Molecular dynamics simulation

CS/CME/BioE/Biophys/BMI 279

Oct. 8 and 10, 2024

Ron Dror

Reminder: Assignment 1

- If you haven’t already tested the software setup you plan to use, please do so today

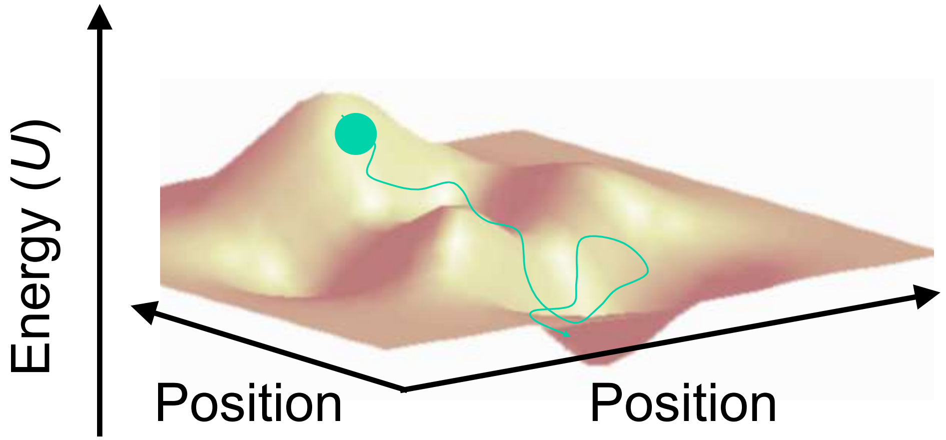
Outline

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

Molecular dynamics: The basic idea

The idea

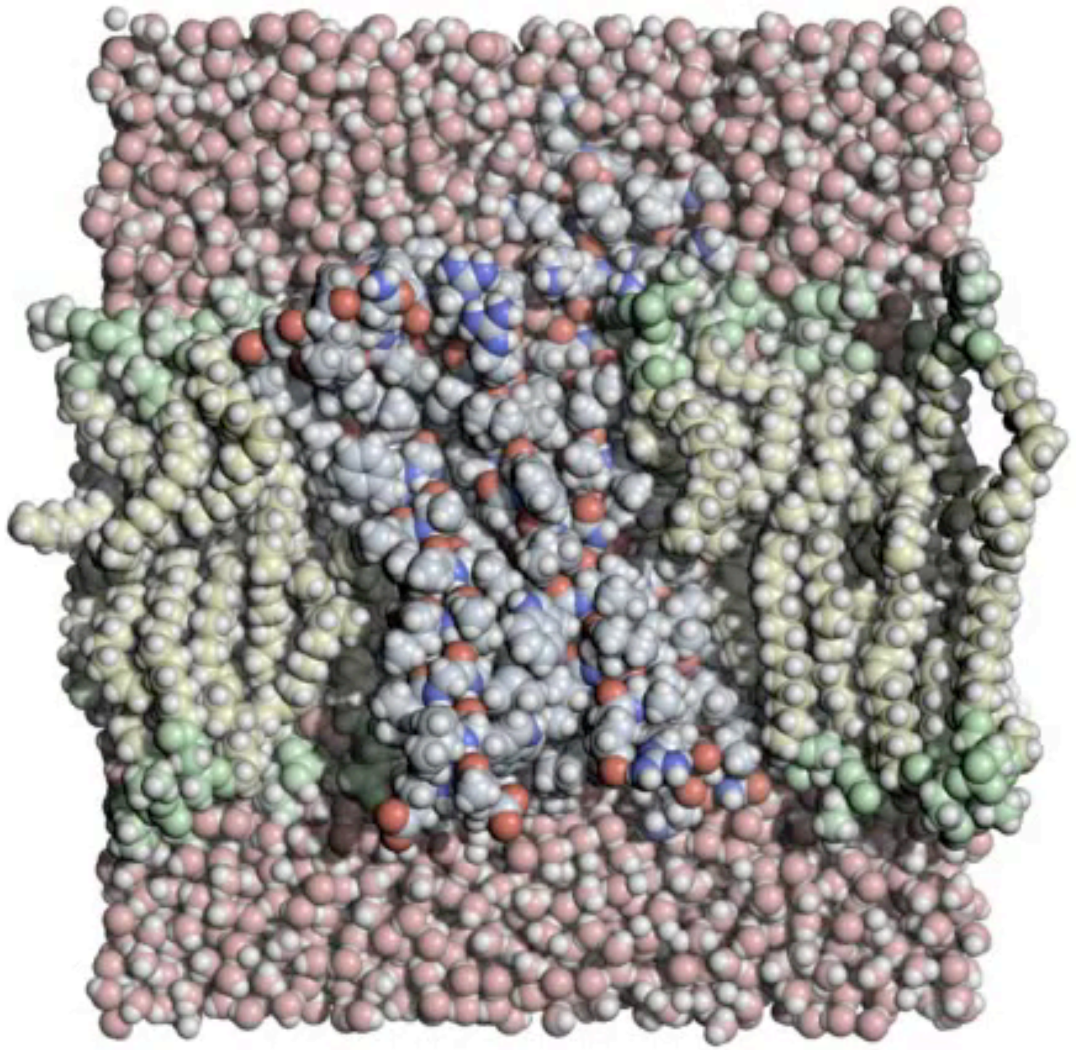
- Mimic what atoms do in real life, assuming a given potential energy function
  - The energy function allows us to calculate the force experienced by any atom, given the positions of the other atoms
  - Newton’s laws tell us how those forces will affect the motions of the atoms



