# Review

CS/BioE/Biophys/BMI/CME 279

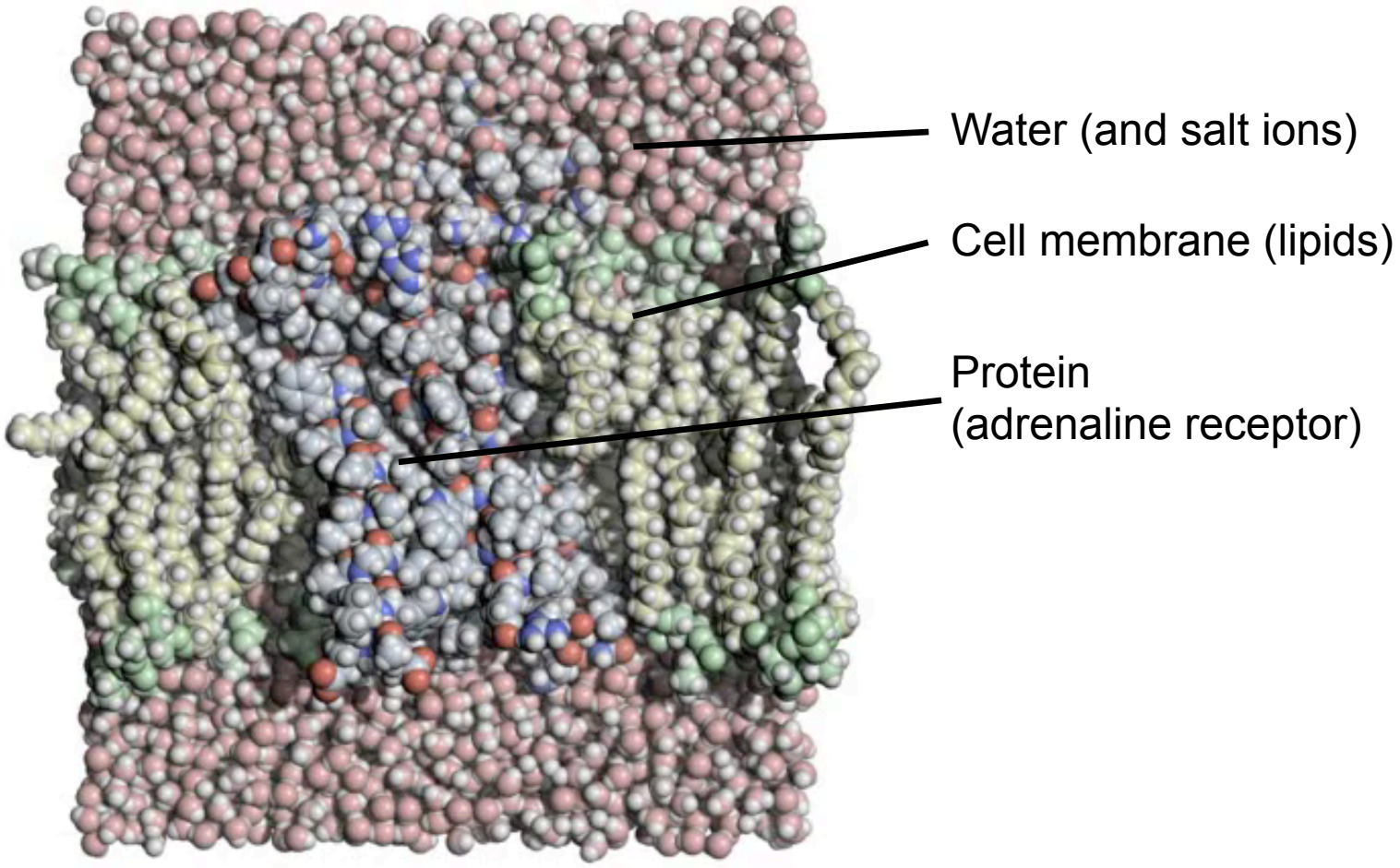
Dec. 2, 2021

Ron Dror

# Outline

- Atomic-level modeling of biological macromolecules
  - Energy functions and their relationship to molecular conformation
  - Molecular dynamics simulation
  - Protein structure prediction
  - Protein design
  - Ligand docking
- Coarser-level modeling and imaging-based methods
  - Fourier transforms and convolution
  - Image analysis
  - Microscopy
  - X-ray crystallography
  - Cryoelectron microscopy
  - Diffusion and cellular-level simulation
- Recurring themes

