Computational Physics II — Project 1

Jan O. Haerter

February 2, 2022

Hand-in date: 6 a.m., Thursday, Feb 18, 2022
to be uploaded to https://moodle.jacobs-university.de

In total, **100 points** can be achieved by solving the problems below (see annotation).

---

1 Pseudo random numbers and cryptography

- Construct a linear congruential random number generator and use the parameters $(a, b, c, x_0) = (57, 1, 256, 10)$ as a pedagogical example. (**5 pts**)
- Determine the period of this random number generator and plot pairs (x_i, x_{i+1}) on a 2D scatterplot. Describe the pattern and comment on possible autocorrelations. (**5 pts**)
- Construct another random number generator, e.g., again using the linear congruent generator above, but with a different set of parameters (or developing another method of your choice). Apply several tests on your random number generator to check for uniformity and correlation. For uniformity, apply a Kolmogorov-Smirnov test to check, whether your random number generator is consistent with a uniform distribution between 0 and 1. Use the autocorrelation function to quantify independence between samples. Examine the distribution function of the resulting correlations: are they consistent with a Gaussian distribution? Further, choose any of the other correlation tests mentioned in the lecture notes, e.g. a gap or poker test. (*Hint*: For both of these tests, define a window $[a, b]$ with $0 \leq a < b \leq 1$, and the expected probability to hit this window $p = b - a$.) Which theoretical distribution would you expect in the case of the gap test, if your samples were truly random? (**15 pts**)
- Using your random number generator, construct a one time pad using the English alphabet (26 letters as well as punctuation ”,” , ”.” and space). For this, map each character onto an integer. In Python, you can use

```
ord(x)
```

which maps a given character x on the respective index in the unicode alphabet. E.g. $x =$ "p" should give you the result "112". Next convert each of these integers into an eight-bit binary number, e.g. $112_{10} = 01110000_2$. Describe how you constructed the one-time pad (in a short paragraph, using the English characters mentioned above), apply your one time pad to encrypt this paragraph, and attach the encrypted paragraph (a bit string) to your hand in. Provide your random number generator (which needs to generate binary numbers) along with the seed as a piece of code, allowing to reconstruct the key required for encoding. (*Note*: In principle, this "key" would of course have to be sent via a different, safe, channel.) (**25 pts**)

2 Random walks and first passage time

- Consider a 1d random walk with $p = q = 1/2$. Use your random number generator (or a built in one) to produce a large number of realizations of random walks, each starting from the origin. For a number of unit steps n compute the histogram of displacements x_n from the origin, that is, the probability density function $PDF(x_n)$. Plot $PDF(x_n)$ for various values of n on a single graph and describe your findings. (**20 pts**)
- for each n (or a sufficient subset of all n), compute the mean $\langle x_n \rangle$ and variance $\langle x_n^2 \rangle - \langle x_n \rangle^2$. Plot both as a function of n and discuss the findings. (**10 pts**)
- Modify your algorithm, so that the random walk will start at the origin and terminate when reaching a barrier at $x_0 = 10$. Note that there is no further barrier, e.g., the walker can move freely at any $x < x_0$. Store the number of steps n taken to first reach the barrier at x_0. Repeat such walks (always starting at the origin) for a large number $w \sim \mathscr{O}(10^3)$ of walkers, yielding a corresponding number w of arrival times n. Plot the histogram of these arrival times on a log-log scale. Repeat the exercise with another value of x_0 and discuss your findings. *Hint*: You will likely need to restrict the total number of steps a walker can take (to some reasonably large number). (**20 pts**)

General remarks for all Projects. You will have to (i) analyze the problem, (ii) select an algorithm (if not specified), (iii) write a Python program, (iv) run the program, (v) visualize the data numerical data, and (vi) extract an answer to the physics question from the data. Which checks did you perform to validate the code? State the results you got for these tests. For each project you will submit a short report describing the physics problem, your way of attacking it, and the results you obtained. Provide the documented Python code in such a form that we can run the code. A Jupyter Notebook including the code and report is fine but not necessary.