Molecular dynamics simulation

CS/CME/BioE/Biophys/BMI 279

Oct. 5 and 10, 2017

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

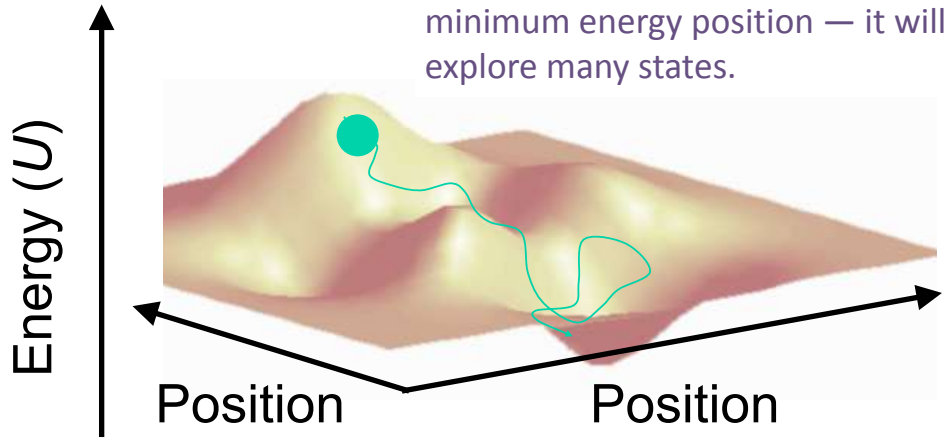
We cover molecular dynamics first because it is an approach that attempts to implement all the physics we just discussed in order to simulate and track the movement of all atoms in a system. In many cases, this is not the best approach (and it is useful to consider why), but serves as a good working example.

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

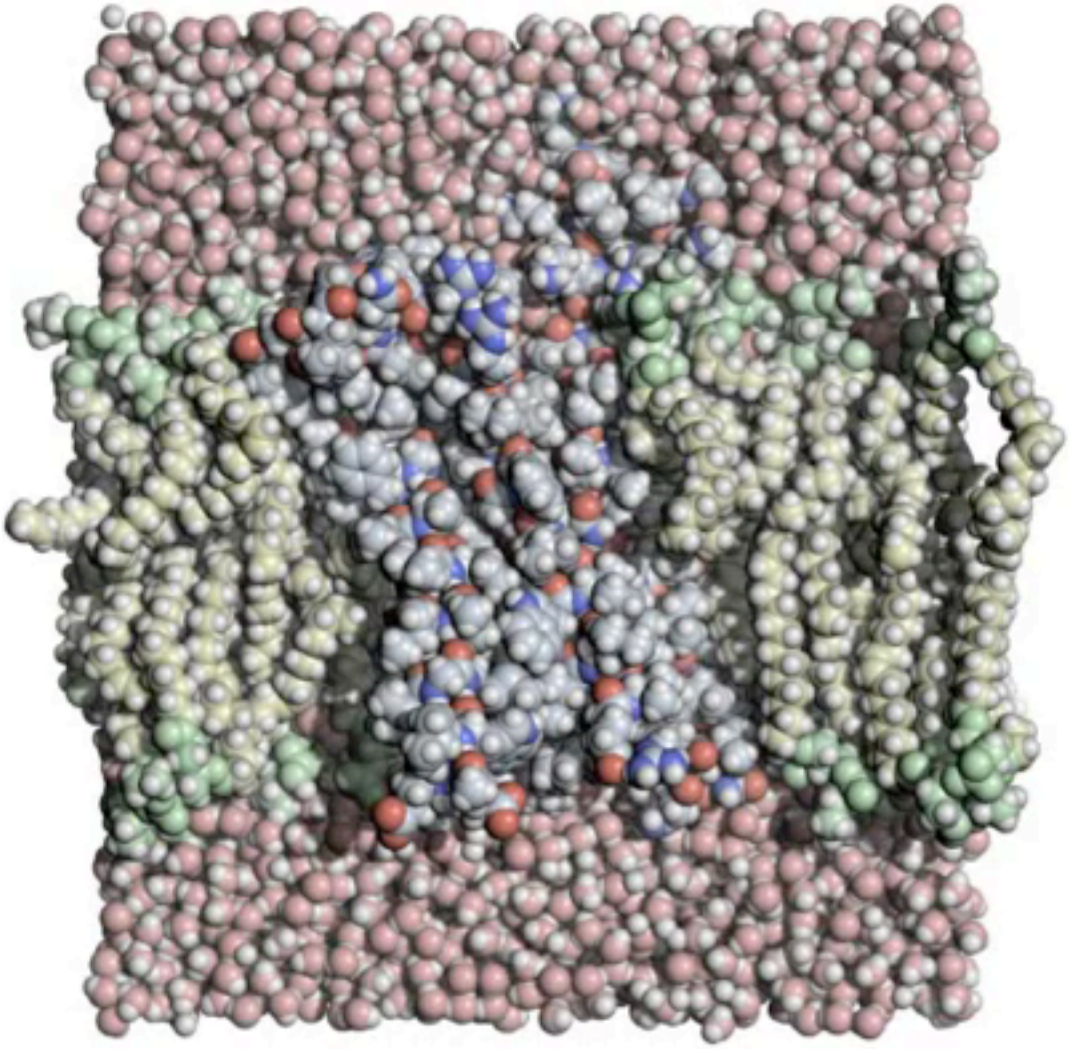
This is like rolling a ball on the (friction-less) surface of the energy function. Like a real ball, it wouldn't just find the minimum energy position — it will have momentum and explore many states.



