MA5233 Computational Mathematics

Lecture 11: Molecular Dynamics

Simon Etter

Semester I, AY 2020/2021

Molecular Dynamics

Disclaimer

The content of this lecture is not examinable. Its only purpose is to illustrate some of the material presented in Lecture 10.

Molecular Dynamics

Introduction

It is well established that matter consists of discrete units called atoms, and that the atomistic nature of matter has profound impacts on many everyday applications.

It turns out that we can study some of these impacts using the ODE techniques developed in Lecture 10.

For example, the following website shows computer simulations of a crack propagating through a silicon crystal (i.e., rocks).

`https://warwick.ac.uk/fac/sci/eng/staff/jrk/`

The following video shows an example from chemistry.

`https://youtu.be/GClPr5Qpd5A`

These types of simulations are called molecular dynamics simulations.

This lecture will demonstrate how we can perform simple molecular dynamics simulations using Julia's `DifferentialEquations.jl` package.

Molecular Dynamics

Hamiltonian systems

According to Newtonian mechanics, the positions $(x_i(t) \in \mathbb{R}^d)_{i=1}^n$ and velocities $(v_i(t) \in \mathbb{R}^d)_{i=1}^n$ of the atoms satisfy

$$\dot{x}_i = v_i, \qquad \dot{v}_i = F_i\big((x_j)_{j=1}^n\big),$$

where $F_i\big((x_j)_{j=1}^n\big)$ denotes the force on atom i and where I assumed for simplicity that the masses of the atoms are $m = 1$.

In molecular dynamics simulations, the force

$$F_i\big((x_j)_{j=1}^n\big) = -\nabla_{x_i} V\big((x_j)_{j=1}^n\big)$$

is minus the gradient with respect to x_i of a function $V\big((x_i)_{i=1}^n\big)$, which is called the potential energy.

Molecular Dynamics

Hamiltonian systems (continued)

Adding the potential energy to the kinetic energy $\sum_{i=1}^{n} \frac{1}{2} \|v_i\|_2^2$ yields the total energy

$$H\big((x_i),(v_i)\big) = \sum_{i=1}^{n} \tfrac{1}{2} \|v_i\|_2^2 + V\big((x_i)\big)$$

which is also called the Hamiltonian of the system.

The equations of motions can be expressed in terms of the Hamiltonian as

$$\dot{x}_i = \nabla_{v_i} H\big((x_j),(v_j)\big), \qquad \dot{v}_i = -\nabla_{x_i} H\big((x_j),(v_j)\big),$$

where for the first equation I used that

$$\nabla_{v_i}\left(\tfrac{1}{2}\|v_i\|_2^2\right) = \|v_i\|_2 \, \frac{v_i}{\|v_i\|_2} = v_i.$$

The equations of motion are thus fully specified once we provide an expression for the Hamiltonian, which in turn is specified once we provide an expression for the potential energy $V\big((x_i)\big)$.

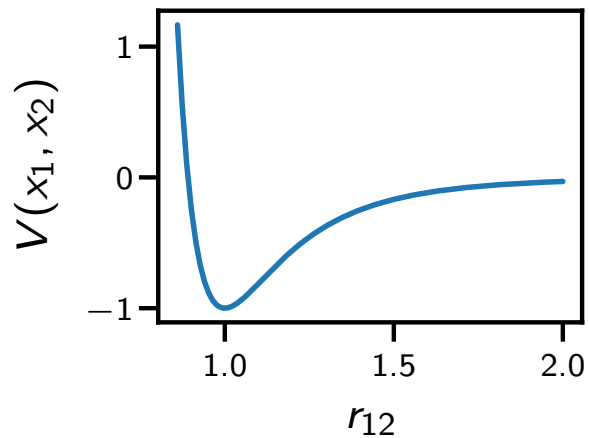
Molecular Dynamics

Lennard-Jones potential

A simple potential energy yielding qualitatively correct results is given by

$$V((x_i)) = \sum_{i<j} \left(r_{ij}^{-12} - 2r_{ij}^{-6} \right), \qquad r_{ij} = \|x_i - x_j\|_2.$$

An intuition for this function can be obtained by considering a system of only $n = 2$ atoms and plotting $V(x_1, x_2)$ as a function of $r_{12} = \|x_1 - x_2\|_2^2$.



