C for the support (range) $x \in [1, \infty)$. Then determine the corresponding distribution $F(x)$ and the inverse transformation $x = F^{-1}(y)$ that is needed for sampling from it using the inverse distribution method and $y \in [0, 1)$. **Show the derivation and $\boldsymbol{x} = \boldsymbol{F^{-1}(y)}$.**
      Power-law distributions typically have exponential cut-offs, $\#(x) \propto x^{-3/2}e^{-x/D}$, which in effect limits the range of the distribution. We ignored this to avoid complications in derivations. Accordingly, in order to set an appropriate range in x, we sample from $y \in (0, A]$. So, substitute y with y/A in the transformation and use that in the following

simulations. **Write down $F^{-1}(y/A)$ for yourself** so you can write the algorithm for later use.

(Behind this pdf there is a stochastic process of multiplicative nature reminiscent of the one in Assignment 1, Problem 2 c), but in what follows we just directly simulate the resulting distribution by the inverse distribution method. This serves the same purpose, namely, it generates simulated "data", something one often needs to do.)

Generate data of 1000 points. (This is the scarce data part – in this case, explode something to get fragments once, and that's it.) Use the scaled $F^{-1}(y/A)$, where $A = 200000$, to simulate the distribution. For numerical reasons, draw y from $U(0, B]$, where $B < A$. Set $B = 190000$. This sufficiently covers the scaled range (0, 200000]. (The numerical values of A and B have no deeper meaning here. They just serve the purpose of setting a useful range for sampling. There is a numerical reason for setting $B < A$.) If you can do the sampling equivalently to what is described above without resorting to scaling, that is just as well.

In what follows, generate the bins and plot the midpoints as shown on the last page of Lecture 2. Use e.g. **hist**-command in **matplotlib** for histograms. For midpoint plots, you can write your own bit of code – or use a library function if there is one. **Use 30 bins**. **Choose ranges for the axis appropriately so that data can be seen.**

b) (w = 2) Do **linear binning** to view the pdf of the generated data. Plot the binned data **as a histogram** of the bins **in linear coordinates.** Then plot the binned data **as a histogram** of the bins **in logarithmic coordinates**. **In all plots include also the line $x^{-3/2}$** possibly multiplied by an appropriate coefficient for the line to "sweep" close to the pdf (this is to make checking in peergrade easier).

c) (w = 2) Determine the midpoints of the bins and **plot the linearly binned data values at these midpoints including the line $x^{-3/2}$ in logarithmic coordinates,** possibly multiplied by an appropriate coefficient for the line to "sweep" close to the pdf.

d) (w = 2) Do **logarithmic binning** of the pdf. **Plot the binned data as midpoints of the bins in logarithmic coordinates including the line $x^{-3/2}$** possibly multiplied by an appropriate coefficient for the line to "sweep" close to the pdf.

e) (w = 1) (The advantage of logarithmic over linear coordinates is obvious. It should also be obvious that midpoint plots are superior to histograms for discerning functional forms.) Looking at your binned plots in c) and d), **what did you gain by doing logarithmic instead of linear binning? (2 – 3 advantages expected.)**

f) (w < 1) One could try to simulate the pdf by rejection method. (You don't need to do the simulation.) Would this be slower or faster than the inverse distribution method used here? Justify your answer.