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

Molecular dynamics movie



Equations of motion

Equations of motion

- Newton's second law: $\boldsymbol{F} = m\boldsymbol{a}$
  - where $\boldsymbol{F}$ is force on an atom, m is mass of the atom, and $\boldsymbol{a}$ is the atom's acceleration
- Recall that: $F(\boldsymbol{x}) = -\nabla U(\boldsymbol{x})$ <— see lecture 3
  - where $\boldsymbol{x}$ 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:

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{v}$$

$$\frac{d\boldsymbol{v}}{dt} = \frac{F(\boldsymbol{x})}{\boldsymbol{m}}$$

This equals acceleration, by Newton's second law.

Solving the equations of motion

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{v}$$

$$\frac{d\boldsymbol{v}}{dt} = \frac{F(\boldsymbol{x})}{\boldsymbol{m}}$$

Clarification: the mass m is written as a vector because it records the mass of every atom.

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

Solving analytically for even three atoms is difficult! ("Three body problem")

Position of atom x at time (i+1) is its position at time i plus how much it moves during the time step.

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \delta_t \boldsymbol{v}_i$$

$$\boldsymbol{v}_{i+1} = \boldsymbol{v}_i + \delta_t F(\boldsymbol{x}_i) / \boldsymbol{m}$$

  - where δ_t is the time step

In practice, this is very computationally expensive! Typically have ~10,000s of atoms and millions of time steps. But you can do a lot of useful simulations with modern GPUs.

Solving the equations of motion

- Straightforward numerical solution:

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \delta_t \boldsymbol{v}_i$$

$$\boldsymbol{v}_{i+1} = \boldsymbol{v}_i + \delta_t F(\boldsymbol{x}_i) / \boldsymbol{m}$$

- In practice, people use "time symmetric" integration methods such as "Leapfrog Verlet"

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \delta_t \boldsymbol{v}_{i+1/2}$$

$$\boldsymbol{v}_{i+1/2} = \boldsymbol{v}_{i-1/2} + \delta_t F(\boldsymbol{x}_i) / \boldsymbol{m}$$

i.e. the times at which position and velocity are computed are offset from each other by 1/2 of a time step.

