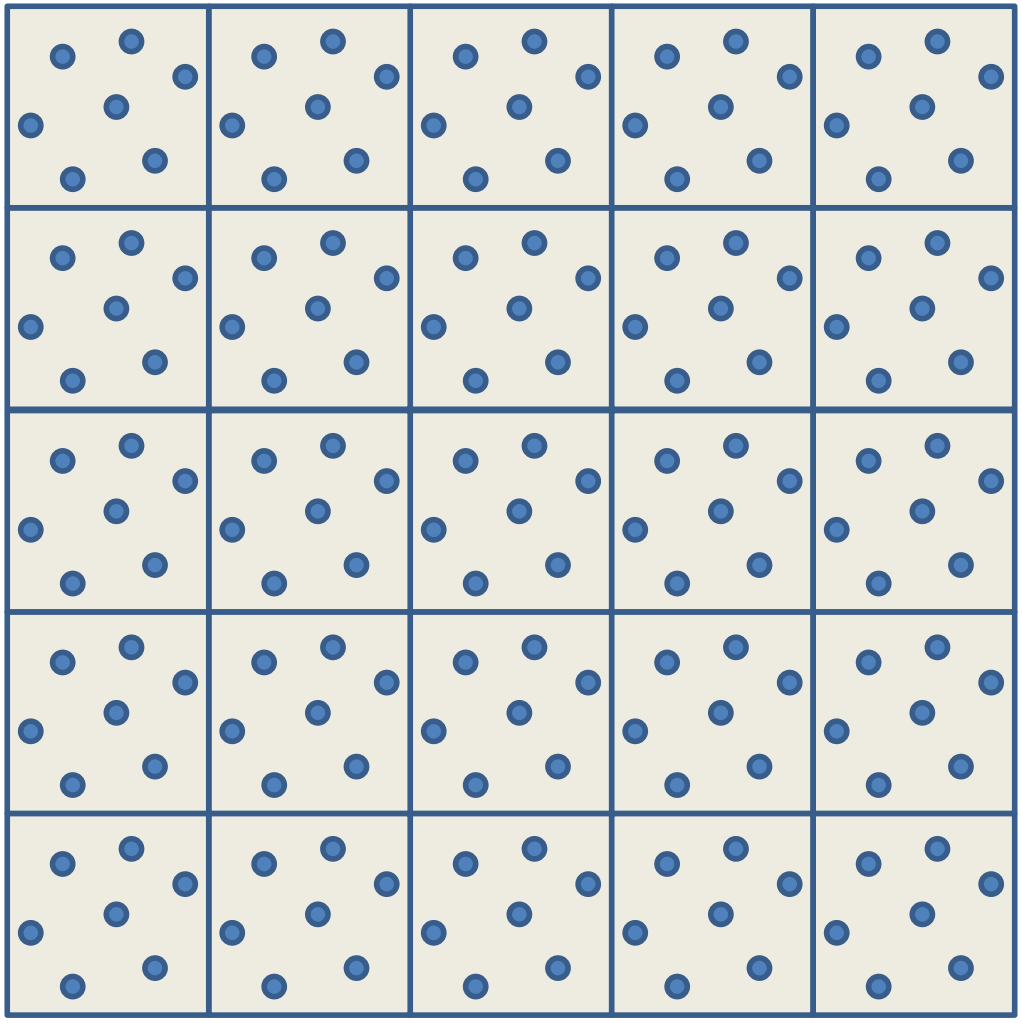
High Performance Computing: Tools and Applications

Edmond Chow
School of Computational Science and Engineering
Georgia Institute of Technology

Lecture 5

Brownian dynamics with periodic boundary conditions

Simulate an *infinite* system of particles using a periodic simulation box with a fixed number of particles



Brownian dynamics with periodic boundary conditions

- When a particle exits the box, an *image* of the particle enters the box
- We only need to keep track of one of these two particles
  - keep track of the original particle that left the box
  - since we need to measure how far it has moved
  - do not need to “wrap” that particle’s positions back into the box
- However, when computing distances to see if particles overlap, we need to consider all possible images of all particles
  - when computing the distance between a particle and another particle and all its images, convert that distance to the smallest distance
  - this is handled by:

```
dx = remainder(ri[0]-rj[0], L);
dy = remainder(ri[1]-rj[1], L);
dz = remainder(ri[2]-rj[2], L);
```

Mean squared displacement (MSD)

In 3D, the mean square displacement of particles is

$$\langle r^2 \rangle = 6Dt$$

where t is the time interval during which particles are moving, and D is the diffusion constant.

"Displacement" is used to mean the *magnitude* of the displacement; not the cumulative distance moved (which is linear in t)

Therefore the diffusion constant can be computed as follows:

- Consider the graph of the mean of the *square* of the displacement for each t (note, this is different from the square root of the average displacement)
- This graph is expected to be linear in t (except for very small t in some cases)
- The slope of this line divided by 6 is D

Simulation box width L

- How many particles? What size box?