# Atomic-level modeling of biological macromolecules



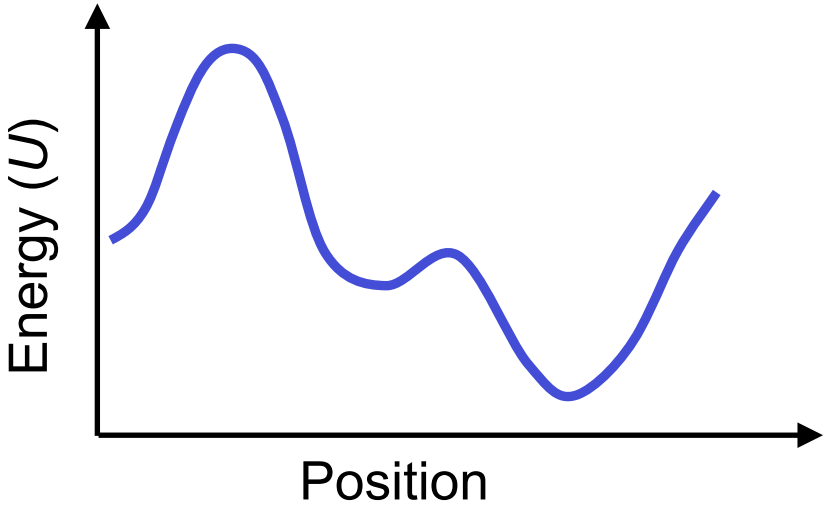

- Biological macromolecules (e.g, proteins) are constantly jiggling around, as are the molecules that surround them.
- Each protein can in fact assume many structures, but they tend to be similar to one another. Usually we talk about the “average” structure, which is (roughly) what’s determined experimentally and deposited in the PDB.
- The surrounding molecules play a key role in determining protein structure.

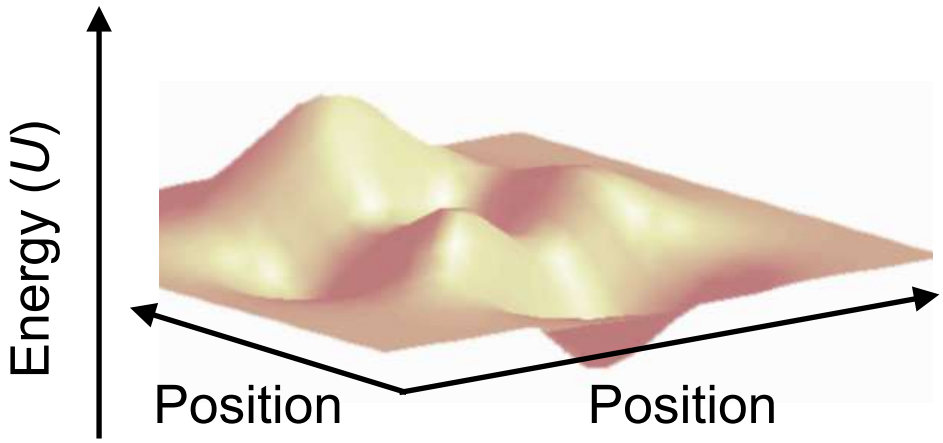
Atomic-level modeling

# Energy functions and their relationship to molecular conformation

# Potential energy functions

- A potential energy function $U(\boldsymbol{x})$ specifies the total potential energy of a system of atoms as a function of all their positions ($\boldsymbol{x}$)
  - For a system with $n$ atoms, $\boldsymbol{x}$ is a vector of length $3n$ ($x$, $y$, and $z$ coordinates for every atom)
  - In the general case, include not only atoms in the protein but also surrounding atoms (e.g., water)
- The force on each atom can be computed by taking derivatives of the potential energy function



# Example of a potential energy function: molecular mechanics force field

$$U = \sum_{\text{bonds}} k_b \left(b - b_0\right)^2$$

Bond lengths ("Stretch")

$$+ \sum_{\text{angles}} k_\theta \left(\theta - \theta_0\right)^2$$

Bond angles ("Bend")

$$+ \sum_{\text{torsions}} \sum_{n} k_{\phi,n} \left[1 + \cos\left(n\phi - \phi_n\right)\right]$$

Torsional/dihedral angles

**Bonded terms**

$$+ \sum_i \sum_{j>i} \frac{q_i q_j}{r_{ij}}$$

Electrostatic

$$+ \sum_i \sum_{j>i} \frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^{6}}$$

Van der Waals

**Non-bonded terms**

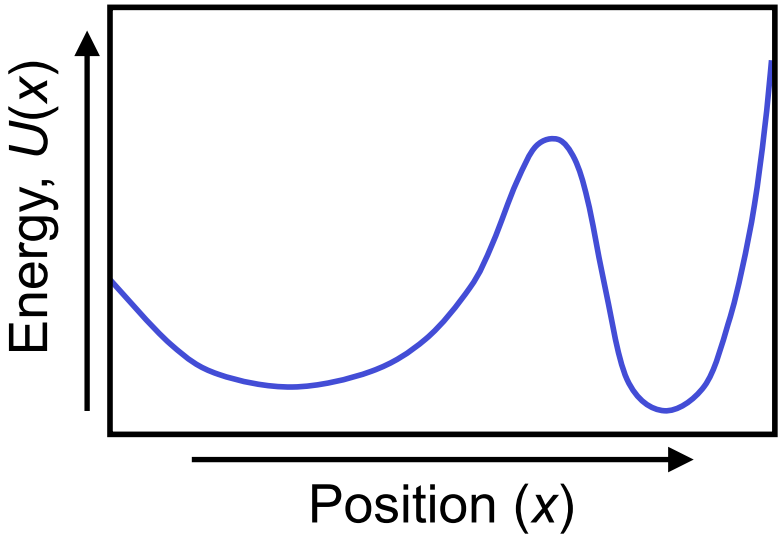
Above is the form of a typical molecular mechanics force field (as used in molecular dynamics simulations, for example). The terms capture different types of inter-atomic interactions.

