Level 7 Advanced Computational Physics – Lecture 8

Dr. Simon Hanna

October 26, 2021

11 Application Example: Molecular Dynamics

11.1 Introduction

- Molecular dynamics (MD) is technique for simulating behaviour of molecules using classical equations of motion.
- Example of an n-body problem.
- Main idea of MD: set of particles interact according to Newton's law of motion, F = ma. Given the initial positions and velocities, as well as a model of the interatomic forces, Newton's equations be integrated numerically to give a trajectory for each particle.
- Commonly, particles placed in a computational box with periodic boundary conditions.
- MD time step composed of two parts:
  1. compute forces on all particles
  2. update positions (integration).
- The computation of the forces is the expensive part.
- Number of atoms may be $\gg 10^6$.
- Typical time step may be 1 fs.

11.2 Force Computation

- Model of potential energy and forces acting between atoms called a force field.
- Typically, potential energy function is sum of bonded, van der Waals, and electrostatic (Coulomb) energy:

$$E = E_{bonded} + E_{Coul} + E_{vdW}$$

- E depends on positions of all atoms in simulation, from which $F = -\nabla E$.
- The bonded energy due to covalent bonds:

$$E_{bonded} = \sum_{bonds} k_i(r_i - r_{i,0})^2 + \sum_{angles} k_i(\theta_i - \theta_{i,0})^2 + \sum_{torsions} V_{i,n}(1 + \cos(n\omega - \gamma))$$

  which are 2, 3 and 4 body interactions respectively.
- Computationally, $E_{Coul} + E_{vdW}$ form the **bulk** of the force calculation (scales as n^2).
- Electrostatic energy from partial charges:

$$E_{Coul} = \sum_i \sum_{j>i} \frac{q_i q_j}{4\pi\epsilon_0 r_{ij}}$$

  and van der Waals energy commonly modelled by Lennard-Jones:

$$E_{vdW} = \sum_i \sum_{j>i} 4\epsilon_{ij} \left[\left(\frac{\sigma_{ij}}{r_{ij}} \right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}} \right)^{6} \right]$$

  where sum is over all atom pairs and r_{ij} is distance between i and j.
- van der Waals forces are termed short-range because they are negligible for large separations.
- Electrostatic forces are long-range, and more of a challenge.

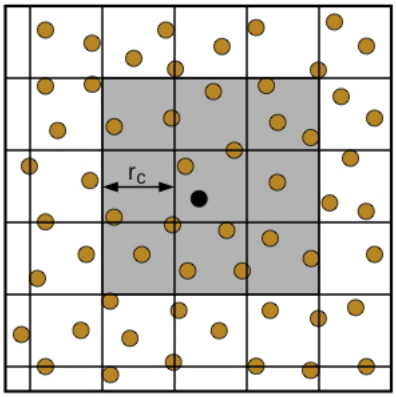
11.3 Computing Short-Range Nonbonded Forces

- Computation can be truncated beyond a cutoff radius, r_c, for each particle.
- Naive approach: examine all particles j and compute distance to particle i. Calculate force for all particles with $r_{ij} < r_c$.

- For n particles, complexity is $O(n^2)$.
- Speed up process with *cell lists* and *neighbour lists*.

11.3.1 Cell Lists

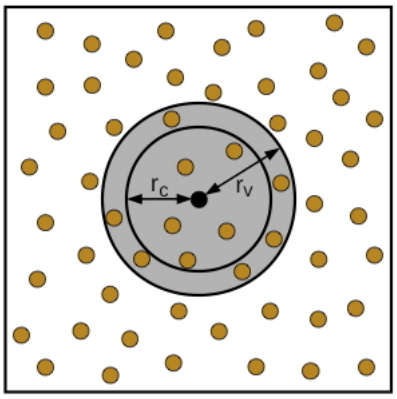
- Divide space into grid with spacing $\gtrsim r_c$



- To compute forces on i, only the particles in the cell containing i and the 8 adjacent cells (26 in 3-d) need be considered.
- Must keep list of particles for each cell – used to compute the forces.
- At each time step, because particles have moved, cell lists must be updated.
- Complexity of this approach is $O(n)$ for computing list and $O(n \times n_c)$ for force computation (n_c is average number of particles in neighbouring cells).

