Molecular dynamics simulation

CS/CME/BioE/Biophys/BMI 279
Oct. 3, 2019
Ron Dror
Outline

- Molecular dynamics (MD): The basic idea
- Equations of motion
- Key properties of MD simulations
- Sample applications
- Limitations of MD simulations
- Software packages and force fields
- Accelerating MD simulations
- Monte Carlo simulation

Monte Carlo is a recurring idea in this class and is one way to sample protein conformations, without requiring one to iterate through every single step of protein folding/motion (described later in this lecture).
Molecular dynamics: The basic idea
The idea

• Mimic what atoms do in real life, assuming a given potential energy function
  – The energy function allows us to calculate the force experienced by any atom given the positions of the other atoms
  – Newton’s laws tell us how those forces will affect the motions of the atoms

We use potential energy to derive forces acting on our system, which we can then translate into acceleration, velocity, and position.

Like rolling a ball along a multi-dimensional, friction-less surface and observing it as it traverses along the energy landscape. Like a real ball with momentum, the conformation of the atom won’t stay static in one position.
Basic algorithm

- Divide time into discrete time steps, no more than a few femtoseconds ($10^{-15}$ s) each

- At each time step:
  - Compute the forces acting on each atom, using a molecular mechanics force field
  - Move the atoms a little bit: update position and velocity of each atom using Newton’s laws of motion

Here we sum all of the forces (described in previous lecture) on each single atom
Molecular dynamics movie
Equations of motion
Equations of motion

• Newton’s second law: \( F = ma \)
  – where \( F \) is force on an atom, \( m \) is mass of the atom, and \( a \) is the atom’s acceleration

• Recall that: \( F(x) = -\nabla U(x) \)
  – where \( x \) represents coordinates of all atoms, and \( U \) is the potential energy function

• Velocity is the derivative of position, and acceleration is the derivative of velocity.

• We can thus write the equations of motion as:

\[
\frac{dx}{dt} = v \\
\frac{dv}{dt} = \frac{F(x)}{m}
\]

It helps to use this form of Newton’s second law: \( a = \frac{F}{m} \), as we are interested in the acceleration of an atom given the force found from the molecular mechanic forcefield. We can plug in the masses of individual atoms.
Solving the equations of motion

\[
\begin{align*}
\frac{dx}{dt} &= v \\
\frac{dv}{dt} &= \frac{F(x)}{m}
\end{align*}
\]

For a system with \( n \) atoms:

- This is a system of ordinary differential equations
  - For \( n \) atoms, we have \( 3n \) position coordinates and \( 3n \) velocity coordinates
- “Analytical” (algebraic) solution is impossible
- Numerical solution is straightforward

\[
\begin{align*}
\mathbf{x}_{i+1} &= \mathbf{x}_i + \delta_t \mathbf{v}_i \\
\mathbf{v}_{i+1} &= \mathbf{v}_i + \delta_t \frac{F(\mathbf{x}_i)}{m}
\end{align*}
\]

- where \( \delta_t \) is the time step

At \( v_0 \), we assign velocities to each atom based on experimental evidence. When simulating, it is common practice to run multiple trials with different sets of initial velocities.
Solving the equations of motion

• Straightforward numerical solution:
  \[ x_{i+1} = x_i + \delta t v_i \]
  \[ v_{i+1} = v_i + \delta t \frac{F(x_i)}{m} \]

• In practice, people use “time symmetric” integration methods such as “Leapfrog Verlet”
  \[ x_{i+1} = x_i + \delta t \frac{v_{i+1/2}}{2} \]
  \[ v_{i+1/2} = v_{i-1/2} + \delta t \frac{F(x_i)}{m} \]

  - This gives more accuracy
  - You’re not responsible for this

Real motions are time reversible, the positions computed should be derivable whether we are moving forwards or backwards in time.

In the above method, we can’t compute positions backward in time as well. This is because we consider the velocity at value at i, despite the velocity changing between step i and i+1. One way to achieve reversibility is to use positions and velocities 1/2-time step apart from each other.
Key properties of MD simulations
Atoms never stop jiggling

- In real life, and in an MD simulation, atoms are in constant motion.
  - They will *not* go to an energy minimum and stay there.
- Given enough time, the simulation samples the Boltzmann distribution
  - That is, the probability of observing a particular arrangement of atoms is a function of the potential energy
  - In reality, one often does not simulate long enough to reach all energetically favorable arrangements
  - This is not the only way to explore the energy surface (i.e., sample the Boltzmann distribution), but it’s a pretty effective way to do so
Energy conservation

