How many steps does an average self-avoiding walk last?

Assignments related to the Lecture "Basic data analys: A guided tour using `python`"

O. Melchert, C. Norrenbrock

General topic

The subsequent assignments are related to the self-trapping effect that induces a strong localization of (genuine) self-avoiding random walkers on the square lattice, as discussed in the article "An average self-avoiding random walk on the square lattice lasts 71 steps" by S. Hemmer and P. C. Hemmer, published in *J. Chem. Phys.* **81**, 584 (1984).

Motivation

A (genuine) self-avoiding random walk (SAW) on a square lattice might end up in a trapped state from which it cannot proceed further (see Fig. 1). For small walk-lengths, the probability $t(n)$ for ending up in a trapped state after n steps can be found exactly. E.g. one finds $t(6)=0$, $t(7)=2/729$. For larger values of n, a "combinatorial explosion" makes it hard to analyze the possible paths by analytic means. Consequently, an analysis of $t(n)$ for larger values of n requires Monte Carlo simulations.

Problems

Copy the `.tar`-archive `ASSIGNMENT_LENGTH_SAW.tar` from the shared folder `share/Melchert` to your computer and untar the archive by typing

```
$ tar -xvf ASSIGNMENT_LENGTH_SAW.tar
```

Enter the unpacked folder and have a look at the script `saw_fragment.py` that implements the code for the simulation of SAWs on a $2D$ square lattice. The class `Walker` generates an instance of a SAW and reports its length n and the geometric distance R between its starting and trapping point. Use the above script to generate a number of

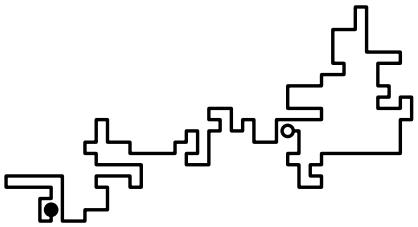

Figure 1: The figure illustrates a single self avoiding walk on a $2D$ square lattice that got trapped after $n = 161$ steps. The open (closed) circle indicates the starting point (end point) of the walk.

$N = 10^5$ SAWs and store the resulting values for n and R in a file. To do so, use the command line and write:

```
$ python saw_fragment.py 0 100000 > saw_2dSquare_s0t100000.dat
```

By completing the following tasks you will put under scrutiny the statistical properties of the SAWs.

Task 1: Complete the script `Task1_fragment.py` that handles the analysis of the walk lengths in order to perform the following analysis:

- Determine the average SAW length $\mathrm{av}(n)$ and the average geometric distance $\mathrm{av}(R)$ between the starting and trapping point.
- What are the median walk length $\mathrm{med}(n)$, the median absolute deviation $\mathrm{mad}(x)$ (i.e. a robust estimator for the sample variability) and the longest SAW length $n_{\max}$ encountered?
- Compute the most probable SAW length n' and estimate an error for that observable using bootstrap resampling considering 20 resampled datasets.

Hint: for an ordered sample $x \equiv \{x_i\}_{i=1}^{N}$ of samplesize N (x_1 (x_N) = smallest (largest) encountered value), the median is defined as

$$\mathrm{med}(x) = \begin{cases} 0.5 \times (x_{N/2} + x_{N/2+1}), & \text{if } N \text{ is even,} \\ x_{(N+1)/2}, & \text{if } N \text{ is odd.} \end{cases} \tag{1}$$

Using the median, the median absolute deviation (i.e. a robust measure of the variability in the dataset) is given by

$$\mathrm{mad}(x) = \mathrm{med}(\{\mathrm{abs}(x_i - \mathrm{med}(x))\}) \tag{2}$$

Note that the absolute value of, say, a variable x, can be computed via the function $\mathrm{abs}(x)$, right away.

Task 2: The function `basicStatistics` described in the lecture notes computes the average of a sample by summing up the elements and dividing by the sample size. Now, given a probability mass function (PMF), it is possible to compute the average according to:

$$\mathrm{av}(x) = \sum_i p_i x_i \tag{3}$$

where x_i signifies the unique values in the PMF, and where $p_i = p_X(X = x_i)$. Consequently, the *uncorrected* variance can be computed as:

$$\mathrm{uVar}(x) = \sum_i p_i (x_i - \mu)^2 \tag{4}$$

Consider the script `Task2_fragment.py` and complete the two functions `pmfAv` and `pmfUVar` that take a PMF (constructed from the raw data by means of the function `getPmf`) and compute the mean and variance. In order to test both functions use the SAW raw data to check whether they yield the same results as obtained by means of the function `basicStatistics`.

Task 3: (Bonus) Previous simulations suggest an approximate expression $t(n) \sim (n - \Delta)^{\alpha} \exp(-n/\beta)$ for the probability of ending up in a trapped state after n steps. Perform a fit (e.g. using `Gnuplot`) to determine the parameters Δ, α and β. What is the meaning of β?

Task 4: (Bonus) Perform a similar analysis for the lengths of SAWs on a $2D$ 8-neighbour lattice (amend the script `saw_fragment.py` accordingly). How does $t(n)$ change qualitatively and what are the values for $\mathrm{av}(n)$, $n_{\max}$, $\mathrm{av}(R)$, and β in this cases?