Basic algorithm

- Divide time into discrete time steps, no more than a few femtoseconds each (1 fs = 10^{-15} s)
- 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

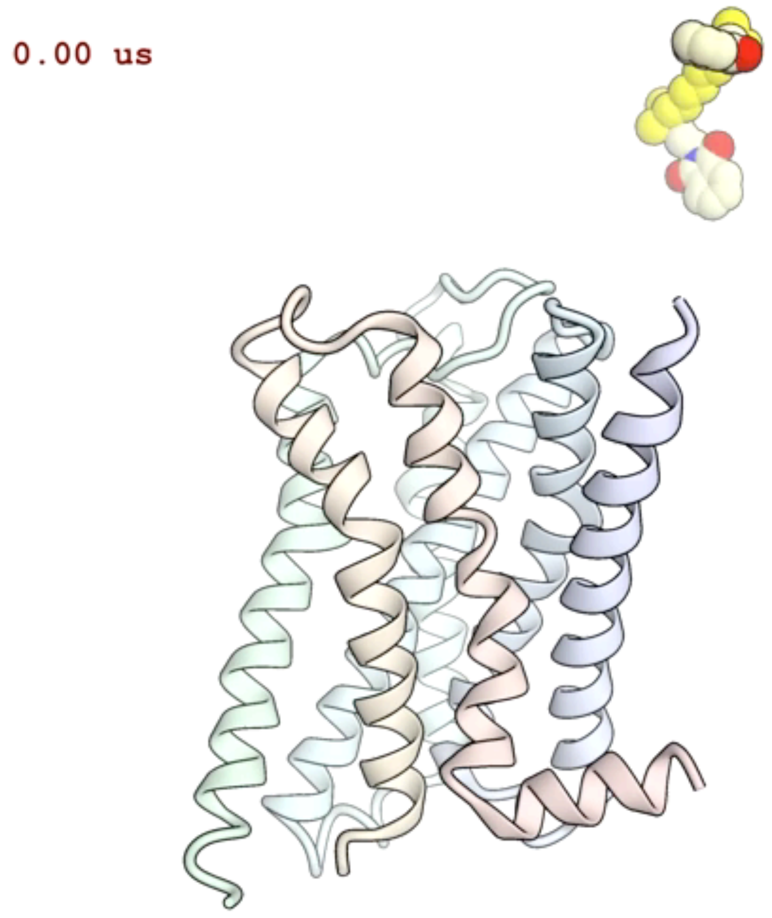


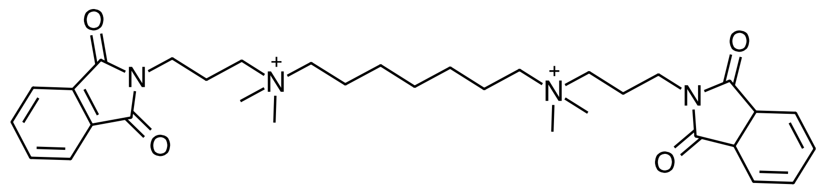
MD as a “computational microscope”

- Molecular dynamics (MD) simulation provides information that is not accessible experimentally
- There is still no experimental technique that allows one to “watch” the motion of every atom in a biomolecule over time

