PHYS 511: Computational Modeling and Simulation - Fall 2017
Assignment #4, due Monday November 20, by 5:00 pm

Classical dynamics of N gravitating bodies; basics of parallel programming with OpenMP

In this assignment we will be simulating the dynamics of N gravitating bodies (essentially particles) using velocity Verlet algorithm. Dynamics of a system of interacting classical particles is a very common problem – molecular dynamics and motion of stars in galaxies are typical examples of this problem. Needless to say that when the number of particles N is large (in fact already when $N > 2$) numerical simulations are the only practical way to approach it. It requires integrating Newton's equations of motion that are represented by the following second-order differential equations

$$m_k \frac{d^2\mathbf{r}_k}{dt^2} = \mathbf{F}_k, \quad \text{for} \quad k = 1, \ldots, N \tag{1}$$

or, equivalently,

$$\frac{d\mathbf{r}_k}{dt} = \mathbf{v}_k, \qquad m_k \frac{d\mathbf{p}_k}{dt} = \mathbf{F}_k, \qquad k = 1, \ldots, N \tag{2}$$

where m_k is the mass of the k-th particle, $\mathbf{r}_k = (x_k, y_k, z_k)$ is its position, $\mathbf{v}_k = (v_{kx}, v_y, v_{kz})$ is its velocity vector, and $\mathbf{F}_k = (F_{kx}, F_{ky}, F_{kz})$ is the force acting on it. The force is determined through the gradient $\nabla_k = (\frac{\partial}{\partial x_k}, \frac{\partial}{\partial y_k}, \frac{\partial}{\partial z_k})$ of the potential energy of interaction U,

$$\mathbf{F}_k = -\nabla_k U. \tag{3}$$

In the absense of external forces it is usually given as a sum of pairwise (and usually central) interactions between the particles:

$$U = \sum_{i<j} U_{ij}, \qquad U_{ij} = U_{ij}(|\mathbf{r}_i - \mathbf{r}_j|). \tag{4}$$

The integration of the above equations is performed over a certain period of time starting from some initial conditions:

$$\mathbf{r}_k(0) = \mathbf{r}_k^{(0)}, \qquad \mathbf{v}_k(0) = \mathbf{v}_k^{(0)}, \tag{5}$$

While many different approaches exist to integrate these equations of motion, one of the simplest and most popular ones are the Verlet algorithm. In particular, a single time step of the velocity Verlet algorithm (there is also so called position Verlet) can be stated as follows:

$$\mathbf{v}_k^{(q+\frac{1}{2})} = \mathbf{v}_k^{(q)} + \frac{1}{2m_k}\mathbf{F}_k^{(q)}\Delta t, \tag{6}$$

$$\mathbf{r}_k^{(q+1)} = \mathbf{r}_k^{(q)} + \mathbf{v}_k^{(q+\frac{1}{2})}\Delta t, \tag{7}$$

$$\mathbf{v}_k^{(q+1)} = \mathbf{v}_k^{(q+\frac{1}{2})} + \frac{1}{2m_k}\mathbf{F}_k^{(q+1)}\Delta t. \tag{8}$$

Here Δt is the time step of the integration.

1. Write a Fortran program that simulates the motion of N (this number is unknown in advance) gravitating objects. The pair interaction potential of two gravitating objects (in some appropriate units where the gravitational constant is unity) is $U_{ij} = -\frac{m_i m_j}{|\mathbf{r}_i - \mathbf{r}_j|}$.

2. The parameters of the simulation are the following: $t^{(0)} = 0$, $t^{(\text{fin})} = 5$, $\Delta t = 0.001$. The masses and initial positions and velocities of the N objects are provided in the input file (`init.dat`).

3. At each integration step print one line on the screen that shows the current iteration number, current time t, current total energy E, and the relative energy change $\Delta E/E$ (with respect to the previous iteration). These values should serve as good indicators of the quality of the integration scheme used. The total mechanical energy of the system is defined as $E = \frac{1}{2}\sum_k m_k \mathbf{v}_k^2 + \sum_{k<l} U_{kl}(|\mathbf{r}_k - \mathbf{r}_l|)$.

4. At the final step output the state of the system in a file (`final.dat`) that has the same format as `init.dat`.

5. Use OpenMP clauses to facilitate parallelism in your code.

6. When you know your code works properly and no longer needs any debugging, recompile it with a compiler flag that corresponds to the highest optimization level, `-O3`, e.g.
   `gfortran -O3 -fopenmp myprog.f90 -o myprog`
   This will ensure that it runs with the highest possible speed.

7. Make test runs of your code with different numbers of CPU cores involved, measure the execution time in each case. The number of cores can be controlled through the value of environmental variable `OMP_NUM_THREADS`, e.g. in order to use two cores just type `export OMP_NUM_THREADS=2` in the terminal before executing the program. Do you observe any speed up when you increase the number of cores? Is it something that you expected? Why yes or why not? The discussion and interpretation of what you observe is an important part of the assignment.

8. Submit your code (`as4.f90`) along with the final output (`final.dat`) and any relevant discussion and comments (`report.txt`) by uploading them to directory `as4` on your google drive that is shared with the instructor.

9. *Bonus task for the geeks.* Can you visualize the dynamics in a video (e.g. `avi`, `mp4`, `mov`, animated `png`)? For that you will need your program to output the coordinates of all particles of the system every few iterations (as a separate file) and then write a shell or Python script that generates plots (e.g. `png` files where each particle is represented as a dot or a tiny ball in 3D) for each of these separate snapshots of the system. Lastly, you can use some programs to generate a movie from the set of sequential png files.