• Total energy (potential + kinetic) should be conserved
  – In atomic arrangements with lower potential energy, atoms move faster
  – In practice, total energy tends to grow slowly with time due to numerical errors (rounding errors)
  – In many simulations, one adds a mechanism to keep the temperature roughly constant (a “thermostat”)

Our “thermostat” will act to decrease energy over time to counteract the rounding error that increases energy
Water is important

• Ignoring the solvent (the molecules surrounding the molecule of interest) leads to major artifacts
  – Water, salt ions (e.g., sodium, chloride), lipids of the cell membrane

• Two options for taking solvent into account
  – Explicitly represent solvent molecules
    • High computational expense but more accurate
    • Usually assume periodic boundary conditions (a water molecule that goes off the left side of the simulation box will come back in the right side, like in PacMan)
  – Implicit solvent
    • Mathematical model to approximate average effects of solvent
    • Less accurate but faster

Solvent molecules often make up > 90% of simulated atoms

If a water molecule is on/close to the boundary, we model it's interaction with molecules on the opposite border to avoid edge errors

This is less accurate, especially in cases where individual water molecules play an important role

Can also fail when we need to take into account other molecules, like ions, that are in the solvent
Explicit solvent

Water (and ions)

Cell membrane (lipids)

Protein
Sample applications
Determining where drug molecules bind, and how they exert their effects

You can simulate how a molecule binds to a protein, and in turn, what effect binding will have on protein structure. You can use this information to design new molecules that bind and cause a different effect.

Dror et al., *Nature* 2013
Determining functional mechanisms of proteins

- We performed simulations in which a receptor protein transitions spontaneously from its active structure to its inactive structure.
- We used these to describe the mechanism by which drugs binding to one end of the receptor cause the other end of the receptor to change shape (activate).

Beta blockers stabilize inactive state
Beta agonists stabilize active state

Simulation started from active structure vs. Inactive structure

Rosenbaum et al., *Nature* 2010; Dror et al., *PNAS* 2011
Understanding the process of protein folding

This is often not the most accurate or most computationally efficient way to predict protein structure. However, you can watch the folding of the protein, which can be informative. One might validate the folding simulation by comparing specific actions of the folding to experimental data. E.g. inducing a mutation of a single amino acid and seeing what effect the mutation has on folding speed.

• For example, in what order do secondary structure elements form?
• But note that MD is generally not the best way to predict the folded structure.

Lindorff-Larsen et al., *Science* 2011
MD simulation isn’t just for proteins

• Simulations of other types of molecules are common
  – All of the biomolecules discussed in this class
  – Materials science simulations
Limitations of MD simulations

Closest to the physics
Timescales

• Simulations require short time steps for numerical stability
  – 1 time step ≈ 2 fs ($2 \times 10^{-15}$ s)
  
  Certain bonds have modes of vibration that are ~10s of fs, so the time step has to be short enough to account for this

• Structural changes in proteins can take nanoseconds ($10^{-9}$ s), microseconds ($10^{-6}$ s), milliseconds ($10^{-3}$ s), or longer
  – Millions to trillions of sequential time steps for nanosecond to millisecond events (and even more for slower ones)

• Until recently, simulations of 1 microsecond were rare

• Advances in computer power have enabled microsecond simulations, but simulation timescales remain a challenge

• Enabling longer-timescale simulations is an active research area, involving:
  – Algorithmic improvements
  – Parallel computing
  – Hardware: GPUs, specialized hardware
Force field accuracy

• Molecular mechanics force fields are inherently approximations. Specific values are essentially impossible to calculate, given the sheer number of atoms that need to be tracked.

• They have improved substantially over the last decade, but many limitations remain. Here force fields with lower scores are better, as assessed by agreement between simulations and experimental data. Even the force fields with scores of zero are imperfect, however!

 Experimental derived from NMR, which measures rapid protein dynamics

  Lindorff-Larsen et al., PLOS One, 2012

• In practice, one needs some experience to know what to trust in a simulation.
Covalent bonds cannot break or form during (standard) MD simulations

- Once a protein is created, most of its covalent bonds do not break or form during typical function.
- A few covalent bonds do form and break more frequently (in real life):
  - Disulfide bonds between cysteines
  - Acidic or basic amino acid residues can lose or gain a hydrogen (i.e., a proton)

There are some workarounds to this limitation, like using quantum mechanics (computationally expensive) to understand a specific interaction, while modeling everything else with MD simulation.

This limitation may not be an issue in certain applications, such as drug-binding interactions.
Software packages and force fields
(These topics are not required material for this class, but they’ll be useful if you want to do MD simulations)
Software packages

- Multiple molecular dynamics software packages are available; their core functionality is similar - GROMACS, AMBER, NAMD, Desmond, OpenMM, CHARMM

- Dominant package for visualizing results of simulations: VMD (“Visual Molecular Dynamics”)

MD software packages are continually being improved - different software packages are preferred in for specific types of proteins / interactions.

