Physik auf dem Computer **SS 2013**

Worksheet 10: Random numbers

June 25, 2013

General Remarks

- Deadline is **Tuesday, 2nd July 2013, 10:00**
- On this worksheet, you can achieve a maximum of 10 points.
- To hand in your solutions, send an email to
  - Olaf (`olenz@icp.uni-stuttgart.de`; Wednesday, 14:00–15:30)
  - Elena (`minina@icp.uni-stuttgart.de`; Wednesday, 15:45–17:15)
  - Tobias (`richter@icp.uni-stuttgart.de`; Friday, 15:45–17:15)
- Attach all required files to the mailing. If asked to write a program, attach the *source code* of the program. If asked for a text, send it as PDF or in the text format. We will *not* accept MS Word files!
- The worksheets are to be solved in groups of two or three people. We will not accept hand-in-exercises that only have a single name on it.
- The tutorials take place in the CIP-Pool of the Institute for Computational Physics (ICP) in Allmandring 3.

Task 10.1 (5 points): Quality analysis of random number sequences

The file random.npy contains 5 data sets of 100000 apparently random numbers in the interval $[0, 1]$. Copy this file to your home directory. Actually only one of those sequences was generated by a good random number generator. The others are in some way inconsistent.

With the following command you can load the data in python:

```
r1, r2, r3, r4, r5 = numpy.load('random.npy')
```

- Task 10.1.1 (1 point) Generate and plot histograms of the probability distribution for the different sequences by using the python function `matplotlib.pyplot.hist` with 100 *bins*. What shape of the distribution do you expect?
- 10.1.2 (1 point) Calculate the means for blocks of 10 consecutive random numbers and generate a histogram for the mean values (statistical χ^2-test). Plot this together with the expected distribution of a real random number sequence.
- 10.1.3 (1 point) Fourier transform the sequences with `numpy.fft.rfft` and plot the absolute values of the Fourier coefficients semilogarithmically (`semilogy`).

- 10.1.4 (1 point) Write a Python script that generates a random number sequence with 100000 elements that fails the χ^2-test, *i.e.* doesn't show the expected distribution for real random numbers, but does pass the other tests.
- 10.1.5 (1 point) In order to find out, which of the still seemingly real random number sequences is indeed made by a good random number generator, we will investigate the *correlation of triplets* in the sequence: $C(p,k) = \sum_{i=0}^{N-p} r_i r_{i-p} r_{i-k}$. For a sequence of truly randomly distributed numbers the triplet correlation evaluates to $C(p,k) = 1/2^3$, if p and k are different positive integers. Calculate the triplet correlation and the error $|C(p,k) - 1/2^3|$ for the given data sets with $p = 250$ and the interval $k = 1(1)249$. Make a plot for the deviation of the triplet correlation from the expected value.

  Hint This seemingly arbitrarily chosen test shows, that even good random number sequences can contain unwanted statistical correlations. One of the data sets was generated by *r250*, which was considered a very reliable and good random number generator. But in 1995 Schmid and Wilding [1] showed that this generator can lead to large errors in certain simulations (see figure 2 in [1]).

Task 10.2 (5 points): Random Walks

- 10.2.1 (1 points) Write a python script that simulates $m = 10000$ one-dimensional random walkers. Every random walker jumps randomly for- or backwards a single step per time step.

  Hint Save the positions of the random walkers in a integer numpy-array: `x = numpy.zeros(m, dtype=int)`. You can also apply the jumps for all random walkers at once (`numpy.random.random_integers`).
- 10.2.2 (2 points) Extend the script from the previous task such that it measures the probability of a walker to return to the starting point (*i.e.* 0) during $N = 2^i$ with $i = 0(1)14$ steps (*Rückkehrwahrscheinlichkeit, probability of return*). Generate a semilogarithmic plot (`semilogx`) for the probability of return p versus the number of steps N.

  Hint Some of the following numpy functions might be helpful:
  `all, any, logical_and, logical_or, compress, count_nonzero`
- 10.2.3 (2 points) Extend your script further so that it measures the probability of return for random walks in $d \in \{1,2,3,4\}$ dimensions. Plot the probability of return again in `semilogx` versus the number of steps. Is there any qualitative difference between random walks in different dimensions?

References

[1] F. Schmid and N. B. Wilding. Errors in Monte Carlo Simulations Using Shift Register Random Number Generators. *International Journal of Modern Physics C*, 6:781–787, 1995. `http://arxiv.org/abs/cond-mat/9512135`.