Sample applications

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

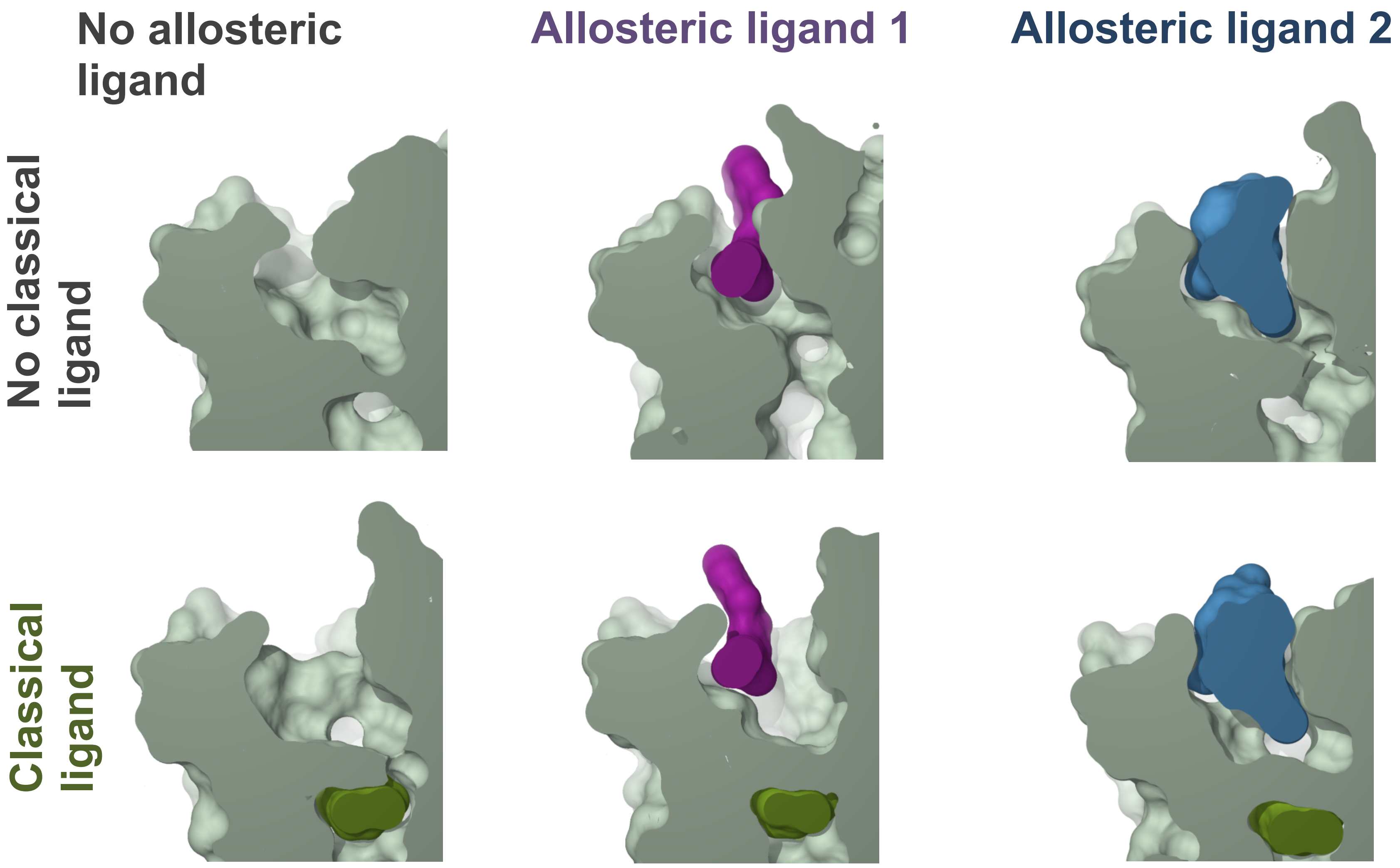




We used simulations to determine where this molecule binds to its receptor, and how it changes the binding strength of molecules that bind elsewhere (in part by changing the protein's structure). We then used that information to modify the molecule such that it has a different effect.

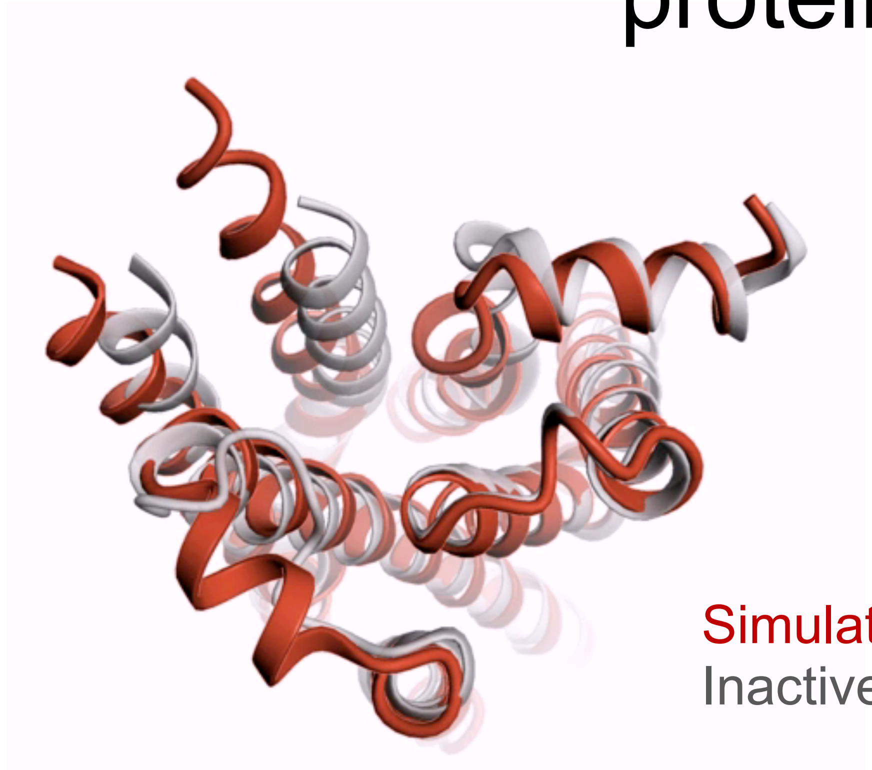
Dror et al., *Nature* 2013

Structural coupling of two binding sites



Dror et al., *Nature* 2013

Determining functional mechanisms of proteins



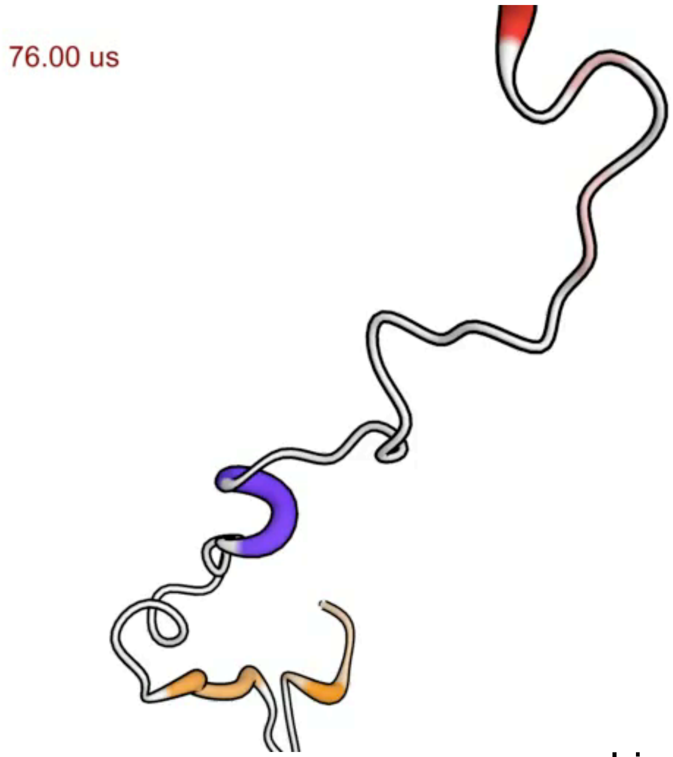
Simulation started from active structure

Inactive structure

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

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

Understanding the *process* of protein folding



Lindorff-Larsen et al., *Science* 2011

- How does the protein get from its unfolded state to its folded state (i.e., how does it “find” its folded structure)?
- Note that if one just wants to know what the folded structure is, MD is generally *not* the best way to predict it (and never has been).