11.3.2 (Verlet) Neighbour Lists

- Cell lists can be efficient in condensed phases with uniform filling of cells with atoms.
- In non-dense systems, cell lists can be inefficient due to clumping of particles.
- Also inefficient because $9 \times n_c$ (or $27\times$ in 3d) can be much greater than number of particles within the r_c.
- Verlet neighbour list is list of particles, n_v, within r_c for each particle.



- Once lists constructed, computing forces is very fast, with minimal complexity $O(n \times n_v)$.
- Constructing list is expensive, $O(n^2)$.
- Advantage is that neighbour lists can be reused for many time steps provided an expanded cutoff, r_v is used to allow a buffer.

11.4 Computing Long-Range Forces

- Challenging to compute because truncation unreliable.
- Several less approximate methods available, avoiding $O(n^2)$ sum over all pairs:

Hierarchical N-body Methods including Barnes-Hut and fast multipole method. Popular for astrophysical particle simulations, but are typically too costly for accuracy required in molecular simulations.

Particle-Mesh Methods in which Poisson's equation:

$$\nabla^2\phi = -\frac{1}{\epsilon}\rho$$

is used to calculate the potential from the charge density ρ on a discrete mesh, from which the force is numerically interpolated.

Ewald Method for periodic systems, where the short-range part of the force is computed using particle-particle methods, and the long-range part found using Fourier transforms.

Particle-Mesh Ewald is a combination of the last two methods.

11.5 Parallel Decompositions

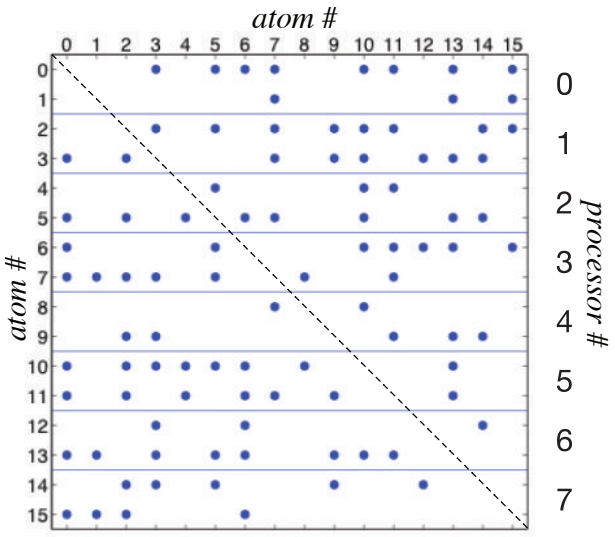
- There are several ways of performing the parallel computation of forces: decomposition by atom, by

force contribution, by region or some combination of the above.

- In each case, there is a possible saving of factor of 2 because $F_{ij} = -F_{ji}$.

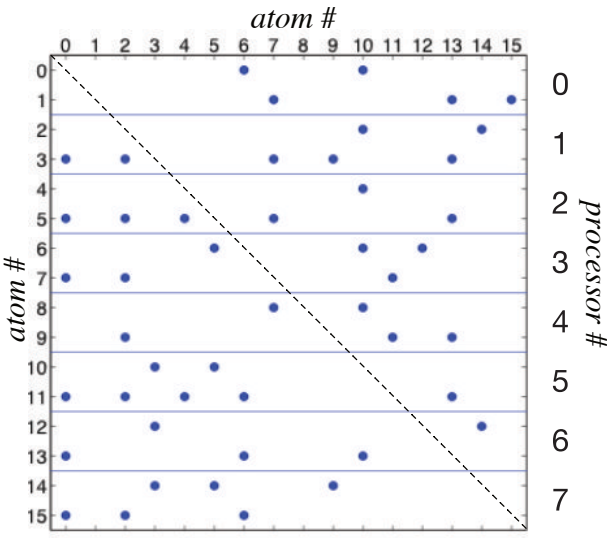
11.5.1 Atom Decompositions

- Each particle assigned to one processor, which computes its forces and updates its position for the entire simulation.
- Consider 16 particles partitioned between 8 processors:



Spots mark the forces needed – sparse due to cut-offs.

