# MA3227 Numerical Analysis II

## Lecture 7: Molecular Dynamics

Simon Etter

**NUS**
National University
of Singapore

Semester II, 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 6.

# 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 6.

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

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

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.

The first step towards this goal is to write down the ODE governing the movement of the atoms. This will be tackled on the next slide.

# Molecular Dynamics

**Newton's equation of motion**

According to Newtonian mechanics, the position $x_i(t) \in \mathbb{R}^d$ and velocity $v_i(t) \in \mathbb{R}^d$ of atom $i \in \{1, \dots, n\}$ satisfy the equations

$$\dot{x}_i = v_i, \qquad \dot{v}_i = F_i\big((x_j)_j\big),$$

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

In friction-less systems, the force is minus the gradient of the potential energy $V\big((x_i)_i\big)$,

$$F_i\big((x_j)_j\big) = -\nabla_{x_i} V\big((x_j)_j\big).$$

The equations of motion are thus fully specified once we provide an expression for the potential energy $V\big((x_i)_i\big)$.

# Molecular Dynamics

**Lennard-Jones potential**

A simple potential energy yielding qualitatively correct results is given by

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

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 $r = \|x_1 - x_2\|_2$.
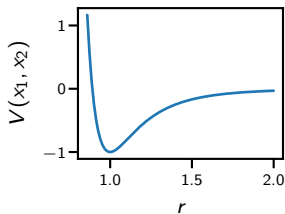


The negative slope for $r_{12} < 1$ implies that atoms repel if they get too close, and the positive slope for $r_{12} > 1$ implies that atoms attract each other if they are far enough apart. Since the slope goes to 0 for $r_{12} \to \infty$, the attraction vanishes if atoms are too far apart.

# Molecular Dynamics

**Lennard-Jones potential (continued)**

The last paragraph on the previous slide described the behaviour of the Lennard-Jones potential for a two-atoms system.

If there are more atoms, then the formula at the top of the previous slide says that the overall potential energy is simply the sum of these two-atom interactions.

# Molecular Dynamics

**Hamiltonian systems**

Newton's equations of motion combined with the Lennard-Jones potential yields the system of ODEs

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

which we can now solve using the Runge-Kutta methods from Lecture 6. One way to do so would be to implement the right-hand-side function

$$f : (x_i, v_i)_i \mapsto \Big(v_i, \, -\nabla_{x_i} V\big((x_j)_j\big)\Big)_i$$

and pass this function to an ODE solver, but `DifferentialEquations` provides a simpler alternative.

# Molecular Dynamics

**Hamiltonian systems (continued)**

This alternative is based on the observation that the above ODE is equivalent to

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

where

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

denotes the total energy of the system. In the context of molecular dynamics simulations, this total energy is also called the *Hamiltonian*; hence the letter $H$.

The equations of motions are thus fully specified once we have implemented the total energy / Hamiltonian.

# Molecular Dynamics

**Numerical implementation**

See the code file up to and including `simulate()`.

Some remarks:

- ▶ Adaptive time-stepping 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.

- ▶ I use periodic boundary conditions, i.e. I make atoms leaving the simulation box on one side reappear on the other side. If I did not do this, then the atoms would one-by-one disappear to infinity and never interact again.
  Periodic boundary conditions require us to check after every time step whether an atom left the simulation box and if so move it to the other end of the box. This can be done using a callback function; see `enforce_pbc`.

# Molecular Dynamics

**Increasing the step size**

We have seen on the last slide that running `simulate()` with the default parameters given in the code seems to work reasonably well and in particular produces physically plausible results.

However, imagine now that you are a time-strapped researcher rushing to get your publication out and hence you do not have the time to watch your simulation unfold. In an attempt to reduce the "time to science", you therefore increase the time step from `dt = 0.01` to `dt = 0.03`.

According to the runtime and convergence theory for Runge-Kutta methods, doing so should buy you a threefold improvement in terms of runtime at the price of a nine times larger error, and since the errors for `dt = 0.01` seem to be small this looks like a worthwhile tradeoff.

Unfortunately, running `simulate()` with this new step size `dt = 0.03` shows that the increase in error is much worse than a factor 9; the simulation goes haywire almost immediately.

The only way to explain this phenomenon is through the Runge-Kutta stability theory developed in Lecture 6; see next slide.

# Molecular Dynamics

**Fixed points of the molecular dynamics ODE**

The first step towards explaining the phenomenon described above is to determine the fixed points of the molecular dynamics ODE and compute the eigenvalues of the Jacobian around these fixed points.

