Introduction

This readme details a small MD code, and the implementation of some different integrators, used to solve equations of motions. Some comparisons between the integrators are made with respect to their numerical stability and the energy conservation of the integrators. This is a good starting point for further exploration of molecular dynamics, however, the code is now somewhat outdated, as the program was originally written in 2007.

The MD code

This code is written in C++, and the particles are implemented in the Atom class. This contains position, velocity, and force currently acting on a particle, as well as a pointer in order to be able to link the Atoms together in a list. This enables us to keep track of an (in principle) arbitrary number of particles, as such a list can be dynamically allocated at run-time. It should be noted that for large systems, this can be a performance bottleneck. The list class is called System, and contains methods for extracting various global informations from the simulation, such as its thermodynamic properties, linear momentum, and energies. Methods to output the result to some different common MD file formats are also included here, as are propagator methods and system scaling methods (linear momentum and velocity). A purely repulsive Weeks-Chandler-Andersen (WCA) potential [1] was chosen to model the interaction between the particles

$$\Phi_{\text{repulsive}}(r) = \begin{cases} \Phi_{\text{LJ}}(r) & \text{if } r < 2^{1/6}\sigma \\ 0 & \text{if } r \geq 2^{1/6}\sigma \end{cases} \tag{1}$$

where $\Phi_{\text{LJ}}(r)$ is the Lennard-Jones potential [2]. Three different integrators were implemented, namely Euler predictor-corrector [3], partitioned Runge-Kutta [4], and velocity Verlet [5]. In order to implement e.g. the Gear predictor-corrector method, higher order derivatives would have to be included.

In the main code, one may specify the timestep for each of the different integrators as desired. Filenames to which output will be written can be changed here, as well as the run-time of the simulation. Velocity rescaling and removal of linear momentum is run every 10th timestep during the first 200 timesteps by default, but can be changed if desired. A routine exists in the System class to output the positions of the atoms to Tinker .arc format, by default a frame is written to the .arc file every 10th timestep. A file called vitals(Integrator).dat is output, in which the kinetic, potential and total energy is written every timestep. The result(Integrator).dat contains positions, velocities and forces for all timesteps. the file single(Integrator).dat traces a single particle through the simulation, a frame is dropped every 10th timestep by default. The code also has energy divergence checking, and automatically terminates any simulation in which the condition

$$E_{n+1} > E_n + 100 \tag{2}$$

is satisfied. This may be a somewhat crude condition, but keeps the run-time of failed simulations to a minimum.

Comparing the integrators

On a short time-scale, the difference between the three methods is not very noticeable. The energy diverges slowly in both the Euler and Runge-Kutta method at $\Delta t = 0.001$ s, while staying constant in the Velocity Verlet, which is what one would expect. If the timestep is made smaller (0.0001 s or so) to conserve energy better in the first two methods, the particle path in xy-space improves a lot over Velocity Verlet run at 0.001s step. Thus, even though Velocity Verlet seems to conserve energy well, it is not by default the best choice for all applications. On a long timescale (10 s) the Velocity Verlet has a very stable total energy, while the Euler and Runge-Kutta diverges as before. The divergence is approximately twice as fast for the Euler integrator when compared to the Runge-Kutta one. To produce a energy stable run over 10s, a timestep of 0.001 s can be used for Velocity Verlet, while for the Runge-Kutta a timestep of 0.0009 s is needed, and the Euler needs a timestep of about 0.0005 s to be stable enough in energy. Plots showing the energy behaviour of the different integrators can be found in `plot1.pdf` ($\Delta t = 0.0005$ s), `plot2.pdf` ($\Delta t = 0.0009$ s), and `plot3.pdf` ($\Delta t = 0.001$ s). The xy trajectories of one simulation can be found as an animation in `simulation.gif`.

Bibliography

[1] Weeks, J. D., Chandler, D. & Andersen, H. C. Role of repulsive forces in determining the equilibrium structure of simple liquids. *The Journal of Chemical Physics* **54**, 5237–5247 (1971). URL `https://doi.org/10.1063/1.1674820`.

[2] Lennard-Jones, J. E. On the determination of molecular fields. —ii. from the equation of state of a gas. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* **106**, 463–477 (1924). URL `http://rspa.royalsocietypublishing.org/content/106/738/463`.

[3] Butcher, J. C. *Numerical Methods for Ordinary Differential Equations* (Wiley, 2003). ISBN: 978-0-471-96758-3.

[4] Koto, T. Partitioned runge-kutta methods for partial differential equations (the numerical solution of differential equations and linear computation). *RIMS* **1320**, 60 (2003). URL `http://hdl.handle.net/2433/43079`.

[5] Swope, W. C., Andersen, H. C., Berens, P. H. & Wilson, K. R. A computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters. *The Journal of Chemical Physics* **76**, 637–649 (1982). URL `https://doi.org/10.1063/1.442716`.