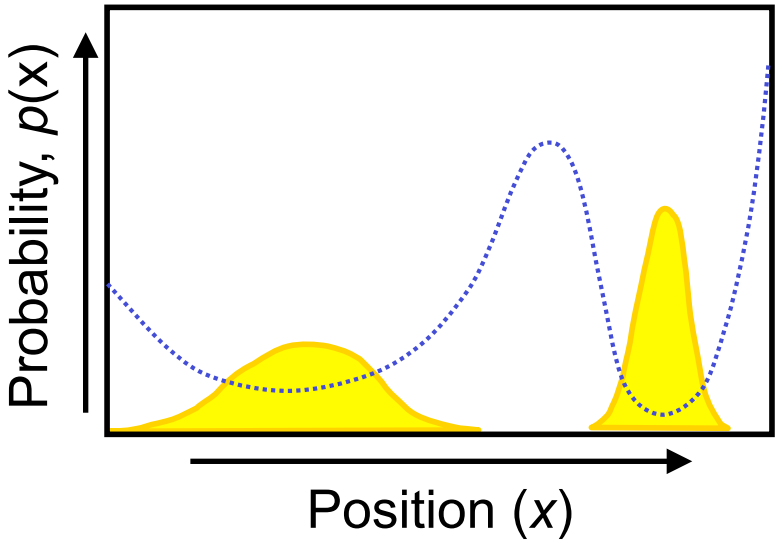
# The Boltzmann Distribution

- The Boltzmann distribution relates the potential energy of a particular arrangement of atoms to the probability of observing that arrangement of atoms (at equilibrium):

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

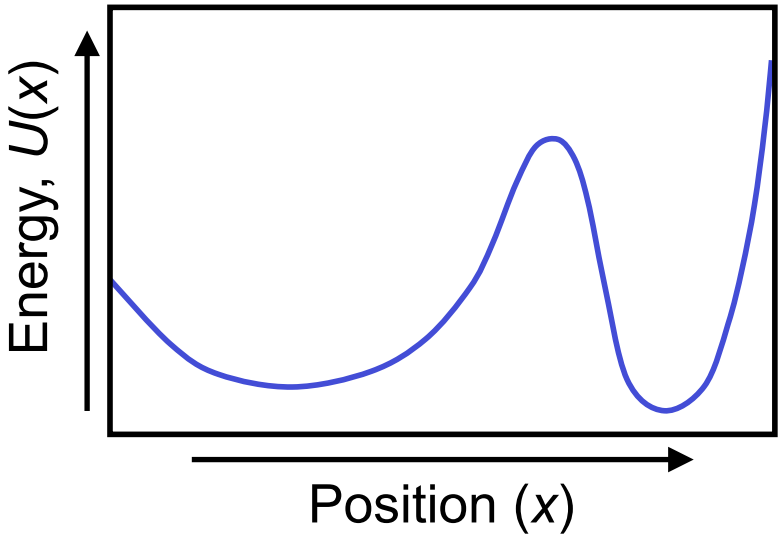
where $T$ is temperature and $k_B$ is the Boltzmann constant

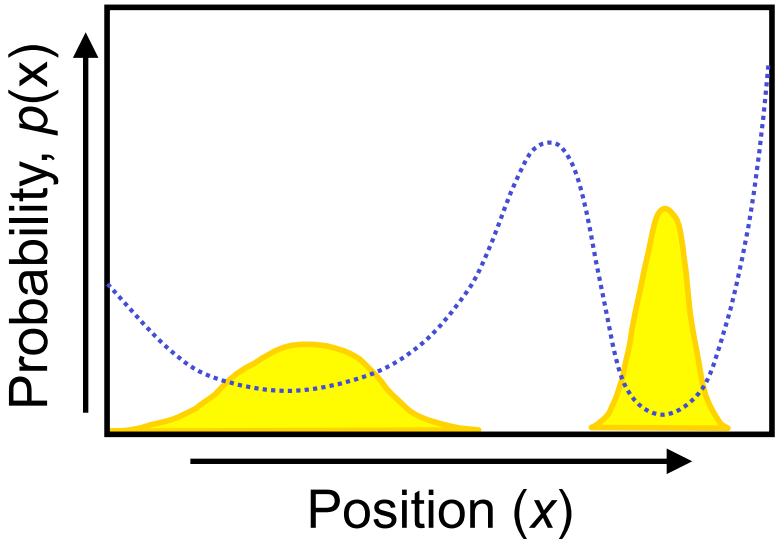


# Macrostates

- We typically care most about the probability that protein atoms will be in some *approximate* arrangement, with *any* arrangement of surrounding atoms
- We thus care about the probability of sets of atomic arrangements, called *macrostates*
  - These correspond roughly to wells of the potential energy function
  - To calculate probability of a well, we sum the probabilities of all the specific atomic arrangements it contains



# Free energies

- The free energy $G_A$ of a macrostate $A$ satisfies:

$$P(A) = \exp\left(-G_A / k_B T\right)$$

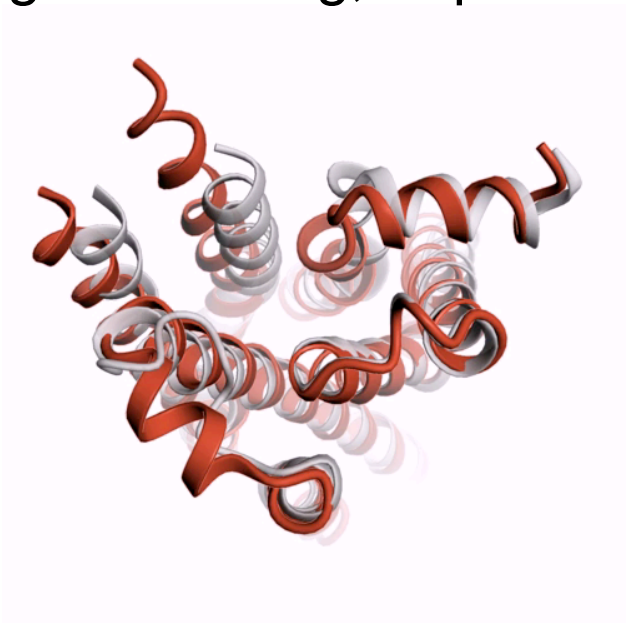
- Clarifications:
  - Strictly speaking, free energies are defined for macrostates
  - In protein structure prediction, protein design, and ligand docking, it's useful to define a "free energy function" that approximates a free energy for some *neighborhood* of each arrangement of protein atoms
    - In applications like structure prediction, we wish to minimize free energy, not potential energy
  - The term "energy function" is used for both potential energy and free energy functions

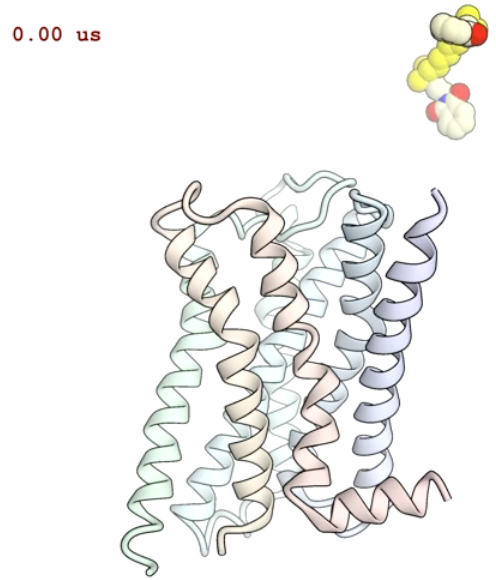
Atomic-level modeling

# Molecular dynamics simulation

# Molecular dynamics (MD) simulation

- An MD simulation predicts how atoms move around based on physical models of their interactions
- Of the atomic-level modeling techniques we covered, this is closest to the physics; it attempts to predict the real dynamics of the system
- It can thus be used to capture functionally important *processes*, including structural changes in proteins, protein-ligand binding, or protein folding





# Basic MD algorithm

- Step through time (very short steps)
- At each time step, calculate force acting on every atom using a molecular mechanics force field
- Then update atom positions and velocities using Newton's second law

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

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

Note: this is inherently an approximation, because we're using classical physics rather than quantum mechanics. (Quantum mechanical calculations are used to parameterize force fields, however.)

# MD is computationally intensive

- Because:
  - One needs to take millions to trillions of timesteps to get to timescales on which events of interest take place
  - Computing the forces at each time step involves substantial computation
    - Particularly for the non-bonded force terms, which act between every pair of atoms

# Sampling

- Given enough time, an MD simulation will sample the full Boltzmann distribution of the system
  - This means that if one takes a snapshot from the simulation after a long period of time, the probability of the atoms being in a particular arrangement is given by the Boltzmann distribution
- One can also sample the Boltzmann distribution in other ways, including Monte Carlo sampling with the Metropolis criterion
  - Metropolis Monte Carlo: generate moves at random. Accept any move that decreases the energy. Accept moves that increase the energy by $\Delta U$ with probability

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

Atomic-level modeling

# Protein structure prediction

# Protein structure prediction

- The goal: given the amino acid sequence of a protein, predict its “average” three-dimensional structure
  - That is, find a structure whose “neighborhood” has minimal free energy
- In theory, one could do this by MD simulation, replicating the process by which the protein folds, but that isn’t practical
- Practical methods for protein structure prediction take advantage of existing data on protein structure (and sequence)
- Similar principles apply for predicting structure of other macromolecules (e.g,. RNA)

# Two classical approaches to protein structure prediction

- Template-based modeling (homology modeling)
  - Used when one can identify one or more likely homologs of known structure (usually the case)
- *Ab initio* structure prediction
  - Used when one cannot identify any likely homolog of known structure
  - Even *ab initio* approaches usually take advantage of available structural data, but in more subtle ways

# Template-based structure prediction: basic workflow

- User provides a *query* sequence with unknown structure
- Search the PDB for proteins with similar sequence and known structure. Pick the best match (the *template*).
- Build a model based on that template
  - One can also build a model based on multiple templates—for example, by using different templates for different parts of the protein.

# Principle underlying template-based modeling: structure is more conserved than sequence

- Proteins with similar sequences tend to be *homologs*, meaning that they evolved from a common ancestor
- The fold of the protein (i.e., its overall structure) tends to be conserved during evolution
- This tendency is very strong. Even proteins with 15% sequence identity usually have similar structures.
  - During evolution, sequence changes more quickly than structure