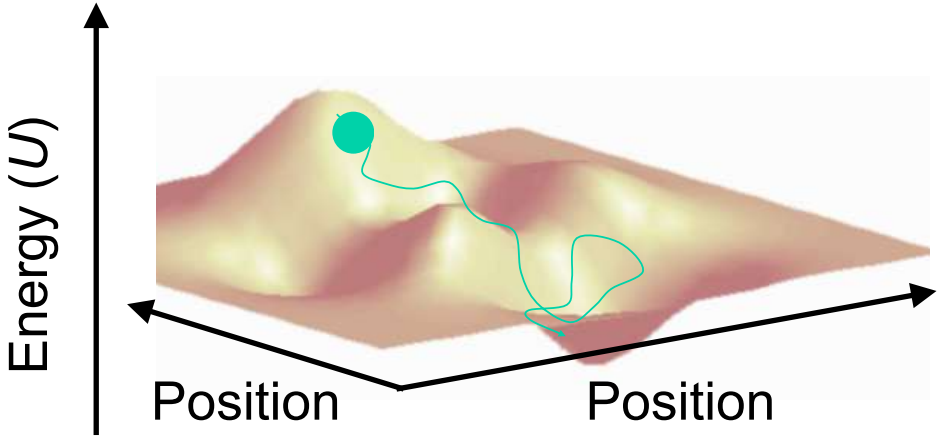
  - This gives more accuracy
  - You're not responsible for this

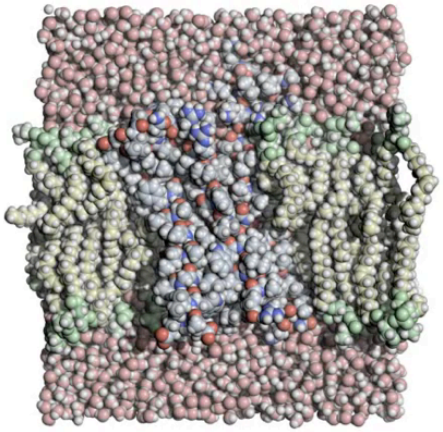
This is more accurate because of "rounding error" — the error that accumulates because computers can only represent a limited number of digits when performing calculations. Time symmetry helps reduce this error.

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

  Same as in isolated systems in high school physics. There is no equivalent of "friction" in these simulations, since you are including all atoms

  - 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")

The thermostat will dampen the increase in energy over time. This is similar to how, in the native protein context, the surrounding atoms will absorb energy from the system if the system increases in 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