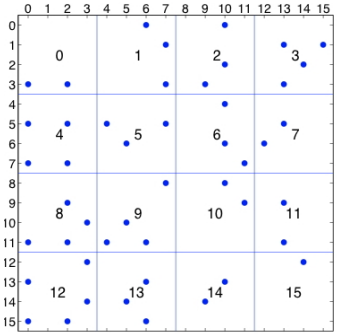
- Each processor assigned approximately same number of particles.
- Generally, each processor needs to communicate with **all** other processors to share updated particle positions.
- Possible algorithm:
  1. At beginning of time step, each processor holds the positions of particles assigned to it.
  2. send/receive particle positions to/from all other processors
  3. if cut-offs used, determine which non-bonded forces to compute
  4. compute forces for particles assigned to this processor
  5. update positions (integration) for particles assigned to this processor
- Save time because $F_{ij} = -F_{ji}$, but must communicate values to other processors
  - Choosing upper or lower triangular part of the force matrix is **bad** because the computational load unbalanced.
  - Better to compute F_{ij} if $i + j$ even in upper triangle, or if $i + j$ odd in the lower triangle.

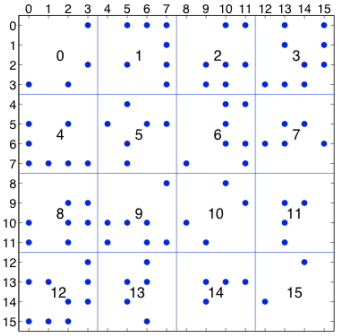


  - Other patterns possible.
- All forces on a particle owned by a processor no longer computed by that processor. Therefore must exchange forces with other processors.
- Revised algorithm:
  1. At beginning of time step, each processor holds the positions of particles assigned to it.
  2. send/receive particle positions to/from all other processors (could do this via a master node)
  3. if cut-offs used, determine which non-bonded forces need to be computed
  4. compute partial forces for particles assigned to this processor
  5. send particle forces needed by other processors and receive particle forces needed by this processor
  6. update positions (integration) for particles assigned to this processor
- This algorithm advantageous if extra communication outweighed by savings in computation (communication doubles in general).

11.5.2 Force Decompositions

- Forces distributed among processors for computation.
- Blocks allow calculation of all forces between groups of particles e.g. below for 16 particles and 16 processors.
- Particles also assigned to processors (as in atom decompositions) for position update. In figure, *assume each processor stores position of one particle*.

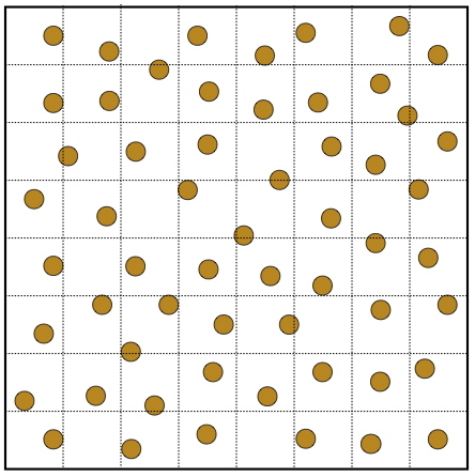




- Consider communication required in one time step for force decomposition e.g. processor 3:
  - Computes partial forces for particles 0 – 3;
  - Needs positions from particles 0 – 3 and 12 – 15 (communicates with processors 0 – 3 and 12 – 15).
  - After all forces computed, collects partial-forces on particle 3 from processors (communicates again with processors 0 – 3).
- Typical algorithm for one time-step might be:
  1. At beginning of time-step, each processor holds positions of particles assigned to it.
  2. send positions of my assigned particles which are needed by other processors;
  3. receive row particle positions needed by my processor (communication between processors in same row)
  4. receive column particle positions needed by my processor (communication generally with processors in another row)
  5. if cutoffs used, determine which non-bonded forces need computing
  6. compute forces for my assigned particles
  7. send forces needed by other processors;
  8. receive forces needed for my assigned particles (communication between processors in same row only)
  9. update positions (integration) for my assigned particles
- For p processors arranged in square, force matrix is partitioned into $\sqrt{p} \times \sqrt{p}$ blocks.
- Communication involves three steps, with $\sqrt{p}$ processors in each.
- Much more efficient than atom decomposition which requires communications among all p processors.
- Exploiting $F_{ij} = -F_{ji}$ results in extra step of communication (4 steps versus 3).