Increasingly, MD is used together with experimental approaches to address more complicated problems

Example:
Suomivuori, Latorraca, …, Dror, *Science*, 2020
"Molecular mechanism of biased signaling in a prototypical G protein–coupled receptor"

Collaboration with Lefkowitz lab (Duke), Kruse lab (Harvard)

Supercomputer Research Redesigns Drugs Without the Side Effects

By Staff report

June 9, 2020

We've all heard the commercials: a drug promises amazing results for treating a disease, and then the remainder of the commercial is filled with a mind-numbingly long list of potential side effects. Side effects plague prescription drugs, sometimes prompting drug approval agencies to reject the drug or making patients wonder if the cure can be worse than the disease. Now,

Source: *HPCWire*

Note: The most challenging part of many MD studies is analyzing the data that the simulations produce

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

How MD simulation works

Basic algorithm

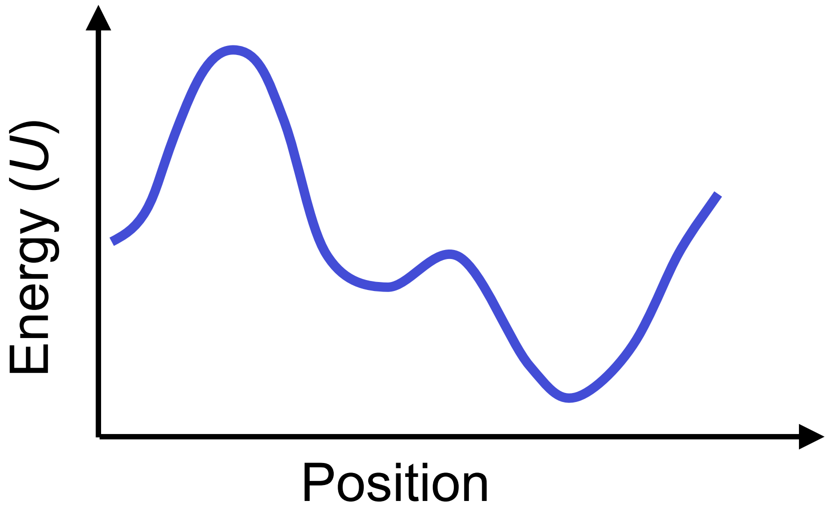
- Divide time into discrete time steps, no more than a few femtoseconds each (1 fs = 10^{-15} s)
- At each time step:
  - Compute the forces acting on each atom, using a molecular mechanics force field
  - Move the atoms a little bit: update position and velocity of each atom using Newton’s laws of motion

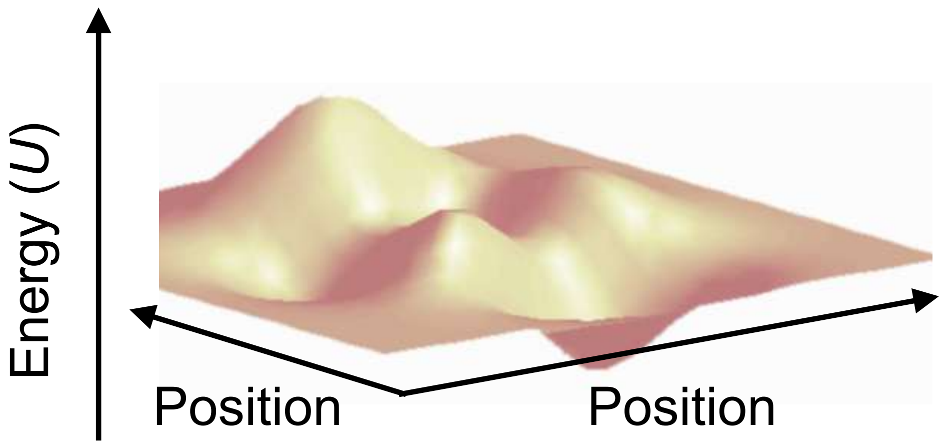
How MD simulation works

Calculating forces on atoms

Energy function

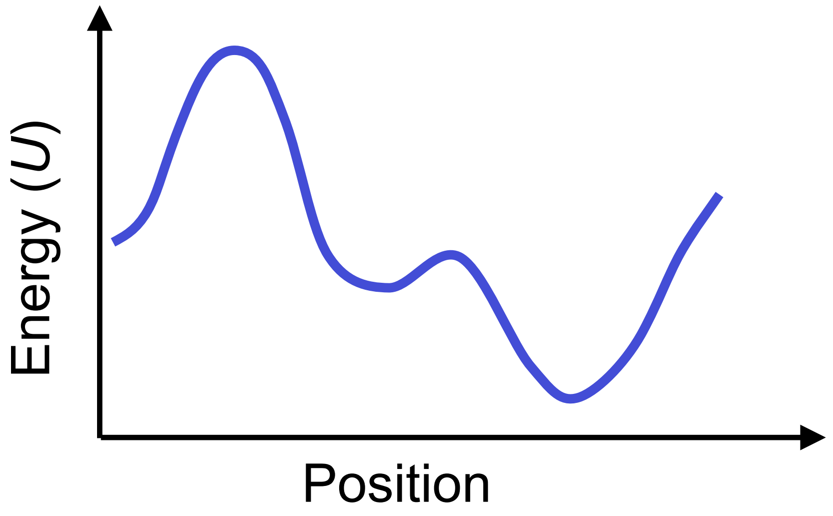
- A potential energy function *U*(***x***) specifies the total potential energy of a system of atoms as a function of all their positions (***x***)
  - In the general case, include not only atoms in the protein but also surrounding atoms (e.g., water)
- The potential energy function *U* is also called a *force field*, because one can use it to compute forces on atoms