Doing so for a system of $n$ atoms would be quite difficult, but it turns out that we can get all the relevant insights by considering a system of only two atoms in one dimension. In this case, the positions of the atoms are encoded by two scalars $x_1, x_2 \in \mathbb{R}$, and without loss of generally we can assume $x_1 < x_2$.

# Molecular Dynamics

**Fixed points of the molecular dynamics ODE (continued)**

The equations of motion are then given by

$$\ddot{x}_1 = -\frac{\partial}{\partial x_1} V(x_1, x_2) = -12r^{-13} + 12r^{-7},$$
$$\ddot{x}_2 = -\frac{\partial}{\partial x_2} V(x_1, x_2) = \phantom{-}12r^{-13} - 12r^{-7},$$

where

$$r = x_2 - x_1 \qquad \text{and} \qquad V(x_1, x_2) = r^{-12} - 2r^{-6}$$

denotes the Lennard-Jones potential from slide 5.

This system of coupled ODEs can be reduced to a single ODE

$$\ddot{r} = \ddot{x}_2 - \ddot{x}_1 = 24\left(r^{-13} - r^{-7}\right)$$

which has precisely one fixed point, namely $r = 1$.

# Molecular Dynamics

**Fixed points of the molecular dynamics ODE (continued)**

We have

$$\tfrac{d}{dr} 24\left(r^{-13} - r^{-7}\right)\Big|_{r=1} \;=\; 24\left(-13 + 7\right) \;=\; 144;$$

thus the linearisation of this ODE around $r = 1$ is given by

$$\ddot{r} = -144\,(r-1)^2 + O\big((r-1)^3\big).$$

Ignoring the higher-order term, the solution to this ODE is given by

$$r(t) = 1 + a\,\cos(12t) + b\,\sin(12t),$$

where $a, b \in \mathbb{R}$ denote two constants determined by the initial conditions.

# Molecular Dynamics

**Fixed points of the molecular dynamics ODE (continued)**

In physical terms, this result means that if we place two atoms at a distance $r$ close but not quite equal to 1, then these atoms will oscillate against each other for all eternity.

In mathematical terms, this finding implies that the Jacobian of the ODE right-hand side evaluated at the fixed point has a pair of eigenvalues $\lambda = \pm 12i$, where

- $\text{real}(\lambda) = 0$ because the oscillation is of constant amplitude, and
- $\text{imag}(\lambda) = \pm 12$ because that is the frequency of oscillation.

This result has been derived in the special case of two atoms and one dimension, but it is clear that "pairs of atoms oscillating against each other" can and will also occur in molecular dynamics simulations with arbitrarily many atoms and in higher dimensions.

Let us therefore assume that the molecular dynamics ODE has fixed points with Jacobian eigenvalues $\lambda = \pm 12i$ regardless of the dimension and number of atoms.
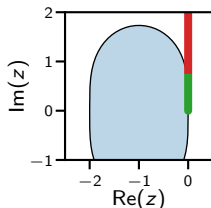
# Molecular Dynamics

**Stability function**

The second step towards explaining our observations on slide 10 is to insert the Jacobian eigenvalues determined above into the stability function of the trapezoidal rule scheme used by `simulate()`.

We have seen in Lecture 6 that the stability function of this scheme is

$$R(z) = 1 + z + \frac{z^2}{2}.$$

Plotting the resulting stability domain along with the ray $i \times [0, \infty)$,



we observe that $|R(12i \, \Delta t)| \approx 1$ for small values of $\Delta t$, but $|R(12i \, \Delta t)| \gg 1$ if $\Delta t$ is larger than some critical step size.

The simulation is hence unstable in the sense that $|R(12i \, \Delta t)| > 1$ for all values of $\Delta t$, but the instability is only mild for small values of $\Delta t$.

## Molecular Dynamics

**Explaining the blow-up (continued)**

We can quantify the above qualitative insight by observing that $k$ midpoint steps with step size $\Delta t$ spuriously increase the amplitude of the oscillations in $\tilde{y}(t)$ by a factor

$$\left| R(12i\,\Delta t) \right|^{k} = \exp\left( k \, \log |R(12i\,\Delta t)| \right) = \exp\left( t \, \frac{\log |R(12i\,\Delta t)|}{\Delta t} \right).$$

This factor is approximately one for

$$t \lesssim T_{\star}(\Delta t) = \frac{\Delta t}{\log |R(12i\,\Delta t)|}$$

but will grow very rapidly for $t \gtrsim T_{\star}(\Delta t)$.