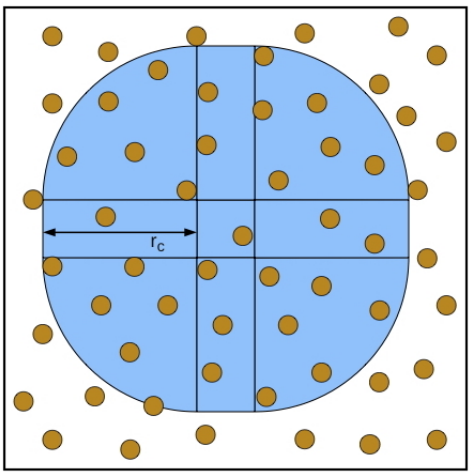
11.5.3 Spatial Decompositions

- Simulation volume decomposed into cells.
- Each cell assigned to a processor which computes forces on particles within the cell:



- Usually, number of cells = number of processors.

- Particles move during simulation, so assignment of particles to cells changes as well.
- *Import region* (shaded) contains particles potentially within the cutoff radius, r_c, of particles in given cell:



- Given cell must import positions of particles in this region to perform its force calculation.
- Not all particles in the given cell will interact with all particles in import region (e.g. if import region large compared to r_c).
- Typical algorithm (one time-step):
  1. At the beginning of the time step, each processor holds the positions of the particles in its cell.
  2. send positions needed by other processors for particles in their import regions;
  3. receive positions for particles in my import region
  4. compute forces for my assigned particles
  5. update positions (integration) for my assigned particles
- Import region is similar to Verlet neighbour list.
- To exploit $F_{ij} = -F_{ji}$, shape of import region can be halved, but extra communication step involved.
- Main advantage of spatial decomposition is it only requires communication between processors corresponding to nearby particles.
- Disadvantage for very large numbers of processors, if import region large compared to the number of particles inside each cell. (Avoid by increasing problem size, or decreasing number of processors).

11.6 Common Integration Schemes

Schemes optimising energy conservation generally favoured for MD.

11.6.1 Verlet method

Consider the second order equation for a single particle:

$$x'' = f(t, x)$$

where $f(t, x) = -\nabla E/m$. Replace LHS with finite difference approximation:

$$\frac{x_{k+1} - 2x_k + x_{k-1}}{h^2} = f(t_k, x_k)$$

Rearrange:

$$x_{k+1} = 2x_k - x_{k-1} + h^2 f(t_k, x_k)$$

(or derive from Taylor series).

- Need alternative method to supply initial step.
- Time-reversible.
- Can be inaccurate due to addition of small h^2 term.

11.6.2 Leap-frog method

- Is algebraically equivalent to the Verlet method, and similar to the mid-point method (modified Euler) used in 2nd year computing:

$$x_{k+1} = x_k + h v_{k+1/2}$$

$$v_{k+1/2} = v_{k-1/2} + h f(t_k, x_k)$$

  where v is velocity, offset from x by half a time-step.
- Formula does not suffer from the same roundoff problems as standard Verlet;
- Velocities available (need to be re-centered with displacements to calculate total energy).

11.6.3 Velocity Verlet

- Velocities computed at same points as displacements.

$$x_{k+1} = x_k + hv_k + \frac{h^2}{2} f(t_k, x_k)$$

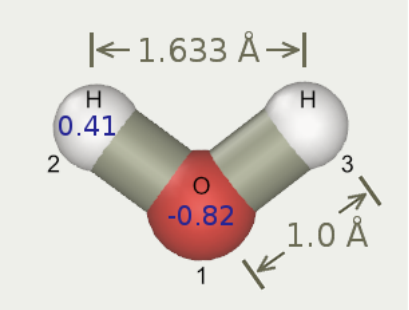
$$v_{k+1} = v_k + \frac{h}{2} \left[f(t_k, x_k) + f(t_{k+1}, x_{k+1}) \right]$$

11.6.4 Comments

- Each method can be implemented so only two sets of quantities are stored (two previous positions, or position and velocity).
- Methods popular due to simplicity: only one costly force evaluation per step.
- Higher-order methods generally not practical (additional compute and communication cost outweighs any accuracy advantage – better to use smaller time-step with simple method).
- Velocity Verlet also basis of multiple time-step algorithms where slowly-varying (typically long-range) forces are evaluated less frequently than quickly-varying (typically short-range) forces.
- Adaptations available to control simulation temperature and pressure.
- Generally speaking, for energy conserving systems, use Verlet or leap-frog type methods. Avoid using these for damped systems.
- For non-energy conserving systems, consider the Runga-Kutta methods, which have smaller errors for given integration time.