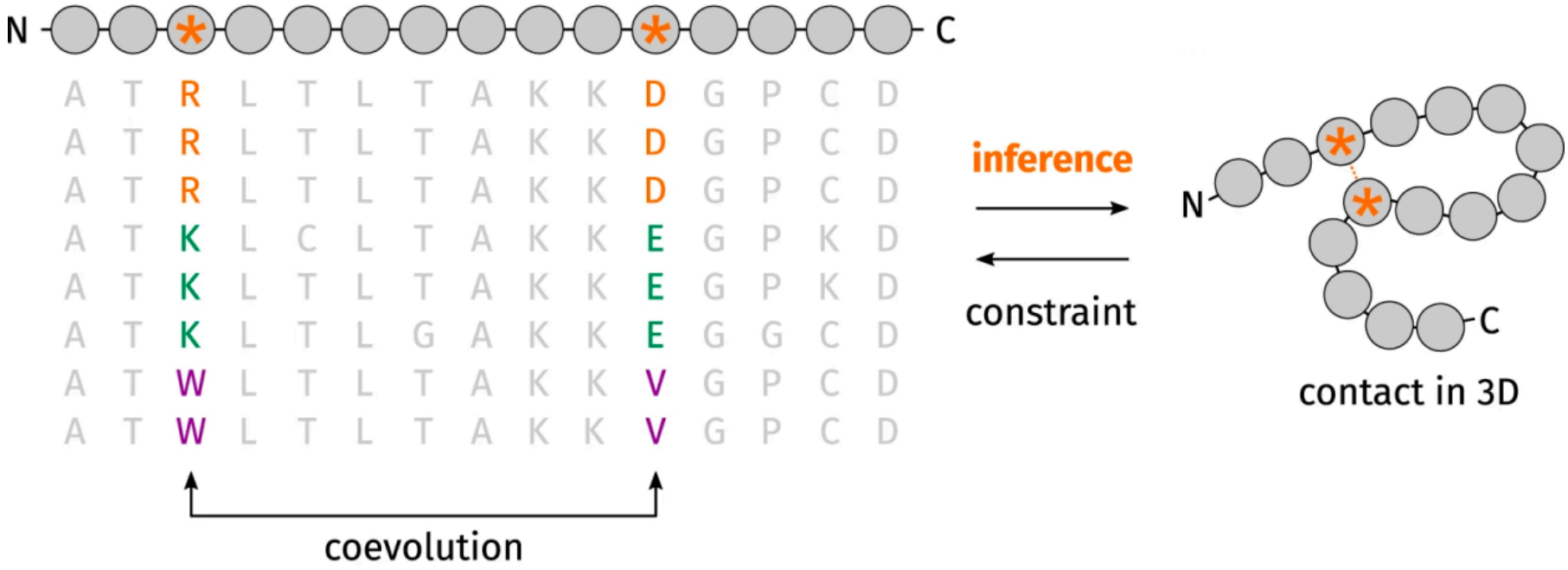
# *Ab initio* structure prediction (as exemplified by Rosetta)

- Search for structure that minimizes an energy function
  - This energy function is knowledge-based (informed, in particular, by statistics of the PDB), and it approximates a free energy function
- Use a knowledge-based search strategy
  - Rosetta uses a Monte Carlo search method involving "fragment assembly," in which it tries replacing structures of small fragments of the protein with fragment structures found in the PDB

# Co-evolution methods

- If no *structure* of a related protein is available, one may still be able to find many *sequences* of related proteins
- One can use these sequences to infer which amino acid residues are in physical contact with one another
- Structure prediction methods based on such information ("co-evolution" or "multiple sequence alignments") have recently developed rapidly

# Amino acids in direct physical contact tend to covary or “coevolve” across related proteins




From: https://www.blopig.com/blog/2021/07/alphafold-2-is-here-whats-behind-the-structure-prediction-miracle/

# Machine learning methods

- A variety of machine learning methods are used to develop energy functions and to infer structure from multiple sequence alignment data
- In the past year, protein structure prediction has improved dramatically thanks to machine learning methods that combine information on structural templates, related sequences, and favorable local arrangements of atoms

Atomic-level modeling

# Protein design

# Protein design

- Goal: given a desired approximate three-dimensional structure (or structural characteristics), find an amino acid sequence that will fold to that structure
- In principle, one could do this by searching over sequences and doing *ab initio* structure prediction for each possible sequence, but that's not practical

# Simplifying the problem

- Instead of predicting the structure for each sequence considered, focus on the desired structure and find the sequence that minimizes its energy
  - Energy is generally measured by a knowledge-based free energy function
- Consider a discrete set of rotamers for each amino acid side chain
  - Minimize simultaneously over identities and rotamers of amino acids
- Assume the backbone is fixed
  - Or give it a bit of “wiggle room”