Simulation box width L

- How many particles? What size box?
  - Suppose that particles have radius $a = 1$
  - Suppose that we want a volume fraction of particles of $\phi = 0.1$ (when particles are not overlapping) for n particles

Simulation box width L

- How many particles? What size box?
  - Suppose that particles have radius $a = 1$
  - Suppose that we want a volume fraction of particles of $\phi = 0.1$ (when particles are not overlapping) for n particles
  - Solve for L in $\frac{4}{3}\pi a^3 n = \phi L^3$
- Different volume fractions will lead to different diffusion constants D

Notes on assignment submissions

- No need to make *pull requests*
- Executables and objects are normally not checked into the repository
- Do not use tabs in your source files
  - My editor generally doesn't map tabs to the same number of spaces as your editor; therefore we don't see your source the way you intended
  - Use your editor settings to change tabs to spaces

HPC Algorithms

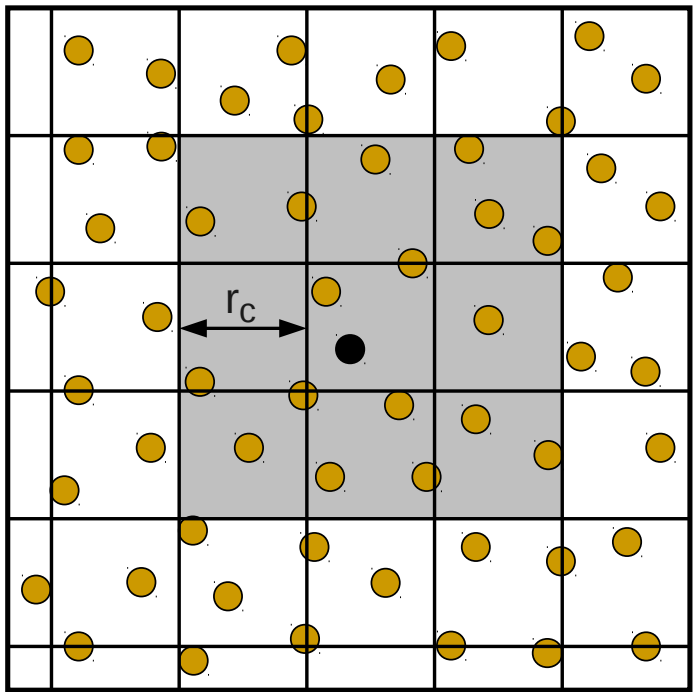
- In HPC research, use of good algorithms is just as important as good parallel implementations
- Usually, the best sequential algorithms are not the best parallel algorithms, and the best parallel algorithms can be very complicated

Computing steric repulsions

- For n particles, there are $n(n+1)/2$ possible interactions
- How to reduce the complexity to $O(n)$?

Cell lists

- Divide space into a set of cells, with cell width $\geq r_c$, where $r_c = 2$ in our case
- Particle *i* interacts only with particles in its own cell and its neighboring cells
- One sweep through all the particles is used to construct the *cell list* data structure (list of particles in each cell)
- When particles move, the cell list data structure must be reconstructed



Interesting optimization for cell lists

- Make the cells slightly larger, so that they can be reused for many time steps (or use cells that are one layer beyond the immediate neighbor cells)

Interesting optimization for cell lists

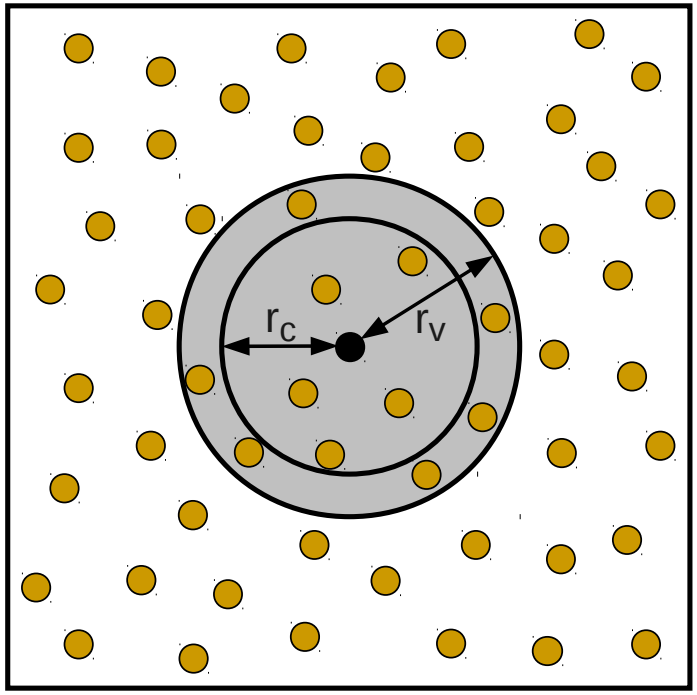
- Make the cells slightly larger, so that they can be reused for many time steps (or use cells that are one layer beyond the immediate neighbor cells)
- How can we optimize to exploit Newton's third law, i.e., don't compute equal and opposite interactions?

Interesting optimization for cell lists

- Make the cells slightly larger, so that they can be reused for many time steps (or use cells that are one layer beyond the immediate neighbor cells)
- How can we optimize to exploit Newton's third law, i.e., don't compute equal and opposite interactions?
- Traversing *all* the particles to construct the cell lists can be expensive

Neighbor list method

- Exploits the idea that particles are moving slowly
- Each particle maintains a list of neighbors within a cutoff of $r_v > r_c$
- The neighbor list can be reused for a given number of time steps, in which it could be guaranteed that an particle beyond r_v does not move closer than r_c
- How to exploit Newton's third law?



How do we construct the neighbor list?

- Naive: $O(n^2)$
- Better: use cell lists whenever the neighbor lists must be reconstructed
  - this is the strategy of many particle codes

Sequential `interactions.c` code for cell lists

Code will be in the repo `ex05` directory.

Exercise 5

- Integrate the (sequential) `interactions` function into your Brownian dynamics code; you are free to make any modifications
- Compare the performance between your Exercise 4 code and your code using the `interactions` function
- Submit your results in the `ex05` directory, including
  - `ex05.c` or `ex05.cpp` source file and `makefile`
  - `ex05.pdf` report with performance comparison
- *Due 10 pm, Wed., Sept. 7*
- We are working toward a parallel version of the interactions function for mini-project 1.