Tasks labeled by the 🎖 sign are optional and are worth extra credits.

General information

Contact: zsolt.lazar@ubbcluj.ro
Course website: MS Teams
Assignment: Submitted assignment should be archived folder with its name containing the assignment number and title of the assignment, e.g., `02_Random_walk_part_1`. The folder should be uploaded as a compressed archive containing the student's name, e.g., `Smith_John_02_Random_walk_part_1.zip`.

Requirements

Reports should include:

1. Students name
2. Title
3. Statement of the problem
4. Hypotheses
5. Strategy/method
6. Conclusions
7. Complete compilation/installation/execution instructions. Specifying dependencies are appreciated.
8. Results from demo runs including screen captures of charts, etc. - where it applies - should be included.
9. Detail the parameters used for the demonstrated runs (on the Figures)
10. Describe the environment(s) where the testing was successful (compilers, libraries, environments,...)
11. Have a list of source files attached.

Source code should be readable:

12. properly segmented, consistently indented
13. generously commented (including the description of the approach)
14. self-explanatory naming of variables

All programing tasks have to be implemented in C/C++. Some of the tasks are to be implemented ALSO in a language with vector/matrix operation capabilities like Python or Matlab. These are marked with a P/M sign.
In these cases emphasis is on employing builtin functions of the language and avoiding loops. The algorithms do not have to be fully equivalent to their C variant but rather focusing on simplicity as primary goal.
Difference between the two approaches (C vs. Python/Matlab), if any, together with benefits, drawbacks and limitations should be discussed.
Submission beyond deadline is possible but penalized.

Week 04 - The rat problem

Deadline: 05.04

1. Rats are territorial animals therefore they can get aggressive when they have to share their immediate vicinity with other individuals. We expect that by increasing the number N of rats in a room of a given enclosed area the overall level of aggressivity, $\rho_a(N) = N_a/N$, N_a is the number of nervous rats, will rise. Study this dependence through a simple model. The study should be statistical (large ensembles) and in the thermodynamic limit (large rat population)

   Rules of the dynamics:

   (a) Rats run randomly through the room of size $L \times L$. From time to time they stop and look around. Each rat can detect only those rats that are within a distance smaller than R.

   (b) If a nervous rat sees no other rat around him it becomes calm. Otherwise remains nervous.

   (c) If a calm rat sees a nervous rat around him it becomes nervous. Otherwise remains calm.

   (d) With a very small, $p \ll 1$, probability a calm rat can become nervous spontaneously.

   Prove, that in the thermodynamic limit ($L, N \to \infty$) the rat system exhibits a phase transition as a function of the rat density, $\rho \equiv N/L^2$, i.e. there is a critical rat density, ρ_c, in the system, so that for $\rho < \rho_c$ the stable dynamic equilibrium is that the nervous rats concentration is zero.

   Hint:

   - Between two vicinity checks the positions of the rats are redistributed randomly (no time correlation).
   - The system requires a few cycles (redistribute $\to$ check vicinity and change state) to reach equilibrium.
   - One can implement one of two approaches:

     (a) positions are represented as continuous (floating point) values stored in one dimensional arrays and for all pairs of rats vicinity is checked by calculating their distance and its relation to the vicinity (range) parameter R. In this case the size of the system should be $L = 1$. Alternatively, R can be set to unity and system size can be varied through L. For the Python version (+10%). Minimize the number of nested loops.

     (b) space is discretized and represented as a 2D array of integers each cell containing a value indicating an 'empty', 'calm' or 'aggressive' state. Vicinity of an array cell at x and y is the set of cells at $x \pm 1$ and $y \pm 1$ (e.g. four or eight neighbors). For the Python version of the lattice approach (+10%):
     Reprezent the matrix as a one dimensional array, identify neighbors using the `neighbors` method from the `snippets.txt` file, for randomizing rat positions make use of the `shuffle` method (see again `snippets.txt`)

   Comment on the advantages and disadvantages of the two approaches.