# Heuristic but effective

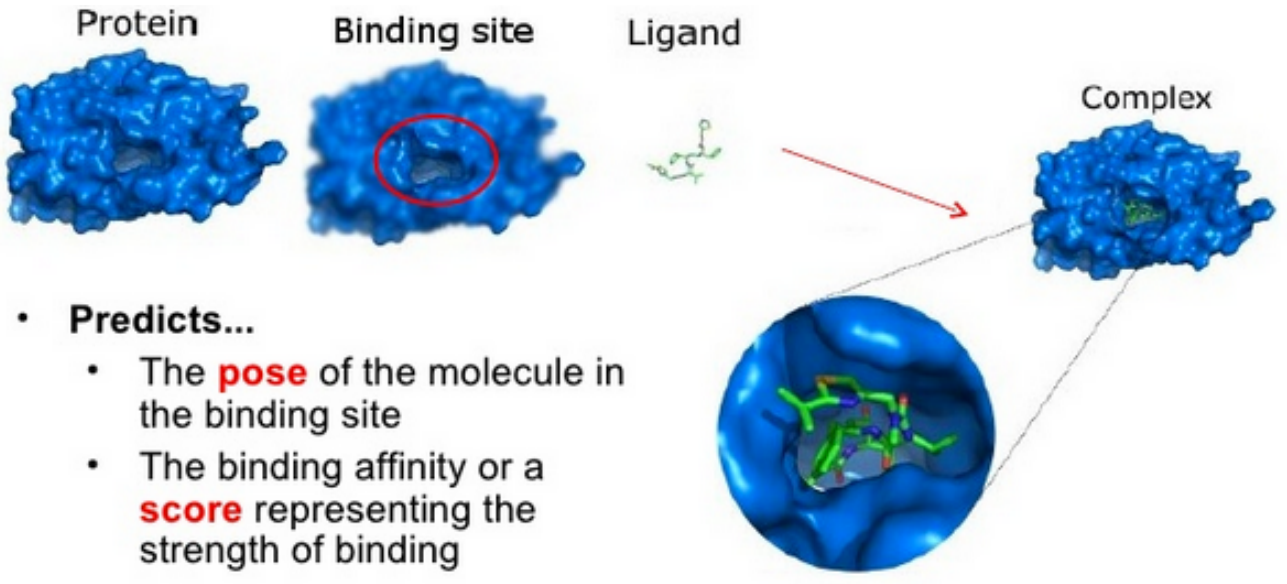
- These simplifications mean that practical protein design methodologies are heuristic, but they've proven surprisingly effective
- The minimization problem is also usually solved with heuristic methods (e.g., Metropolis Monte Carlo)

Atomic-level modeling

# Ligand docking

# Ligand docking

- Goals:
  - Given a ligand known to bind a particular protein, determine its binding *pose* (i.e., location, orientation, and internal conformation of the bound ligand)
  - Determine how tightly a ligand binds a given protein





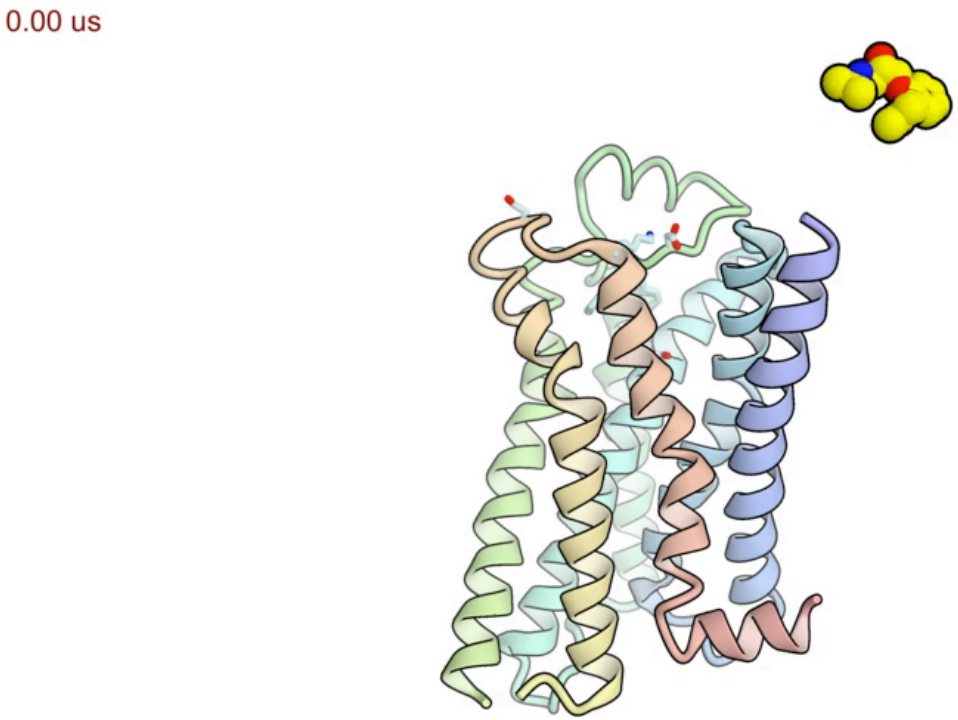

http://www.slideshare.net/baoilleach/proteinligand-docking-13581869

# Binding affinity

- *Binding affinity* quantifies the binding strength of a ligand to a protein (or other target)
- Conceptual definition: if we mix the protein and the ligand (with no other ligands around), what fraction of the time will the protein have a ligand bound?
- Affinity can be expressed as the difference $\Delta G$ in free energy of the bound state and the unbound state, or as the concentration of unbound ligand molecules at which half the protein molecules will have a ligand bound

# Docking is heuristic

- In principle, we could estimate binding affinity by measuring the fraction of time the ligand is bound in an MD simulation, but this isn't practical



# Docking methodology

- Ligand docking is a fast, heuristic approach with two key components
  - A *scoring function* that very roughly approximates the binding affinity of a ligand to a protein given a binding pose
  - A *search method* that searches for the best-scoring binding pose for a given ligand
- Most ligand docking methods assume that
  - The protein is rigid
  - The approximate binding site is known
    - That is, one is looking for ligands that will bind to a particular site on the target

# Coarser-level modeling and imaging-based methods

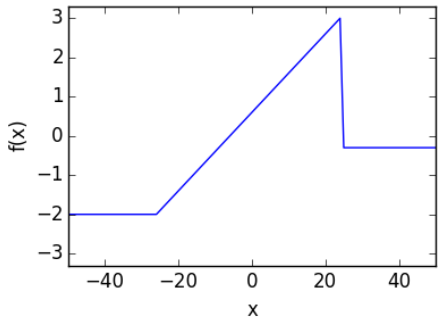
# Coarser-level modeling and imaging-based methods