The negative slope for $r_{12} < 1$ implies that atoms repel if they get too close, and the positive slope for $r_{12} > 1$ implies that atoms attract if they are far enough apart. Since the slope goes to 0 for $r_{12} \to \infty$, the attraction vanishes if atoms are too far apart.

Molecular Dynamics

Numerical implementation

See the code file.

Some remarks:

- Adaptive time-stepping usually yields no performance benefit for molecular dynamics simulations: the point of adaptive time-stepping is to "zoom in" on those times when things change rapidly, but in a large enough molecular dynamics simulation there will always be some rapid change somewhere in the simulation box. I therefore disable adaptive time-stepping by passing `adaptive = false` to the `solve()` function.
- Our simulation uses periodic boundary conditions: if an atom leaves the simulation box on one side, it reappears on the other side.
  If we did not do this, then the atoms would one-by-one disappear to infinity and never interact again.
  Periodic boundary conditions require us to map the atoms back into the simulation box after every time step. This can be done using a callback function; see `enforce_pbc`.

Molecular Dynamics

Energy conservation

An easy calculation reveals

$$\begin{aligned}\frac{d}{dt} H\big((x_i),(v_i)\big) &= \sum_{i=1}^{n}\Big(\dot{x}_i \cdot \nabla_{x_i} H\big((x_i),(v_i)\big) + \dot{v}_i \cdot \nabla_{v_i} H\big((x_i),(v_i)\big)\Big) \\ &= \sum_{i=1}^{n}\Big(-\dot{x}_i \cdot \dot{v}_i + \dot{v}_i \cdot \dot{x}_i\Big) = 0,\end{aligned}$$

i.e. the total energy $H\big((x_i),(v_i)\big)$ of the system does not change over time.

Of course, it is unreasonable to expect that our numerical simulation preserves the energy exactly. However, `energy_conservation()` shows that not only is the total energy not preserved, it drifts over time.

You might have to hide the potential and kinetic energy from the plot to see the variation in the total energy.

This energy drift is a problem if we are interested in simulating over very long time spans. It can be avoided by using special ODE solvers like `VerletLeapfrog()` which are designed specifically to preserve the energy of the system.

Molecular Dynamics

Stiffness-induced time-step constraints

Recall from Lecture 10 that explicit Runge-Kutta methods incur a step size constraint when applied to converging or oscillating ODEs.

This phenomenon also occurs in molecular dynamics simulations. We can see this by replacing `dt = 0.06` with `dt = 0.07` in `energy_conservation()`.

In molecular dynamics simulations, it is virtually impossible to provide a rigorous mathematical analysis of this phenomenon. However, explaining it on a heuristic level is fairly straightforward.

If two atoms are a distance $r_{ij} \approx 1$ apart, they oscillate against each other with a characteristic frequency $\omega > 0$, i.e.

$$r_{ij}(t) \approx 1 + C \cos(\omega t).$$

This frequency roughly corresponds to an imaginary eigenvalue $\lambda = \omega\iota$ of the Jacobian $\nabla f(y_F)$ of $f(y) = \dot{y}$ at the fixed point $r_{ij} = 1$.

Molecular Dynamics

Stiffness-induced time-step constraints (continued)

To avoid spurious exponential growth of the numerical solution, we must ensure that $\lambda \Delta t = \Delta t \, \omega \imath$ is in the stability region of the Runge-Kutta scheme, or at least close to it.

For explicit Runge-Kutta schemes, this effectively means that we must choose $\Delta t \lesssim \frac{1}{\omega}$, i.e. we must choose the time step Δt small enough such that we have at least a constant number of time steps per oscillation.

A constant number of time steps per oscillation is not an issue if this oscillation is the phenomenon that we are interested in. However, in many molecular dynamics simulations we have some atoms which oscillate against each other with high frequencies but only small amplitudes. These pairs of atoms lead to a separation of time scales and hence a stiff ODE.

This stiffness could be resolved by switching to an implicit ODE solver, but that would require solving a huge nonlinear system of equations in every time step which is not practical either.

Molecular Dynamics

Stiffness-induced time-step constraints (continued)

Instead, computational physicists have recognised that as long as the oscillations have only small amplitudes, we can simply eliminate them by forbidding the atoms to oscillate against each other.

Doing so introduces a small error, but it reduces the separation of time scales and hence allows us to solve the ODE much more efficiently.

Example

Water is often treated as a molecule with fixed bond lengths and angles rather than a system of three atoms for precisely the above reason.