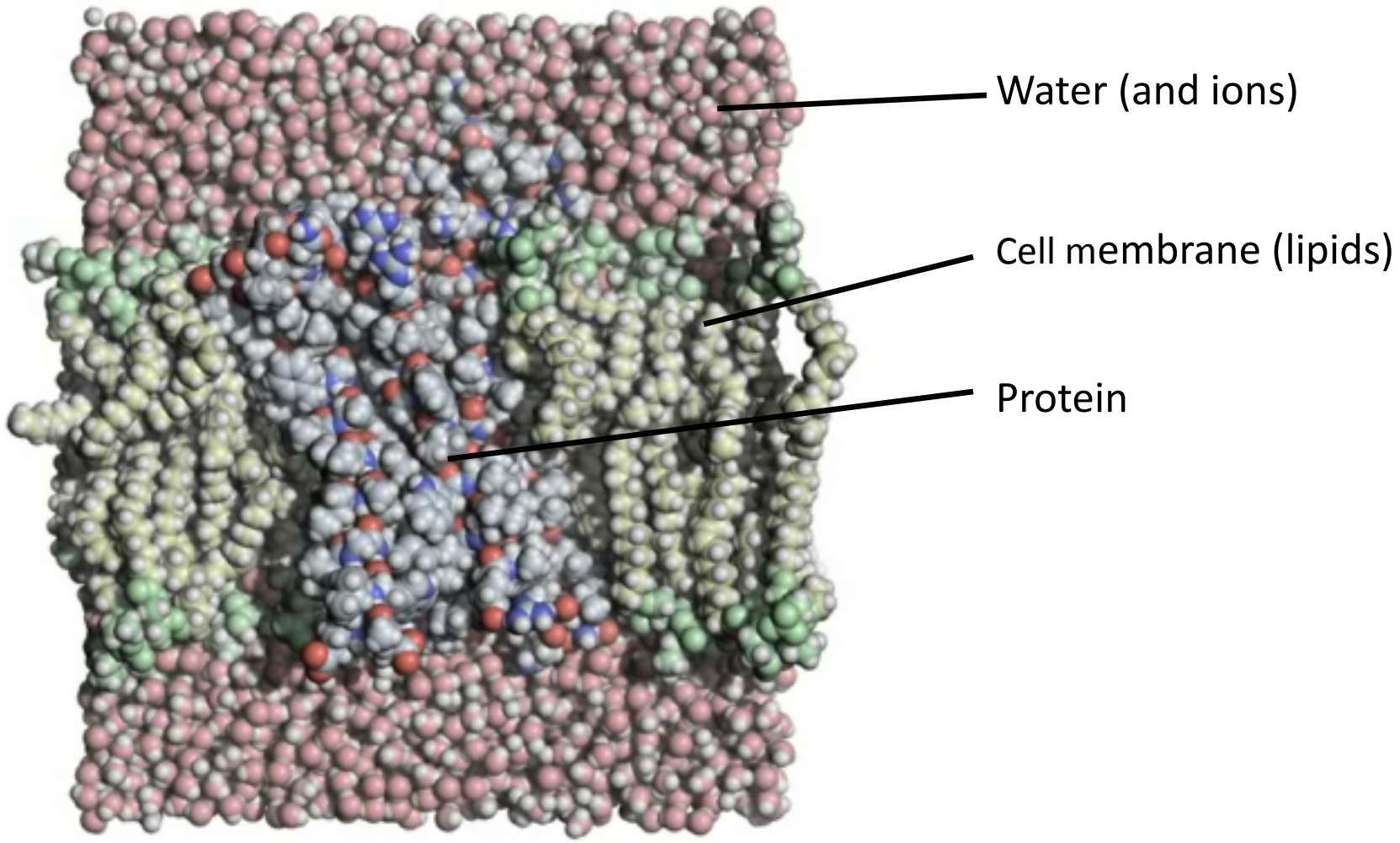
    Typically, around 90% of the atoms in the simulation will be from the solvent!
    - High computational expense but more accurate
    - Usually assume periodic boundary conditions (a water molecule that goes off the left side of the simulation box will come back in the right side, like in PacMan)

    Periodic boundaries is like pretending you have another copy of the whole system at each edge of your system (usually cube shaped). You can't just assume that there's a vacuum outside the system, because water molecules at the boundary will line up in an orderly manner.
  - Implicit solvent
    - Mathematical model to approximate average effects of solvent
    - Less accurate but faster

      Also need to implicitly account for the effect of salt ions in the solvent.

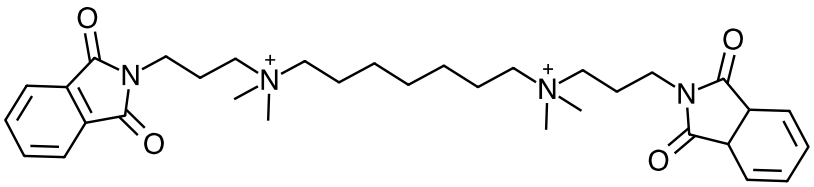
This method is also limited because sometimes the position of a single water molecule can be important, especially within or immediately surrounding the protein. (e.g. specifically positioned water molecules can stabilize a conformation by forming hydrogen bonds.)

