P. Koumoutsakos, P. Hadjidoukas
ETH Zentrum, HG D1.1/G3
CH-8092 Zürich

Spring semester 2018

Set 10 - CUDA and N-body problem (continuation)

Issued: 7 May 2018
Hand in: 14 May 2018

Question 1: N-body

In this exercise you will have to extend the N-body solver from the previous exercise in order to verify the correctness of the code and to extract some quantities of interest (QoI).

Since the particle system is closed and potential, the total energy (sum of potential energy and kinetic energy) should be conserved. In order to check that, you will have to calculate the two components of the total energy:

$$\begin{aligned} E &= E^K + E^P \\ E^K &= \sum_{i=1}^{N} \frac{m v_i^2}{2} \\ E^P &= \sum_{i=1}^{N} \sum_{j=1}^{i-1} U^{LJ}(|\boldsymbol{r}_i - \boldsymbol{r}_j|) \end{aligned} \tag{1}$$

Note that summation for the potential only considers every pair of the particles once.

a) You have to write two functions that compute the two energies. Use the reduction approaches discussed during the lecture. The potential energy kernel should be independent of the exact potential expression (just like the forces kernels), so it's a good idea to extend the Lennard-Jones functor with the function `float energy(float3 p1, float3 p2)` and use that function in the kernel.

b) Compute the two components of the system energy every 100 or 1000 timesteps and verify that the total energy drifts by at most 1-2% when the simulation time reaches 1.0 (use the first test case of Exercise 9). You may need to decrease the time-step to see the energy conservation.

A lot of molecular dynamics simulations are performed in a so called canonical ensemble, so that the temperature of the system stays constant during the simulation. Previously your simulation was in a microcanonical ensemble, i.e. at constant total energy. In the limit of very many particles in the system the difference between the two ensembles vanishes, but with relatively small systems care must explicitly be taken to enforce constant temperature of the system.

Assuming Boltzmann constant k_B is unity, system temperature is related to the kinetic energy this way:

$$T = \frac{2}{3N} E^K \tag{2}$$

In order to maintain the desired temperature, you will have to rescale the velocities of all the particles by a certain value from time to time.

c) Calculate the rescaling factor assuming given current temperature T_{cur} and target temperature T_0.

d) Implement temperature control in the code every 10-50 timesteps. Rescale the velocities on the GPU directly.

It is useful to observe the configuration of the particles from time to time. For that you will have to dump the positions of the particles in the system in an XYZ file format.

e) Write a function that takes a *host* pointer to particle coordinates and dumps them into an XYZ file with the following format:

```
<number of particles N>
<comment line>
0 <x1> <y1> <z1>
0 <x2> <y2> <z2>
...
0 <xN> <yN> <zN>
```

Those files may be opened with VMD or VisIt software to produce a visual support to your simulation.

f) Call this function every 100 or 1000 time-steps. When downloading the data, overlap the GPU to CPU transfer with computations by using two different non-blocking streams. Take care about synchronization and data consistency, i.e. the transfer **cannot** be overlapped with integration as it changes particle positions.

Finally you have to prepare your code to plug it into Pi4U UQ framework (next exercise).

g) Pass the LJ parameters ε and σ via the command line

The quantity of interest for the UQ will be the system pressure at given density and temperature. Instantaneous pressure of the system is expressed in terms of particle pairwise forces with the help of virial:

$$P = \frac{k_B T N}{V} + \frac{1}{3V} \sum_{1 \leqslant i < j \leqslant N} \boldsymbol{F}_{ij} \cdot \boldsymbol{r}_{ij} \tag{3}$$

Here $V = L^3$ is the total volume of the system.

h) Write a function that computes the system pressure.