Computer simulation of physical systems I

Task IV: random number generators and 1D random walks

In this exercise you will perform a series of tests on a set of random number generators (RNGs), namely, `ran0`, `ran1`, `ran2`, and `ran3` provided in the *Numerical Recipes* book (reference in the task moodle webpage). Then you can utilize one of these RNGs to produce pseudo-random numbers (RNs) distributed according to a Gaussian normal distribution (using two different methods) and, finally, you can also generate a random walk in one dimension and compute the diffusion coefficient.

Codes and illustrative input files are contained in the archive `Task4.zip`, which can be download from the webpage of the course and unpacked with `unzip -x`. More detailed instructions for each of these steps follow.

The archive contains different scripts:

- `ran.py` contain the random number generators,
- `test_ran.py` test the random number generators probability distribution,
- `test_corr.py` test the correlations of the random number generators,
- `ran_gauss.py` use the uniform distribution random number generator to produce a normal distribution random number generator,
- `ran_walk.py` use the RNG to generate random walks.
- `plot_*.py` plot the figure from appropriate files

1. Generate sequences of RNs using different RNGs: the interactive code `test_ran.py` calls the selected RNG for the desired number of times and performs a binning of the sequence of RNs so obtained. For the moment being, specify nexp=1, e.g.

```
python test_ran.py
iran:0
number of tries:1100
number of bins:11
number of experiments:1
```

(ignore the warning about nexp). In this example, the histogram will be saved in a file named `histo-ran0_1x1100-11`.

Visualize the frequency histogram and compare two histograms built from RN sequences of different lengths (see the accompanying pyplot script `plot_histo.py`).

Since the number of trials is not very large, for each bin the frequency obtained in this numerical experiment may be much different from the ideal value for a uniform deviate (i.e., constant between 0 and 1). How to understand if this observed fluctuations are statistically acceptable?

2. Perform several statistical tests to check statistical fluctuations and correlations:

- perform a χ^2 statistic test (see Giordano&Nakanishi's book): use the code `test_ran.py` and specify a large number of experiments (at least 1000). This performs a sufficiently high number of experiments and, for each of them, computes the χ^2, which is then collect in a histogram. The integral of this histogram (i.e., the cumulative distribution of the χ^2), should be the incomplete Gamma function, P(a,x), with a equal to half the number of degrees of freedom (in our case a=(nbins-1)/2) and x=(χ^2) / 2. The frequency of χ^2 values and its integral are stored in the second and third column of `chisq-ran0_1000x1100-11`, respectively. Compare the cumulative distributions obtained from different RNGs with the theoretical one, P(5,χ^2 / 2). You can use the script `plot\_chisq.py`.

  Even if all RNG pass successfully this simple check, this does not mean that they are equally well performing. There are more complicated (and subtle) statistical test that some of them may fail (see Numerical Recipes, ch. 7).

- make a correlation test in k-space using `test_corr.py`; all the RNGs provided here should work well (i.e. they should fill the k-dimensional space uniformly), unless you use a linear congruential generator (LCG) with non-optimal parameters (use `iran=9` to select the LCG parameters (a,c,m,i0)); you can produce 2-dimensional plots from the data in the files `corr-ran?_2dim-???` (use `idim=2`)

- `test_corr.py` also prints the average of all RNs immediately following a very small RN (if there is at least one such event); compare `ran0` with the other RNGs

- (Optional) compute the distribution of the product and of the difference of pairs of RNs using `test_corr.py`; compare the data in `distprod-ran?_???` (columns 2 against 1 for the difference, 4 against 3 for the product) with the corresponding theoretical distributions

- (Optional) perform a two-dimensional χ^2 test for pairs of RNs (you can use the script `test_chisq2d.py`); `ran0` and `ran1` should fail the test for very long RN sequences (but shorter than the RNG period). As the script takes ages to run, the output can be found in the subfolder `output_test_chisq2d`.

3. Generate normally distributed RNs: with the program `ran_gauss.py` you can generate a sequence of RNs distributed according to a normal Gaussian distribution using the Box-Muller method or the Central Limit theorem. A histogram of the frequencies is computed and saved in a file named, e.g., `histo-gaussBM_10000`: visualize it and compare with the ideal normal distribution function, e.g.`1/sqrt(2*pi)*exp(-0.5*x**2)` in the gnuplot syntax.

In the CL method you can choose the number of uniform random variables that should be added to obtain a single gaussian distributed variable. Check how the distribution of the sum changes as you increase that number (it should approximate a gaussian already at a reasonably small number).

(Optional) Modify the code to compute the χ^2 test for the Gaussian distribution.

4. Perform a 1D random walk and compute the diffusion coefficient using the `ran_walk.py` code. Compare the coefficient with the theoretical value (see Giordano&Nakanishi).