## Fourier transforms and convolution

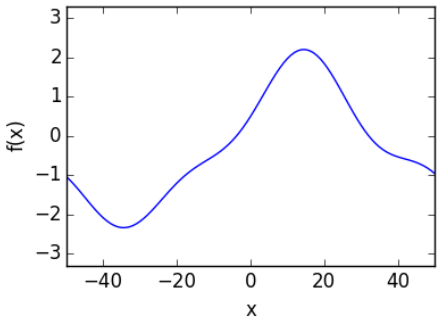
# Writing functions as sums of sinusoids

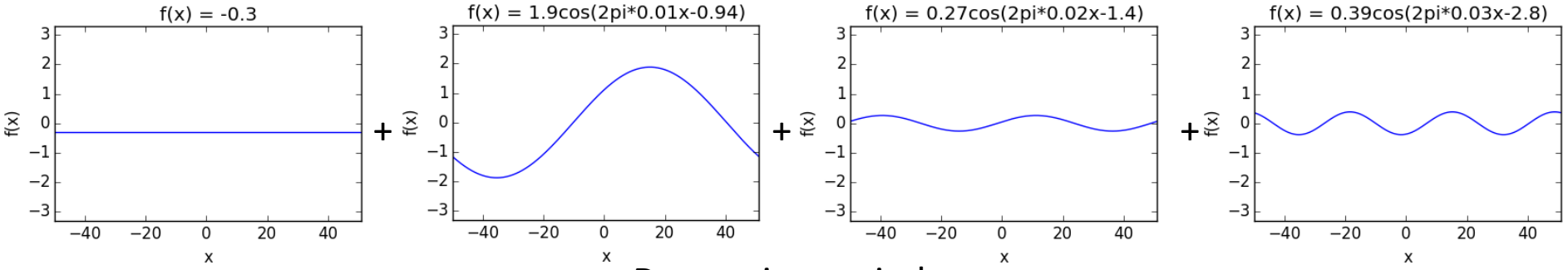
- Given a function defined on an interval of length *L*, we can write it as a sum of sinusoids with the following frequencies/periods:
  - Frequencies: 0, 1/*L*, 2/*L*, 3/*L*, ….
  - Periods: constant term, *L, L/2, L/3, …*

Original function




Sum of sinusoids below



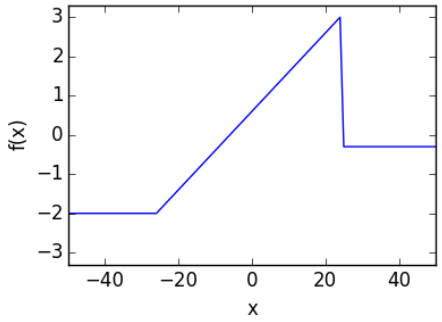

Decreasing period

Increasing frequency

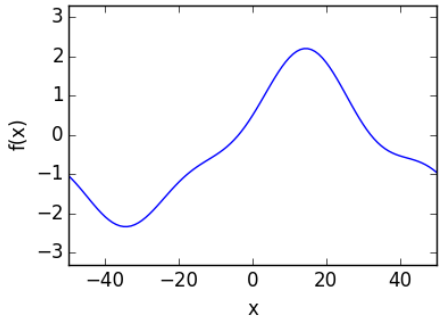
# Writing functions as sums of sinusoids

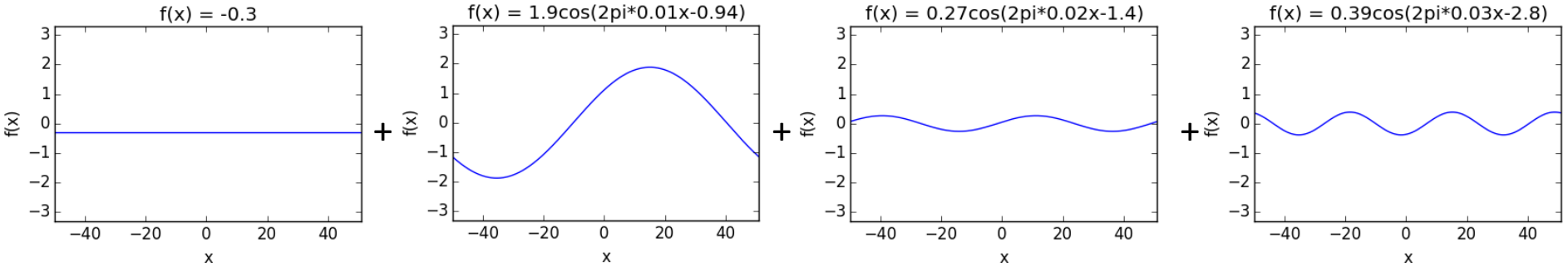
- Each of these sinusoidal terms has a magnitude (scale factor) and a phase (shift).