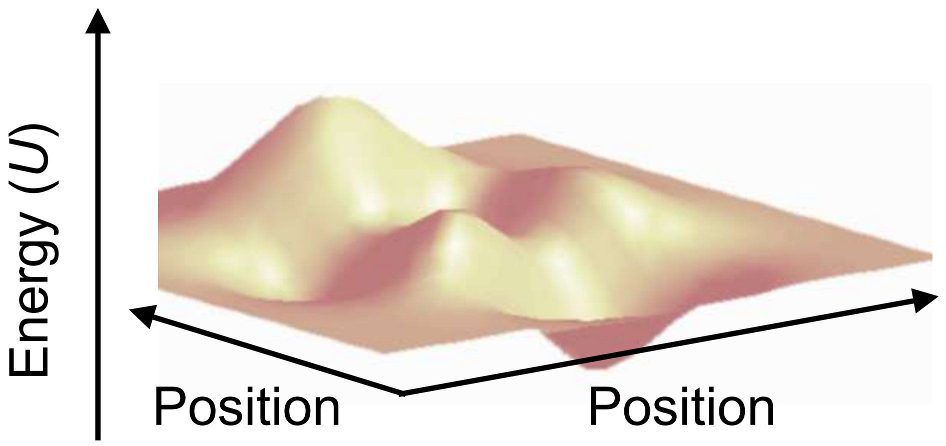




Relationship between energy and force

- Discuss: Given the potential energy function for a molecular system, and the current position of each atom, how can you calculate the force acting on each atom?

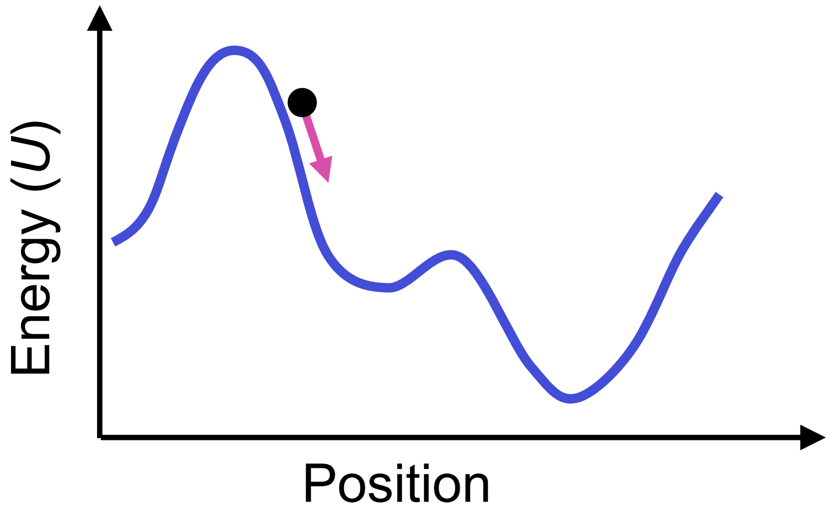


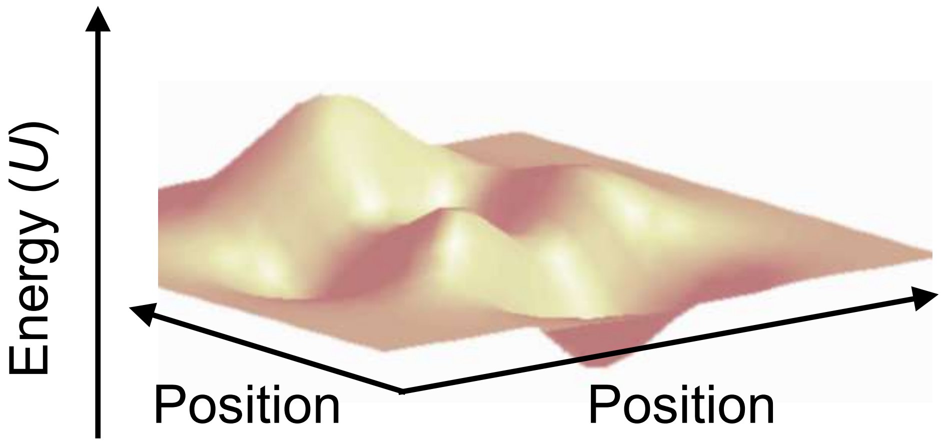


Relationship between energy and force

The intuition

- To determine the force on an atom, consider how quickly the potential energy of the system changes as you move that atom a tiny bit in each direction, without moving any other atom
  - The faster the energy changes, the larger the force
  - The force is in the direction of decreasing energy
  - Calculate how the energy changes when you increase the *x* coordinate of the atom a tiny bit. Then the *y* coordinate, then the *z* coordinate.