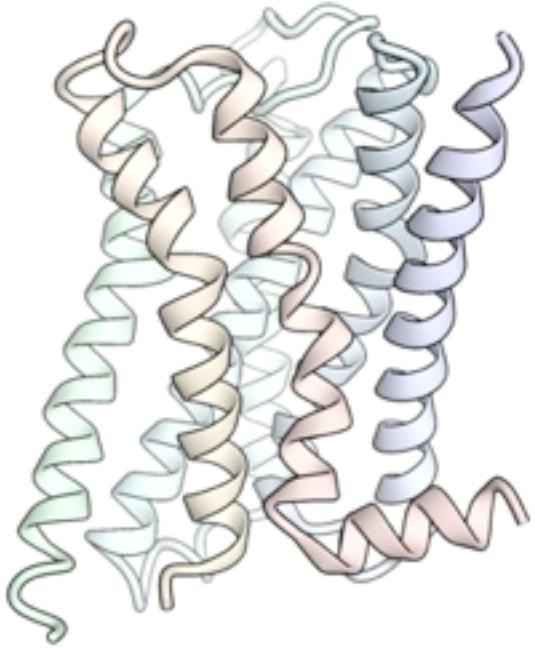
Explicit solvent



Sample applications

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



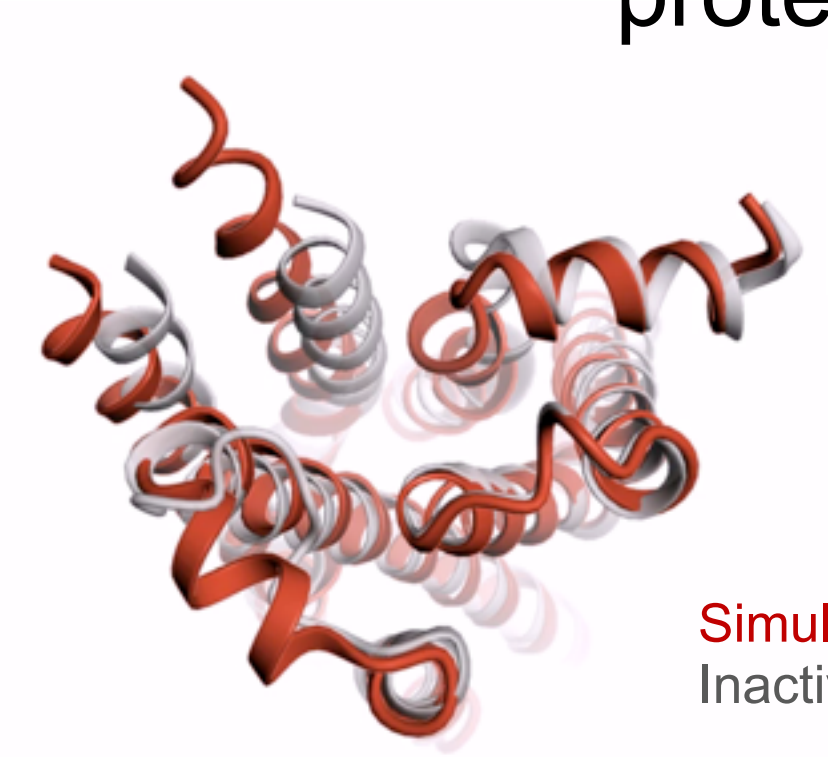


We used simulations to determine where this molecule binds to its receptor, and how it changes the binding strength of molecules that bind elsewhere (in part by changing the protein's structure). We then used that information to alter the molecule such that it has a different effect.

In simulations, you can watch the molecule bind to the protein and observe the structural effect on the protein. Using this information, you can then design new molecule variants that bind and produce a different effect.

Dror et al., *Nature* 2013

Determining functional mechanisms of proteins



This receptor is similar to the one on the previous slide — both are GPCRs. This view is from the inside of the cell (bottom side in the previous slide). The structure of the receptor in its inactive and active conformations have been solved experimentally.

Simulation started from active structure vs.
Inactive structure

- 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)

