Introduction to numerical projects

Here follows a brief recipe and recommendation on how to write a report for each project.

- Give a short description of the nature of the problem and the eventual numerical methods you have used.
- Describe the algorithm you have used and/or developed. Here you may find it convenient to use pseudocoding. In many cases you can describe the algorithm in the program itself.
- Include the source code of your program. Comment your program properly.
- If possible, try to find analytic solutions, or known limits in order to test your program when developing the code.
- Include your results either in figure form or in a table. Remember to label your results. All tables and figures should have relevant captions and labels on the axes.
- Try to evaluate the reliabilty and numerical stability/precision of your results. If possible, include a qualitative and/or quantitative discussion of the numerical stability, eventual loss of precision etc.
- Try to give an interpretation of you results in your answers to the problems.
- Critique: if possible include your comments and reflections about the exercise, whether you felt you learnt something, ideas for improvements and other thoughts you've made when solving the exercise. We wish to keep this course at the interactive level and your comments can help us improve it.
- Try to establish a practice where you log your work at the computerlab. You may find such a logbook very handy at later stages in your work, especially when you don't properly remember what a previous test version of your program did. Here you could also record the time spent on solving the exercise, various algorithms you may have tested or other topics which you feel worthy of mentioning.

Format for electronic delivery of report and programs

The preferred format for the report is a PDF file. You can also use DOC or postscript formats. As programming language we prefer that you choose among C/C++, Python and Fortran90/95. You could also use Java as programming language. Matlab/Maple/Mathematica/IDL are not allowed as programming languages for the handins, but you can use them to check your results where possible. The following prescription should be followed when preparing the report:

- Use Classfronter to hand in your projects, log in at blyant.uio.no and choose 'fellesrom fys3150 og fys4150'. Thereafter you will see an icon to the left with 'hand in' or 'innlevering'. Click on that icon and go to the given project. There you can load up the files within the deadline.
- Upload **only** the report file and the source code file(s) you have developed. The report file should include all of your discussions and a list of the codes you have developed. Do not include library files which are available at the course homepage, unless you have made specific changes to them.
- Comments from us on your projects, approval or not, corrections to be made etc can be found under your Classfronter domain and are only visible to you and the teachers of the course.

Finally, we do prefer that you work two and two together. Optimal working groups consist of 2-3 students. You can then hand in a common report.

Project 3, Random walks in one and two dimensions, deadline monday 22 october 12am (midnight)

For this project you can build upon program programs/chapter9/program2.cpp (or the f90 version). You will need to compute the expectation values $\langle x(N)\rangle$, $\langle y(N)\rangle$ and

$$\langle \Delta R^2(N)\rangle = \langle x^2(N)\rangle + \langle y^2(N)\rangle - \langle x(N)\rangle^2 - \langle y(N)\rangle^2$$

where N is the number of time steps. The length of every jump is unity.

a) Enumerate all random walks on a square lattice for $N = 2$ steps and obtain exact results for $\langle x(N)\rangle$, $\langle y(N)\rangle$ and $\langle \Delta R^2(N)\rangle$. Verify your results by comparing your Monte Carlo simulations with the exact results. Assume that all four directions are equally probable.

b) Do a Monte Carlo simulation to estimate $\langle \Delta R^2(N)\rangle$ for $N = 8, 16, 32$ and 64 using a reasonable number of trials for each N. Assume that we have the asymptotic behavior

$$\langle \Delta R^2(N)\rangle \sim N^{2\nu},$$

and estimate the exponent ν from a log-log plot of $\langle \Delta R^2(N)\rangle$ versus N. If $\nu \approx 1/2$, estimate the magnutide of the self-diffusion coefficient D given by

$$\langle \Delta R^2(N)\rangle \sim 2dDN,$$

with d the dimension of the system.

c) Compute now the quantities $\langle x(N)\rangle$, $\langle y(N)\rangle$, $\langle \Delta R^2(N)\rangle$ and

$$\langle R^2(N)\rangle = \langle x^2(N)\rangle + \langle y^2(N)\rangle,$$

for the same values of N as in the previous case but now with the step probabilities 0.4, 0.2, 0, 2 and 0.2 corresponding to right, left, up and down, respectively. This choice corresponds to a biased random walk with a drift to the right. What is the interpretation of $\langle x(N)\rangle$ in this case? What is the dependence of $\langle \Delta R^2(N)\rangle$ on N and does $\langle R^2(N)\rangle$ depend simply on N? This is an example of what is called a restricted random walk. The next exercise is also such an example.

d) Consider now a random walk that starts at a site that is a distance $y = h$ above a horisontal line (ground). If the probability of a step down towards the ground is bigger than the probability of a step up, we expect that the walker will eventually reach a horisontal line. This walk is a simple model of the fall of a rain drop in the presence of a random swirling breeze. Assume that the probabilities are 0.1, 0.6, 0, 15 and 0.15 corresponding to up, down, right and left, respectively. Do a Monte Carlo simulation to determine the mean time τ for the walker to reach any site on the line at $x = 0$. Find the functional dependence of τ on h. Can you define a velocity in the vertical direction? Since the walker does not always move vertically, it suffers a net displacement Δx in the horizontal direction. Compute $\langle \Delta x^2\rangle$ and find its dependence on h and τ.

e) We consider now a one-dimensional random walk, but instead of having a random walker which can move away to infinity, we assume that we have a random walker which moves along a reflecting one-dimensional lattice. That means that when the walker reaches $x = -L$ or $x = L$ it gets reflected to the previous step. If we reach $x = L$, it gets reflected to $x = L - 1$. If we reach $x = -L$, it gets reflected to $x = -L + 1$.

We start at $x = 0$ at $t = 0$ and let our walker jump with equal probability to the left or right. The step length is still one. Your task is to build up a histogram (run for many Monte carlo cycles in order to get good enough statistics, at least 10^4 Monte Carlo cycles) which represents the probability that the walker is at a site x after N steps. We call this probability $p_N(x)$. The histogram you build up contains the number of times a walker has visited a site x after N steps. Assume $L = 50$. Compare this probability with the case where you have no reflecting sites. Can you distinguish the two probability distributions if N is close to L? At what value of N can you first distinguish the two distributions?

What do these distributions look like? Comment your results

f) (**Optional**). In all the previous exercises we have assumed that the length of a jump is always unity. Now we consider a walker in one dimension with jumps of all possible lengths. The probability that the length of a single step is between x and $x\Delta x$ is $p(x)\Delta x$, where $p(x)$ is the chosen PDF.

Assume now that $p(x) = e^{-x}$, making therefore the step length to be given by $-ln(1-r)$ where r is a random number given by the uniform distribution, viz. $r \in [0, 1]$.

Modify your previous code to have a random walker which can jump with such a step length with the exponential probability distribution. You need to define a bin for which to collect data. This bin should be given by Δx as an input parameter. When you run your code for N steps a given number of Monte Carlo cycles, you count the number of times a random walker visits a site between x and $x + \Delta x$. This gives you the simulated distribution $p_N(x)$.

What is the form of $p_N(x)$? Can you recognize its form? Comment your results.