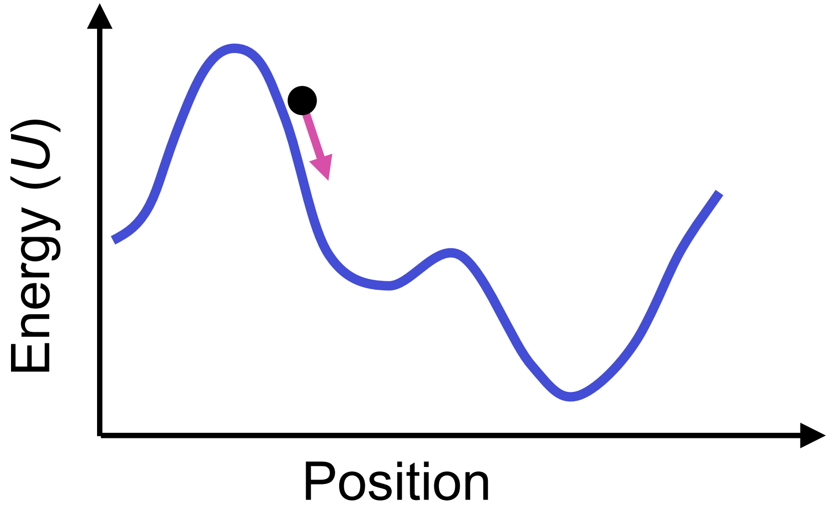
Relationship between energy and force

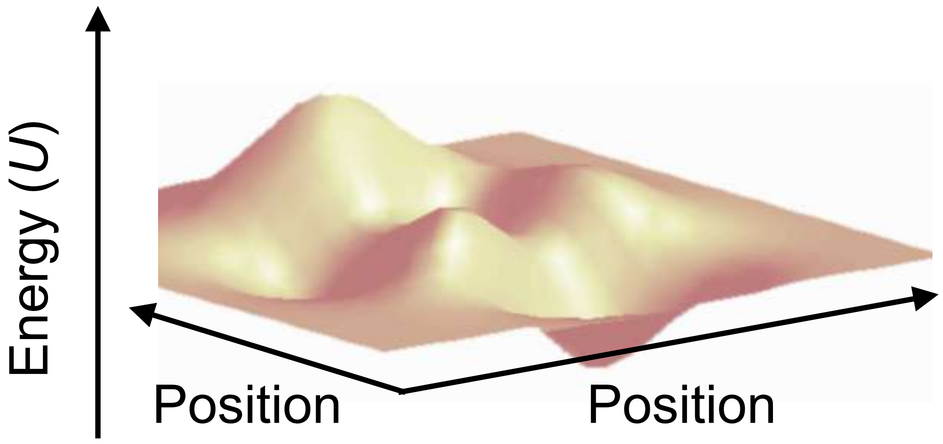
The math

- Force on atom *i* is given by derivatives of *U* with respect to the atom’s coordinates x_i, y_i, and z_i

$$F(\boldsymbol{x}) = -\nabla U(\boldsymbol{x})$$

- At local minima of the energy *U*, all forces are zero





Force vector

- A single vector ***F*** specifies the force acting on every atom in the system
- For a system with *N* atoms, ***F*** is a vector of length 3*N*
  - This vector lists the force on each atom in the *x*-, *y*-, and *z*- directions
- Notation:
  - Force on atom 1 in the *x*-direction: $F_{1,x}$
  - Rate of change of *U* as x_1 increases: $\dfrac{\partial U}{\partial x_1}$

$$\mathbf{x} = \begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ x_2 \\ y_2 \\ z_2 \\ \vdots \\ x_N \\ y_N \\ z_N \end{pmatrix} \qquad \mathbf{F} = \begin{pmatrix} F_{1,x} \\ F_{1,y} \\ F_{1,z} \\ F_{2,x} \\ F_{2,y} \\ F_{2,z} \\ \vdots \\ F_{N,x} \\ F_{N,y} \\ F_{N,z} \end{pmatrix} = - \begin{pmatrix} \frac{\partial U}{\partial x_1} \\ \frac{\partial U}{\partial y_1} \\ \frac{\partial U}{\partial z_1} \\ \frac{\partial U}{\partial x_2} \\ \frac{\partial U}{\partial y_2} \\ \frac{\partial U}{\partial z_2} \\ \vdots \\ \frac{\partial U}{\partial x_N} \\ \frac{\partial U}{\partial y_N} \\ \frac{\partial U}{\partial z_N} \end{pmatrix}$$

Note that *U* depends on which atoms are present in the system and the covalent bonds between them. Two different molecular systems with the same number of atoms will have different potential energy functions.

How MD simulation works

Equations of motion

Equations of motion

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

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

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

Solving the equations of motion

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

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

- 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

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

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

  - where δ_t is the time step (e.g., 2 fs)

Solving the equations of motion

- Straightforward numerical solution:

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

- In practice, it's best to use "time-symmetric" integration methods such as "Leapfrog Verlet"