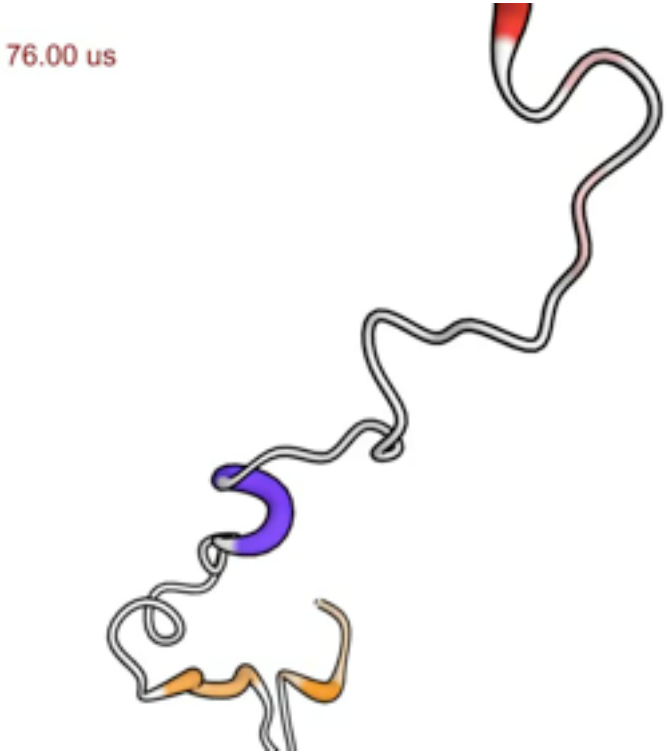
Rosenbaum et al., *Nature* 2010; Dror et al., *PNAS* 2011

Understanding the *process* of protein folding

This is often not the most accurate nor the most computationally efficient way to predict protein structure. However, you can watch the folding of the protein, which can be informative.

Question: how do we know if the order of domain folding is right?
A: In general, you can compare simulations to known experimental data to validate it, and then use the simulations to understand things that you can't understand via experiment.

Question: do proteins start folding during translation as they exit from the ribosome?
A: Experiments show that if you completely unfold a protein (denature) by putting it in a bad solvent, and then refold it by putting it in a better solvent, then it will spontaneously fold back. In practice, it is thought that order of translation doesn't matter too much.



Lindorff-Larsen et al., *Science* 2011

- 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

Limitations of MD simulations

Timescales

- Simulations require short time steps for numerical stability
  - 1 time step ≈ 2 fs (2×10^{-15} s)

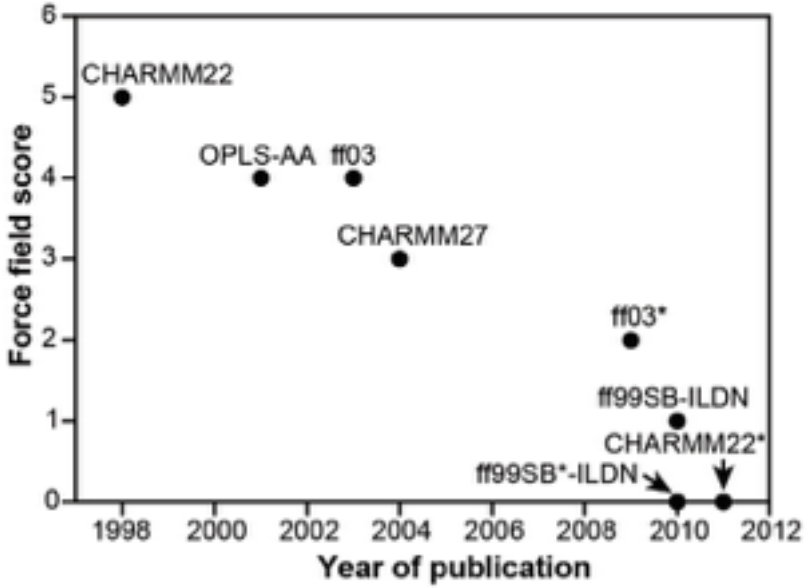
This is because certain bonds have modes of vibration that are ~10s of fs, so your time step has to be shorter than this.

- 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

Some proteins fold in microseconds, but many require milliseconds or seconds.

Force field accuracy

- Molecular mechanics force fields are inherently approximations
- 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 data are 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)

For many applications, this limitation is not too bad. e.g. if you are focusing on how a drug binds to a protein target.

Note: there are people working on special MD simulations that allow bonds to form/break during simulation.

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
  - AMBER, NAMD, GROMACS, Desmond, OpenMM, CHARMM
- Dominant package for visualizing results of simulations: VMD (“Visual Molecular Dynamics”)

All of these are good, although CHARMM was one of the first and its code is outdated.
AMBER was also one of the first, but its code was completely rewritten.

