Dr. N. Gopan, S. Yaskovets, Prof. I. F. Sbalzarini

Summer 2023

Exercise 5

Release: 11.05.2023
Due: 16.05.2023

The following exercise will be mostly implementation of data structures that are useful for upcoming exercises. Therefore, choose now a programming language that suits you best and stick to it through the rest of the course. Write the code in a modular, re-usable manner. In case the there was not enough time to cover the material during the lecture, please familiarize yourself with the concept of Cell/Verlet Lists via reading the script or watching the video materials.

Question 1: Cell and Verlet Lists

In the lecture, you have been introduced to particle methods and their fundamental difference to "classical" numerical methods like Finite Differences and Finite Volumes. One key prerequisite to evaluate Particle-Particle (PP) interactions efficiently is obviously fast access to neighboring particles for each particle. Two data structures are presented in the script: Cell lists and Verlet lists.

a) Implement first a routine that creates cell lists from a set of particles in 1D/2D without symmetry. The function could read like this:

```
%%%%%%%%%%%%%%%%%%%%%%%%%
% Code for Exercise 5 - Creation of cell lists
%%%%%%%%%%%%%%%%%%%%%%%%%
% Input
% particlePos:  (numParticles x dim)-Matrix of particle positions
% lBounds:      Scalar lower bound on all particle positions
% uBounds:      Scalar upper bound on all particle positions
% cellSide:     Scalar value of the cell's side length
%
% Output
% particleMat:  (numParticles x dim + 1)-Matrix that contains the
% particle positions and the cell index it belongs to
% cellList:      Matlab cell structure of length number of overall
% cells (prod(numCells))
% numCells:     (dim x 1)- Vector that contains the number of cells
% per dimension

function [particleMat, cellList, numCells] =
createCellList(particlePos, lBounds, uBounds, cellSide)
```

Each cell should get a unique integer ID. Use the built-in functions *sub2ind* and *ind2sub* for this unique assignment. The output matrix *particleMat* consists of the particle positions and the ID of the cell it is contained in. The cell list should be an Octave cell data type where each element is a vector of arbitrary length.

b) Given your cell list implementation, it is now possible to tackle Verlet lists. Implement a routine that creates Verlet lists for particle distributions in 1D/2D. The function should read like this:

```
%%%%%%%%%%%%%%%%%%%%%%%%%
% Code for Exercise 5 - Creation of Verlet lists without symmetry
%%%%%%%%%%%%%%%%%%%%%%%%%
% Input
% particleMat:  (numParticles x dim + 1)-Matrix of particle positions
%                       and cell index
% cellList:             Matlab cell structure of length prod(numCells)
%                       (number of cells)
% numCells:     (dim x 1)- Vector that contains the number of cells
%                       per dimension
%
% cutoff:               Scalar distance cutoff that defines
%                       the neighborhood. It should be
%                       equivalent to the side length of a cell
%
% Output
% verletList:   Matlab cell structure of length numParticles.
%Each element contains the indices of the particles,
%i.e. the row numbers in the particleMat matrix
%

function verletList =
createVerletList(particleMat, cellList, numCells, cutoff)
```

The cutoff input argument should be equivalent to the cell side length that has been used to create the cell list. The output Verlet list should be a Matlab cell data type where each element is a vector of arbitrary length.

c) The next task is dedicated to visually verifying your implementation. Write a program that creates 10,000 particles in 2D uniformly at random in the unit box. Create a cell list and a Verlet list with cell side length (cutoff) 0.05. Create a figure and plot all particles in blue. Choose a random cell index and color all particles that are contained in this cell in green. Color the particles in adjacent cells in yellow. Pick a random particle of the randomly chosen cell and color its Verlet neighbors in red.

Question 2: Neighbor lists and Quorum Sensing

The phenomenon of Quorum (= critical density) sensing (QS) is a type of decision-making process used by many bacteria. Bacteria use QS to coordinate certain behaviors based on the local density of the bacterial population. The luminescence of the bacterium Vibrio fischeri (see Fig. 1) that lives in symbiosis with deep-sea animals (e.g. squids) is probably the most striking example of QS.

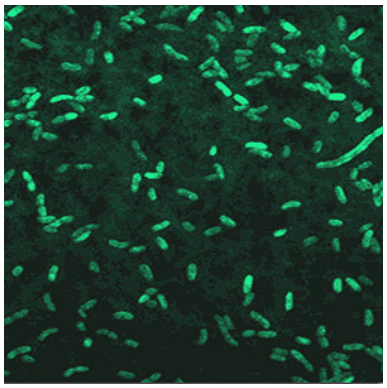

Figure 1: Microscopy image of a glowing V. fischeri colony.

V. fischeri's ability to glow is directly linked to the local density of neighbors. If a bacterium senses a critical number of neighbors around itself, it starts glowing. Using this coupling, the bacterium ensures that the host provides enough food for the colony.

Imagine now that you have given the exact 2D positions of 5,000 bacterial cells but you don't know whether a given bacterium is glowing or not. A first guess could be to find areas with high density of bacteria. In these areas, the bacteria will most likely glow. In an approximate manner, we can achieve this with cell lists. Fig. 2 schematically shows this approach. An exact approach uses Verlet lists. For each bacterium, one can efficiently determine the number of neighbors it has.

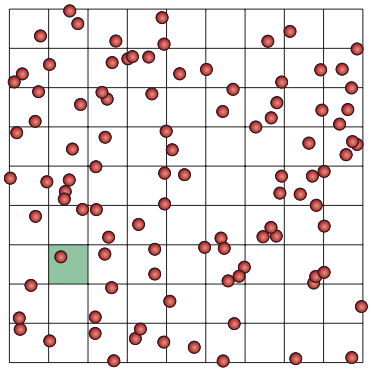

Figure 2: Schematic view of the bacterial cell population (dots). The bacterium in the shaded cell, e.g., is isolated and will probably not glow.

a) We have prepared a text file **QSBacterialPos.dat** that stores the 2D positions of 5,000 bacteria in the [0, 10] box into a matrix particlePos. Write a script (e.g. *QS_detect.m* in MATLAB) that loads the data file. Use a cell side length of 0.5 to create a cell list of

the bacterial positions. Find all cell indices that contain at least a critical density of 80 (20 bacteria / cell). Plot all bacteria in blue, and color the bacteria in the high-density cells in red.

Create a Verlet list using the constructed cell list and a cutoff of 0.5. Find all bacteria that have at least 50 neighbors within the cutoff. Create a new figure and plot again all bacteria and mark the high-density bacteria red. Compare the two plots. Report also the bacteria that have the most neighbors within the prescribed cutoff. How many neighbors do they have?

b) (Optional) Enhance the Verlet list code using intermediate cell lists and **symmetry** in 3D.