11.7 Practical example - Gromacs

- GROningen MAchine for Chemical Simulations (GROMACS) is a general purpose molecular dynamics package. It is:
  - Free to download
  - Available on BlueCrystal
  - Flexible to use
  - Very fast
- Large scale enterprise:
  - 1.66 million lines of code
  - 478 person years of development
  - Estimated cost $26.2M
- Very high performance:
  - Written in C using **instruction-level** parallelism.
  - Excellent CUDA-based GPU acceleration
  - Can be run in parallel, using MPI.
  - Also uses OpenMP where available.
  - Uses neutral territory methods for domain decomposition.
- For example, to model water, GROMACS needs to know atom types:




```
[ atomtypes ]
; name      at.num    mass     charge ptype  sigma       epsilon
OW_spc       8      15.9994 -0.8476   A    3.16557e-01  6.50629e-01
HW_spc       1       1.0080  0.4238   A    0.00000e+00  0.00000e+00
```

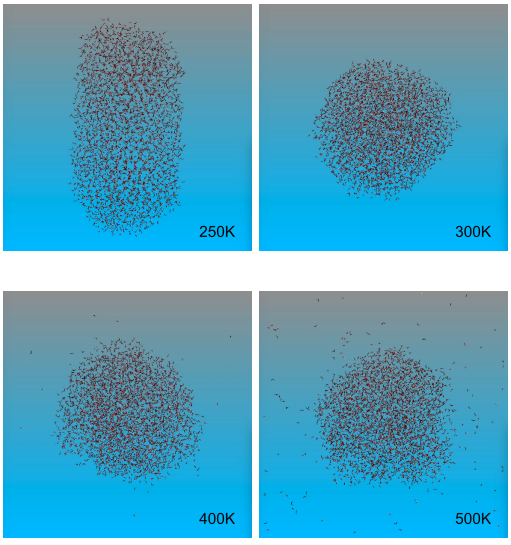
- and topology:

```
[ moleculetype ]
; molname nrexcl
water 2

[ atoms ]
;nr    type     resnr    residue      atom     cgnr     charge       mass
  1   OW_spc      1       water        OW       1       -0.8476   15.99940
  2   HW_spc      1       water        HW1      1        0.4238    1.00800
  3   HW_spc      1       water        HW2      1        0.4238    1.00800

[ bonds ]
;ai    aj     funct     c0        c1
1       2       1       0.1     345000
1       3       1       0.1     345000

[ angles ]
;ai    aj      ak     funct       c0     c1
2       1       3       1       109.47  383
```

- Example output:

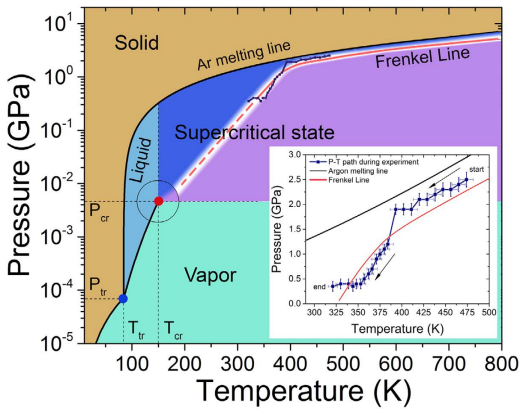
```
GROMACS version:    2016.3
Precision:          single
Memory model:       64 bit
MPI library:        thread_mpi
OpenMP support:     enabled (GMX_OPENMP_MAX_THREADS = 32)
GPU support:        CUDA
SIMD instructions:  AVX2_256
FFT library:        fftw-3.3.5

Running on 1 node with total 4 cores, 8 logical cores, 1 compatible GPU
Hardware detected:
  CPU info:
    Vendor: Intel
    Brand:  Intel(R) Core(TM) i7-6700K CPU @ 4.00GHz
    Family: 6   Model: 94   Stepping: 3
    SIMD instructions most likely to fit this hardware: AVX2_256
```

```
GPU info:
  Number of GPUs detected: 1
  #0: NVIDIA GeForce GTX 980, compute cap.: 5.2, ECC:  no, stat: compatible

    R E A L   C Y C L E   A N D   T I M E   A C C O U N T I N G

On 1 MPI rank, each using 8 OpenMP threads

 Computing:          Num   Num      Call    Wall time         Giga-Cycles
                     Ranks Threads  Count      (s)         total sum    %
-----------------------------------------------------------------------------
 Neighbor search        1    8      60001      64.049       2053.679   1.5
 Launch GPU ops.        1    8    6000001     121.844       3906.797   2.9
 Force                  1    8    6000001     256.602       8227.690   6.2
 PME mesh               1    8    6000001    3005.303      96361.943  72.3
 Wait GPU local         1    8    6000001     411.912      13207.544   9.9
 NB X/F buffer ops.     1    8   11940001     118.395       3796.201   2.8
 Write traj.            1    8       6003      14.567        467.073   0.4
 Update                 1    8    6000001     145.755       4673.474   3.5
 Rest                                          17.903        574.054   0.4
-----------------------------------------------------------------------------
 Total                                       4156.331     133268.455 100.0
-----------------------------------------------------------------------------
 Breakdown of PME mesh computation
-----------------------------------------------------------------------------
 PME spread/gather      1    8   12000002    1097.041      35175.490  26.4
 PME 3D-FFT             1    8   12000002    1768.003      56689.197  42.5
 PME solve Elec         1    8    6000001     133.854       4291.882   3.2
-----------------------------------------------------------------------------

 GPU timings
-----------------------------------------------------------------------------
 Computing:                         Count  Wall t (s)      ms/step       %
-----------------------------------------------------------------------------
 Pair list H2D                      60001       4.135        0.069     0.1
 X / q H2D                        6000001     104.749        0.017     2.9
 Nonbonded F kernel               5940000    3319.543        0.559    92.7
 Nonbonded F+ene+prune k.           60001      58.148        0.969     1.6
 F D2H                            6000001      94.029        0.016     2.6
-----------------------------------------------------------------------------
 Total                                       3580.604        0.597   100.0
-----------------------------------------------------------------------------

Average per-step force GPU/CPU evaluation time ratio: 0.597 ms/0.544 ms = 1.098
For optimal performance this ratio should be close to 1!

               Core t (s)   Wall t (s)        (%)
       Time:    33250.650     4156.331      800.0
                         1h09:16
                 (ns/day)    (hour/ns)
Performance:       62.363        0.385
Finished mdrun on rank 0 Sun Oct 22 20:03:55 2017
```



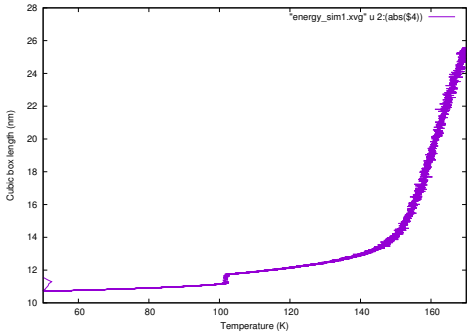
11.8 Simulation of Argon

- Only one energy term needed: Lennard-Jones potential with parameters $\sigma = 0.34\,\text{nm}$ and $\varepsilon = 0.99774\,\text{kJ/mol}$.
- FCC crystal structure (4 atoms per unit cell) with lattice parameter $a \simeq 0.582\,nm$ at 100 K.
- Phase diagram well known. e.g.:



from Bolmatov et al. Nature Scientific Reports (2015).

- Simulating at 10 atm and $50 \leq T \leq 170\,\text{K}$ shows two transitions (s-l & l-g), as seen in the simulation box size versus temperature:



Exercise Week 5

By now you should have chosen your mini-project and should be writing parallel programs.

If you have any concerns about your choice of mini-project please make contact with Dr Hanna as soon as possible.

Please make use of the weekly drop-in sessions for support from our friendly demonstrators Cale and Mike.