Force fields for molecular dynamics

- Three major force fields are used for MD
  - CHARMM, AMBER, OPLS-AA
  - 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

It used to be that MD software packages used only built-in force fields. These days, most MD software supports a variety of force fields.

Note that CHARMM and AMBER refer to both MD software packages and force fields!

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. e.g. van der Waals or electrostatics
    - 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.

e.g. you can "coarse grain" by grouping atoms as "super-atoms" and computing their interactions with other super-atoms

Current algorithms use Fourier transform and achieve ~ n log(n) time.

- 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).

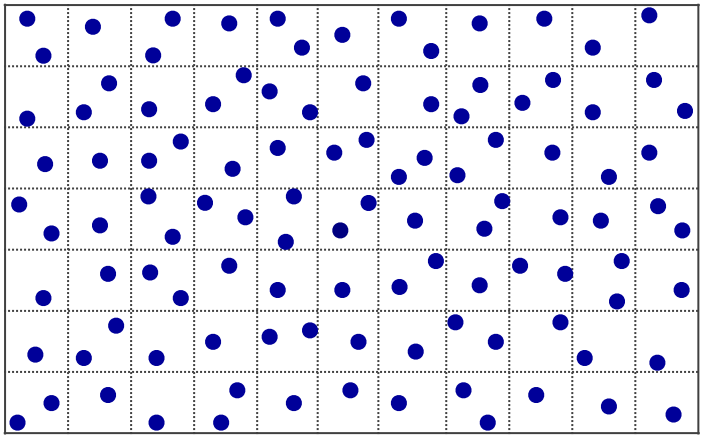
e.g. if you freeze bonds to hydrogens, which have the fastest fluctuations, you can increase the time step by 4-5x

- 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.

But so far no general methods, there's still a tradeoff.

Parallelize the simulation across multiple computers

- Splitting the computation associated with a single time step across multiple processors requires communication between processors.



Sometimes adding too many compute nodes actually slows things down, because of the overhead for each machine to communicate. You can improve the parallelization algorithm to a certain extent, to spend less time doing communication.

Number of atoms per unit volume in a system is roughly constant, which makes this load balancing easier in most cases.

  - 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.

The Pande group at Stanford has done a lot of work in this area.

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.

GPUs started off as highly specialized chips for graphics (i.e., rendering what's on your monitor — computer games benefit tremendously from GPUs!). Over time, companies making GPUs added flexibility to the kinds of computation they can perform. People realized how to map MD computations to the kinds of computations GPUs are really good at. They're also now very widely used for machine learning. A single GPU can now get you the same kind of processing you'd get out of a mid-size cluster of CPUs.

GPU

Specialized chip

Professor Dror worked at DE Shaw Research before coming to Stanford, designing specialized chips for MD. At least half a dozen such projects preceded Anton, but Anton has been especially successful.

Monte Carlo simulation

Monte Carlo simulation

Big idea: Start with a MD force field. For some applications, we don't care EXACTLY how atoms move as a function of time. We're more interested in finding the low energy regions of the Boltzmann distribution.

- 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

You get to choose the rules that specify the "random move"

If you don't accept the move, you just throw it out and continue to use your last accepted configuration.

Metropolis criterion ensures that simulation will sample the Boltzmann distribution

- The Metropolis criterion for accepting a move is:
  - Compute the potential energy difference (ΔU) 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

This should make sense — you've made a move that made the potential energy more favorable, so why wouldn't you accept?

  - If $\Delta U > 0$, accept the move with probability $e^{-\Delta U / k_B T}$

Intuition here: if you ONLY accepted moves when U decreases, you'll shoot into the nearest local minimum and stay there. But that's not what you want — you want to explore the full energy space. To do that, sometimes you have to "move uphill."

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

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

Note the similarity between this formula and the formula used in the Metropolis criterion — this is not an accident.

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

When you get to very low temperatures, you're no longer good at jumping out of local minima. But this turns out to be a good way to do global minimization.