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

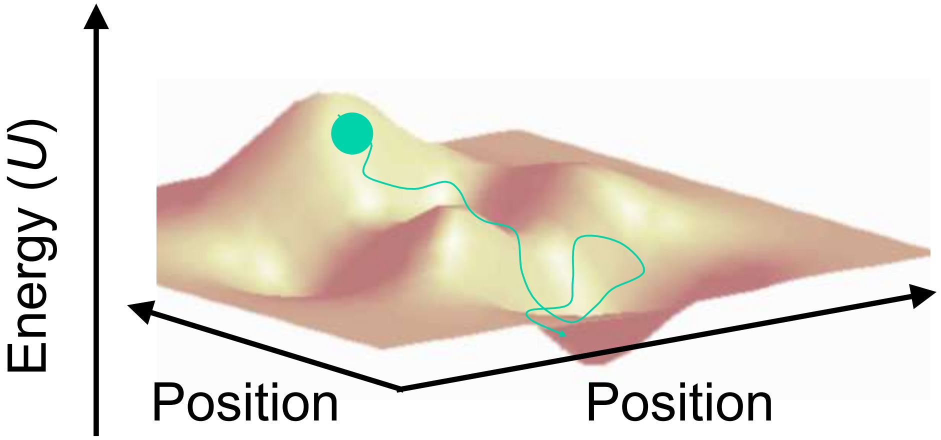
  - This gives better accuracy over long simulations

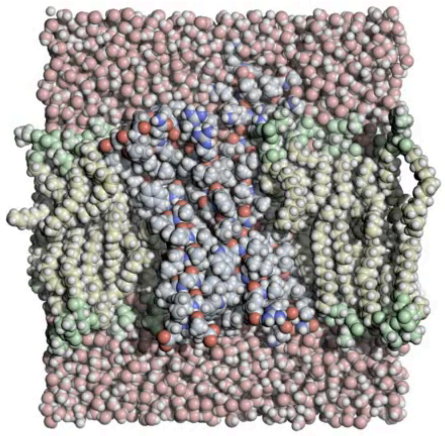
Optional material

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* simply go to an energy minimum and stay there
- *Given enough time, the simulation samples the Boltzmann distribution*
  - That is, the probability of the system adopting a particular arrangement of atoms in simulation matches that specified by the Boltzmann distribution
  - 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





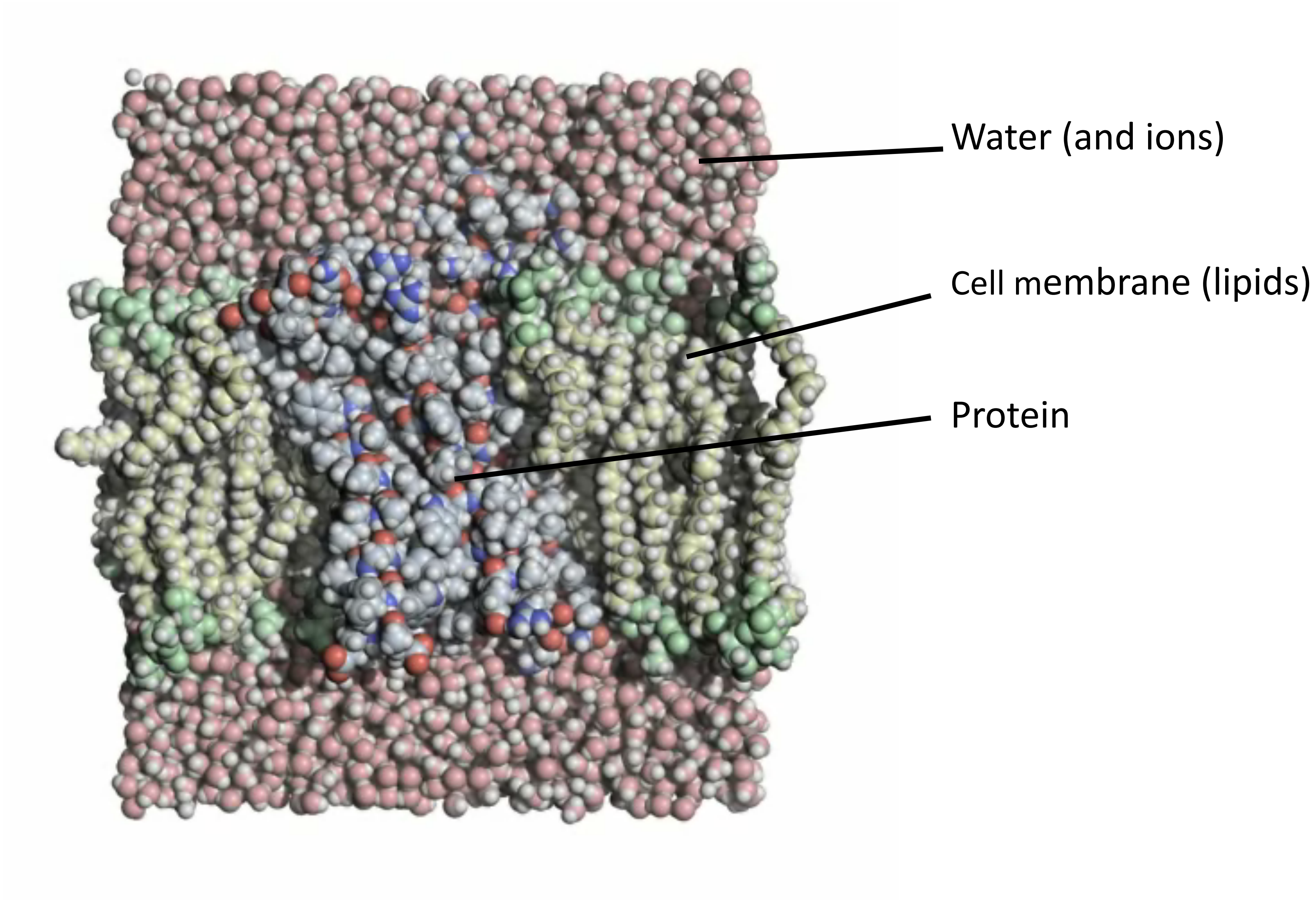
Energy conservation

- 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”)

