Tools of high performance computing 2024

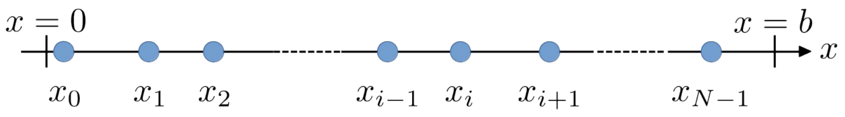
Exercise 8

Return by Monday 18.3.2024
Exercise session: Tuesday 19.3.2024

Problem 1. *(24 points)*

Download a simple dynamical simulation[1] code for a 1D anharmonic system from the course Moodle page[2]. It integrates the equations of motion of an ensemble of particles interacting with an anharmonic potential. The system has N [variable name in the code: `nat`] particles and its size is b [`box`] which in the code is taken to be N. Initial positions of the particles are

$$x_i = i - 1.$$



Periodic boundary conditions are applied. This means that if a particle is moved outside the system it is brought from the other end (wrap-around). Likewise, when calculating the potential energy atom 0 feels atom $N-1$ and vice versa. The potential is described by harmonic (k_1) and anharmonic (k_2) terms. The potential energy U_i [`ep(i)`] of particle i is written as

$$U_i = \frac{1}{2}[k_1(x_i - x_{i-1} - d)^2 + k_1(x_{i+1} - x_i - d)^2 + k_2(x_i - x_{i-1} - d)^3 + k_2(x_{i+1} - x_i - d)^3]$$

Here d [`d`] is the 'equilibrium bond length'. Masses of all particles are set to one, meaning that the force is equal to acceleration.

Equations of motion of the system are integrated by the so called leap-frog algorithm using time step δt [`dt`]:

$$v_i\left(t + \frac{\delta t}{2}\right) \leftarrow v_i\left(t - \frac{\delta t}{2}\right) + a_i\delta t \quad (a_i \text{ is the acceleration of atom } i)$$

$$x_i\left(t + \delta t\right) \leftarrow x_i(t) + v_i\left(t + \frac{\delta t}{2}\right)\delta t$$

$$v_i(t) = \frac{1}{2}\left[v_i\left(t - \frac{\delta t}{2}\right) + v_i\left(t + \frac{\delta t}{2}\right)\right].$$

The code attached is a complete serial program. Compile and try to run it:

```
gfortran -O2 -o md1d md1d.f90
./md1d
```

1 For basics of molecular dynamics simulations, see the lecture notes of course FYS2088 Introduction to molecular dynamics simulations: intro2md_2023.pdf. Note that this is not necessary for solving the exercise.

2 File `md1d.f90` or `md1d.cpp`

```
usage: …
```

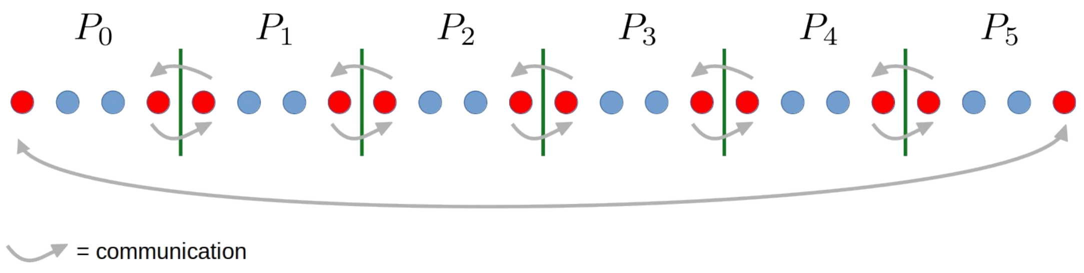
Suitable command line for testing might be

```
./md1d 100 0.01 10000 1 1
```

Program prints the time (`dt*n`, column 1) and total (`epotsum+ekinsum`, column 2), potential (`epotsum`, column 3) and kinetic (`ekinsum`, column 4) energies to `stdout` at simulation step intervals of `eout`.

Your task is to parallelize the program using MPI. Each processor takes care of particles in a certain part of the system (spatial decomposition, or more precisely, bonding graph decomposition). Due to the fixed bonding topology all particles always interact only with the same two nearest neighbors they have in the beginning. This means that you need not redistribute the particles among the processes during the simulation.

You may assume that the number of particles is exactly divisible by the number of processors.



There are three phases where communication is needed:

1. Input parameters are distributed to all processors [*only in the beginning of the simulation*]. However, you can do this so that all processes read in the command line arguments if you wish, meaning no need to any communication.
2. In order to calculate potential energies and forces coordinates of edge particles are communicated to neighboring processors (see the figure above) [*every time step*]. Here `MPI_Sendrecv` is handy (possibly with `MPI_Cart_shift`). See the attached Fortran example[1].
3. Kinetic and potential energies of all atoms are summed and printed [*every* `eout` *time step*]. This means that you locally calculate the sum of these quantities and then call `MPI_Reduce` with `MPI_SUM`.

Send the parallel code, explain its inner workings and give examples of output when run on many processors. How does the wall clock time of the program execution behave as a function of the number of processors (1..8)? Describe the platform on which you run the code (mainly whether it is shared or distributed memory).

1 `mpi_topo1_sendrecv.f90`