$T_{\star}(\Delta t)$ thus gives us a rough indication of the time scale over which we expect the simulation to be stable, and we indeed observe excellent qualitative agreement between this theoretical expectation and empirical observations when comparing the values

$$T_{\star}(0.01) \approx 400, \qquad T_{\star}(0.02) \approx 50, \qquad T_{\star}(0.03) \approx 15$$
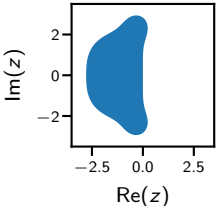
against the output of `simulate()`.

# Molecular Dynamics

**Avoiding the blow-up**

Now that we understand how the phenomenon described on slide 10 arises, we can next look into how it can be avoided.

Perhaps the simplest way to improve the situation is to consider an explicit Runge-Kutta scheme with a larger stability domain. For example, the stability function and stability domain of the RK4 scheme are given by

$$R(z) = \sum_{k=0}^{4} \frac{z^k}{k!} \qquad \text{and}$$



,

respectively, and since the stability domain reaches much further along the imaginary axes, we conclude that this scheme should be able to take much larger steps without blowing up the simulation.

Indeed, we observe that RK4 appears to be stable even with step sizes as large as $\mathtt{dt} = 0.1$, but of course the issue of exploding simulations eventually reappears for large enough $\mathtt{dt}$ (specifically, $\mathtt{dt} = 0.14$).

# Molecular Dynamics

**Avoiding the blow-up (conclusion)**

We have seen in Lecture 6 that stable simulations regardless of the step size can only be attained if we switch to an implicit Runge-Kutta scheme, but we have also seen that doing so would require us to solve a nonlinear system of equations $F(X) = 0$ where the function $F(X)$ involves the highly nonlinear Lennard-Jones potential and the unknowns $X$ are the positions and velocities of all $n$ atoms.

Solving such a system of equations efficiently is virtually impossible. Implicit Runge-Kutta schemes are hence rarely used for molecular dynamics simulations, and time step constraints are generally considered unavoidable instead.

Having said this, it is worth taking another look at why we are interested in large time steps in the first place. The answer to this question is not as straightforward as one might think; see next slide.

# Molecular Dynamics

**Separation of time scales in molecular dynamics**

On second thought, it may appear that while time step constraints in molecular dynamics simulations are perhaps somewhat surprising, they should actually be fairly harmless. The argument leading to this conclusion goes roughly as follows.

▶ For the most part, the purpose of a molecular dynamics simulation is of course to illustrate the movement of atoms.

▶ Hence when running such a simulation, we anyway want to compute snapshots of the ODE solution at a frequency roughly proportional to the frequency of oscillation of the atoms.

▶ This frequency of oscillation is also what determines the step size constraint. Hence we do not have a separation of time scales and the step size constraint should not be a problem.

This argument and its conclusion are correct for simple simulations like the one implemented in `simulate()`, but it breaks down once we consider more complex molecular dynamics simulations.

# Molecular Dynamics

**Separation of time scales in molecular dynamics (continued)**

Specifically, separation of time scales arises as soon as we introduce molecules into our simulation.
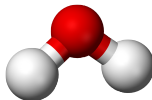


Molecules are collections of atoms held together by very strong bonds, and these bonds cause the atoms to oscillate against each other with very high frequencies but small amplitudes.

The separation of time scales then arises because the high-frequency internal oscillations of the molecules force us to take small time steps while the relatively slow movements of the molecules as a whole force us to consider long time intervals.

See https://www.youtube.com/watch?v=GClPr5Qpd5A for a full-fledged example of such a simulation.

# Molecular Dynamics

**Separation of time scales in molecular dynamics (continued)**

Perhaps the best way to avoid excessively small time steps due to molecular vibrations would be to consider an implicit time stepping scheme, but as pointed out earlier this is rarely feasible in molecular dynamics simulations.

Instead, the most commonly employed approach to avoid small time steps is to "freeze" the internal degrees of freedom and consider a molecule as a single rigid object instead.

This approach has two important drawbacks:

▶ Molecule are not rigid bodies; hence simulating them as such introduces some modeling errors.

▶ Rigid body dynamics are significantly more complicated than the point particle dynamics discussed earlier in this lecture.

However, it turns out that these drawbacks are outweighed by the fact that freezing the high-frequency oscillations reduces the separation of time scales and hence significantly expands the type of systems we can simulate.