VMD vs. PyMOL

PyMOL - good for observing individual structures
VMD - great for observing lots of protein structures/simulation
Force fields for molecular dynamics

• Three major force fields are used for MD
  – CHARMM, AMBER, OPLS-AA
    • Multiple versions of each
  – Do not confuse CHARMM and AMBER force fields with CHARMM and AMBER software packages

• They all use strikingly similar functional forms
  – Common heritage: Lifson’s “Consistent force field” from mid-20th-century
Accelerating MD simulations
Why is MD so computationally intensive?

• Many time steps (millions to trillions)
• Substantial amount of computation at every time step
  – Dominated by non-bonded interactions, as these act between every pair of atoms.
    • In a system of $N$ atoms, the number of non-bonded terms is proportional to $N^2$
  – Can we ignore interactions beyond atoms separated by more than some fixed cutoff distance?
    • For van der Waals interactions, yes. These forces fall off quickly with distance.
    • For electrostatics, no. These forces fall off slowly with distance.
How can one speed up MD simulations?

- Reduce the amount of computation per time step
- Reduce the number of time steps required to simulate a certain amount of physical time
- Reduce the amount of physical time that must be simulated
- Parallelize the simulation across multiple computers
- Redesign computer chips to make this computation run faster

I want you to understand why simulations are computationally expensive and slow, and to have a sense of the types of things people try to speed them up. You are not responsible for the details of these speed-up methods.
How can one speed up MD simulations?

• Reduce the amount of computation per time step
  – Faster algorithms
  – Example: fast approximate methods to compute electrostatic interactions, or methods that allow you to evaluate some force field terms every other time step.

• Reduce the number of time steps required to simulate a certain amount of physical time
  – One can increase the time step several fold by freezing out some very fast motions (e.g., certain bond lengths).

• Reduce the amount of physical time that must be simulated
  – A major research area involves making events of interest take place more quickly in simulation, or making the simulation reach all low-energy conformational states more quickly.
  – For example, one might apply artificial forces to pull a drug molecule off a protein, or push the simulation away from states it has already visited.
  – Each of these methods is effective in certain specific cases.
Parallelize the simulation across multiple computers

• Splitting the computation associated with a single time step across multiple processors requires communication between processors.
  
  – Usually each processor takes responsibility for atoms in one spatial region.
  – Algorithmic improvements can reduce communication requirements.

• Alternative approach: perform many short simulations.
  – One research goal is to use short simulations to predict what would have happened in a longer simulation.
Redesign computer chips to make this computation run faster

• GPUs (graphics processor units) are now widely used for MD simulations. They pack more arithmetic logic on a chip than traditional CPUs, and give a substantial speedup.
  – Parallelizing across multiple GPUs is difficult.
• Several projects have designed chips especially for MD simulation
  – These pack even more arithmetic logic onto a chip, and allow for parallelization across multiple chips.
Monte Carlo simulation
Monte Carlo simulation

Instead of simulating the entire process of folding/interaction, we will make educated guesses of where the protein will move based on the MD force field. We will build in a random amount of noise, so that we can explore many minima.

- An alternative method to discover low-energy regions of the space of atomic arrangements
- Instead of using Newton’s laws to move atoms, consider *random* moves
  - For example, consider changes to a randomly selected dihedral angle, or to multiple dihedral angles simultaneously
  - Examine energy associated with resulting atom positions to decide whether or not to “accept” (i.e., make) each move you consider
Metropolis criterion ensures that simulation will sample the Boltzmann distribution

- The Metropolis criterion for accepting a move is:
  - Compute the potential energy difference ($\Delta U$) between the pre-move and post-move position
    - $\Delta U < 0$ if the move would decrease the energy
  - If $\Delta U \leq 0$, accept the move
  - If $\Delta U > 0$, accept the move with probability

$$e^{-\frac{\Delta U}{k_B T}}$$

We accept moves that increase $U$ w/ some probability, so we don’t get stuck in local minima

- After you run such a simulation for long enough, the probability of observing a particular arrangement of atoms is given by the Boltzmann distribution

$$p(x) \propto \exp\left(\frac{-U(x)}{k_B T}\right)$$

- If one gradually reduces the temperature $T$ during the simulation, this becomes a minimization strategy (“simulated annealing”).

When you get to very low temperature, it is much harder to escape local minima, as energy contributes more significantly to probability
Talk: Janet Iwasa
“Animating Molecular Machines”

- Tuesday, Oct. 8, 4:15 p.m.
- Beckman Center, Munzer Auditorium