Water is important

- Ignoring the solvent (the molecules surrounding the biomolecule of interest) leads to major artifacts
  - In a biomolecular system, the solvent is usually mostly water, typically with some salt ions (e.g., sodium, chloride). Membrane proteins are also partly surrounded by lipids.
- Two options for taking solvent into account
  - Explicitly represent solvent molecules
    - More computationally expensive 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 from the right side, like in Pac-Man)
  - Implicit solvent
    - Mathematical model to approximate average effects of solvent
    - Less accurate but faster

Explicit solvent



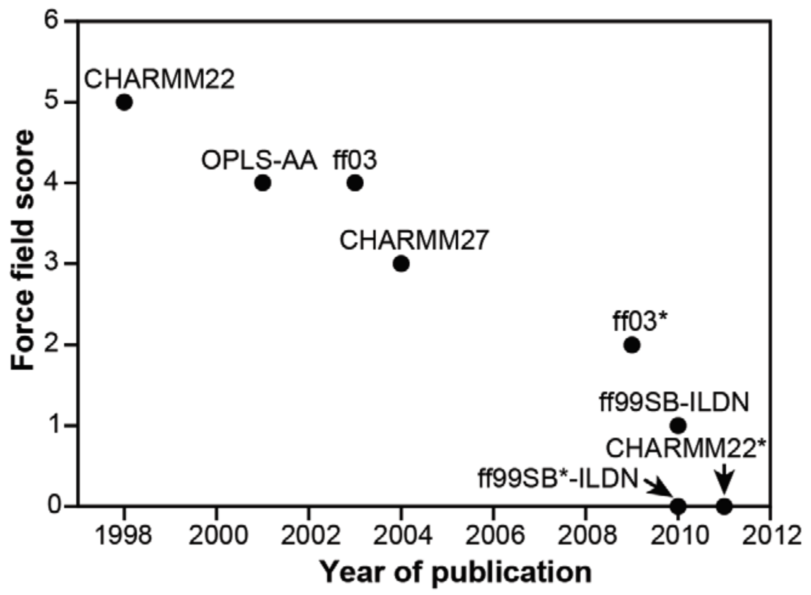
Limitations of MD simulations

Timescales

- Simulations require short time steps for numerical stability
  - 1 time step ≈ 2 fs (2×10^{-15} s)
- 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 about 10 years ago, simulations of 1 microsecond were rare
- Advances in computer power have enabled simulations of several microseconds on individual GPUs (graphics processing units), 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
- They have improved substantially over time, but various limitations remain



Here force fields with lower scores are better, as assessed by agreement between simulations and several types of experimental data. Even the force fields with scores of zero are imperfect, however!

Lindorff-Larsen et al., *PLOS One*, 2012

- In practice, one needs some experience to know what to trust in a simulation (as is the case for many experimental methods)

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)
  - Acidic or basic amino acid residues can lose or gain a hydrogen (i.e., a proton)
- Various more advanced techniques do allow simulations to capture breaking or formation of at least some covalent bonds

Software packages and force fields

(Next two slides are optional material,
but they'll be useful if you want to use 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”)
  - PyMOL is an alternative

Force fields for molecular dynamics

- Most MD simulations today use force fields from one of three families:
  - 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

MD simulations of biomolecules consume about one-third of U.S. supercomputer resources (for which usage data is available)

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, usually yes. These forces fall off quickly with distance.
    - For electrostatics, usually 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

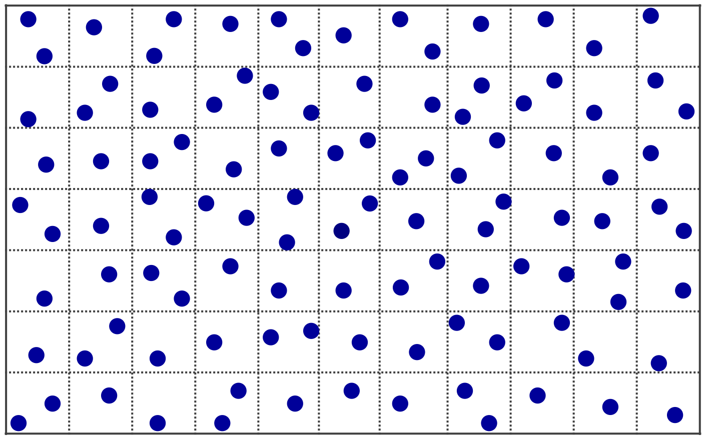
I want you to understand why simulations are computationally expensive and slow, and to have a sense of the types of things people try to speed them up. You are not responsible for the details of these speed-up methods.

How can one speed up MD simulations?

- Reduce the amount of computation per time step
  - Faster algorithms
  - Example: fast approximate methods to compute distant 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., fix the lengths of certain bonds).
- 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—but not all—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.

GPU

Specialized chip

Monte Carlo simulation

Monte Carlo simulation

- 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” or “reject” (i.e., make or not 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 (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
  - If $\Delta U > 0$, accept the move with probability $e^{-\Delta U / k_B T}$

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

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

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

A completely different approach

- A recent research direction is using machine learning to sample from the Boltzmann distribution without moving between nearby arrangements of atoms
  - Currently, one needs to have already sampled the low-energy conformational states in order to train the machine learning models, but ongoing research may solve that.
- Such techniques cannot capture dynamics (i.e., produce a movie), but at least in certain cases, they can sample the Boltzmann distribution very efficiently

Prominent example: Noé et al., *Science* 2019

"Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning"