TP 2 : Random walk

Cours de modélisation numérique

3 March 2023

Introduction

In this exercise, you will have to implement a random walk method to simulate a simple case of a drunk man walking around. In the second exercise, you will implement a similar model for a system of N particles, then you will compare the observed behaviour with the diffusion equation solved last week.

The walk of a drunk man

To get the hang of the problem, let us first study the simple case of a drunk man walking around. Consider a rectangular grid of height H and length W, which we divide in N segments of length $\Delta x := W/N$. At time $t = 0$, we place the man in the center of the domain. The end position of the next step is determined by moving by a distance Δx in one of the four possible directions (north, south, east, west), randomly chosen. If you reach one of the boundaries of the domain, the velocity direction gets inverted in order to stay within the domain boundaries.

To do

Write your own Python code to simulate this scenario. Represent on a plot the trajectory followed by our drunk man through time. By choosing long enough simulation times, will we be able to explore each square of the domain ? What about the case in which the space is continuous (not discretised) ?

Discrete diffusion model

Last week we implemented a scheme that allowed us to solve a diffusion equation in a simple domain. In reality, the phenomenon modelled by this equation involves discrete particles that move randomly, exactly like our drunk man : the famous *Brownian* motion of particles. We would like to show that diffusion emerges as a statistical limit of a group of particles following a simple but probabilistic behaviour.

Consider a rectangular domain as previously described. At time $t = 0$, we inject a quantity of particles P from the left and bottom boundaries of the domain, so as to have a constant particle density (on average) on these boundaries. These particles are randomly distributed. Also, we keep this amount of particles constant on these boundaries through time, by constantly adding

to the number of particles. Lastly, particles that exit from the right and top boundaries of the domain are eliminated.

To do

You will find on Moodle a skeleton code modelling the above-mentioned situation.

You have to implement yourself the function `move_particles(pos)`, which takes as input parameters the array containing the coordinates (x, y) of every particle. This function updates the position of particles, following the random walk rule seen in the course. Remember, each particle must be moved by one $\delta x = \delta\vec{v} \cdot \Delta t$, where $\delta\vec{v}$ is a random vector of norm 1, and Δt is the duration of one iteration, which in our case will be equal to 1 for simplicity.

Then, create a figure showing the number of particles in the domain, at the halfway point of the domain (you could use and histogram, for example). Compare the distribution with the temperature curve at the halfway point obtained solving the Laplace equation in last week's exercise.