Original function




Sum of sinusoids below




Magnitude: -0.3
Phase: 0 (arbitrary)

Magnitude: 1.9
Phase: -.94

Magnitude: 0.27
Phase: -1.4

Magnitude: 0.39
Phase: -2.8

# The Fourier Transform: Expressing a function as a set of sinusoidal term coefficients

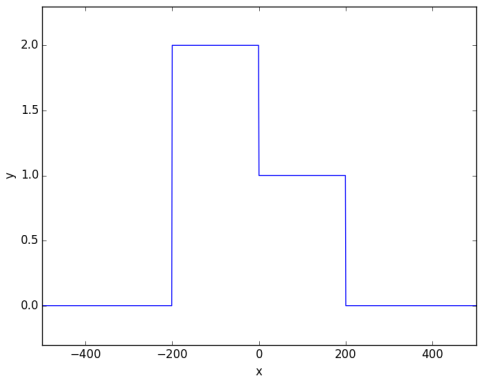
- We can thus express the original function as a series of magnitude and phase coefficients
  - We can express each pair of magnitude and phase coefficients as a complex number
- The Fourier transform maps the function to this set of complex numbers, providing an alternative representation of the function.
- This also works for functions of 2 or 3 variables (e.g., images)
- Fourier transforms can be computed efficiently using the Fast Fourier Transform (FFT) algorithm

| Constant term (frequency 0) | Sinusoid 1 (period $L$, frequency $1/L$) | Sinusoid 2 (period $L/2$, frequency $2/L$) | Sinusoid 3 (period $L/3$, frequency $3/L$) |
|---|---|---|---|
| Magnitude: -0.3 | Magnitude: 1.9 | Magnitude: 0.27 | Magnitude: 0.39 |
| Phase: 0 (arbitrary) | Phase: -.94 | Phase: -1.4 | Phase: -2.8 |

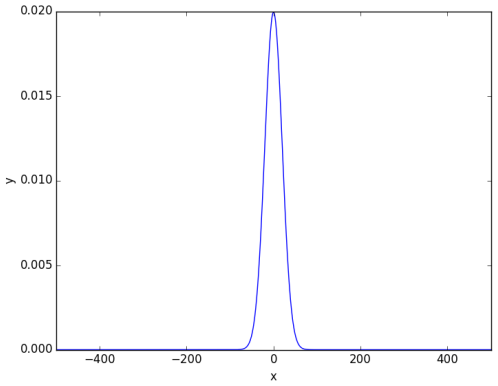
# Convolution

- Convolution is a weighted moving average
  - To convolve one function with another, we compute a weighted moving average of one function using the other function to specify the weights

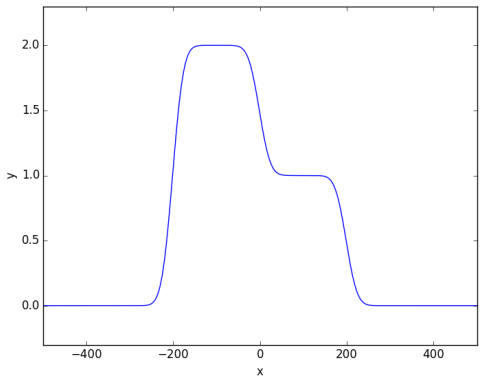
f




g




f convolved with g



# Convolution = multiplication in frequency domain

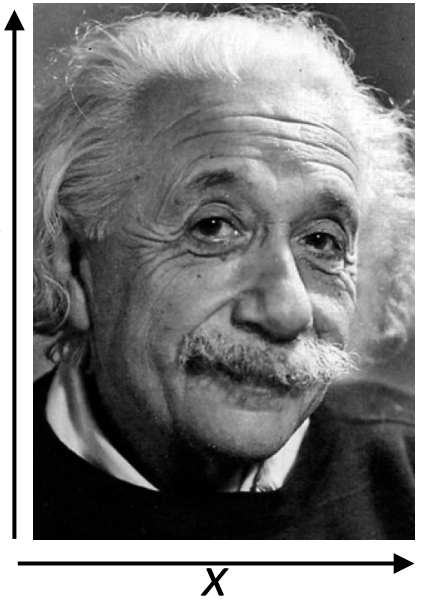
- One can compute the convolution of two functions by taking the Fourier transform of both functions, multiplying the resulting coefficients at each frequency, and then performing an inverse Fourier transform
- Why is this important?
  - It provides an efficient way to perform large convolutions (thanks to the FFT)
  - It allows us to interpret convolutions in terms of what they do to different frequency components (e.g., high-pass and low-pass filters)

# Coarser-level modeling and imaging-based methods

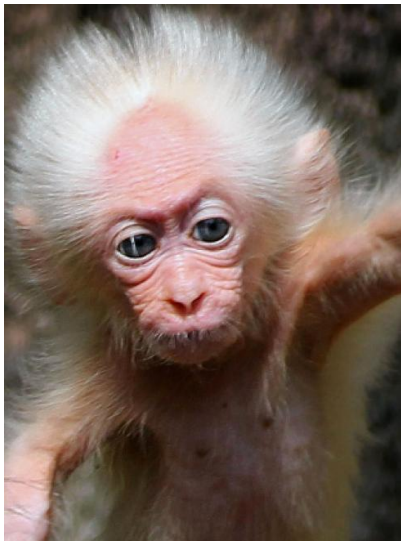
## Image analysis

# Representations of an image

- We can think of a grayscale image as:
  - A two-dimensional array of brightness values
  - A function of two variables (*x* and *y*), which returns the brightness of the pixel at position (*x*, *y*)




- A color image can be treated as three separate images (red, green, blue), or as a function that returns three values (red, green, blue) for each (*x*, *y*) pair

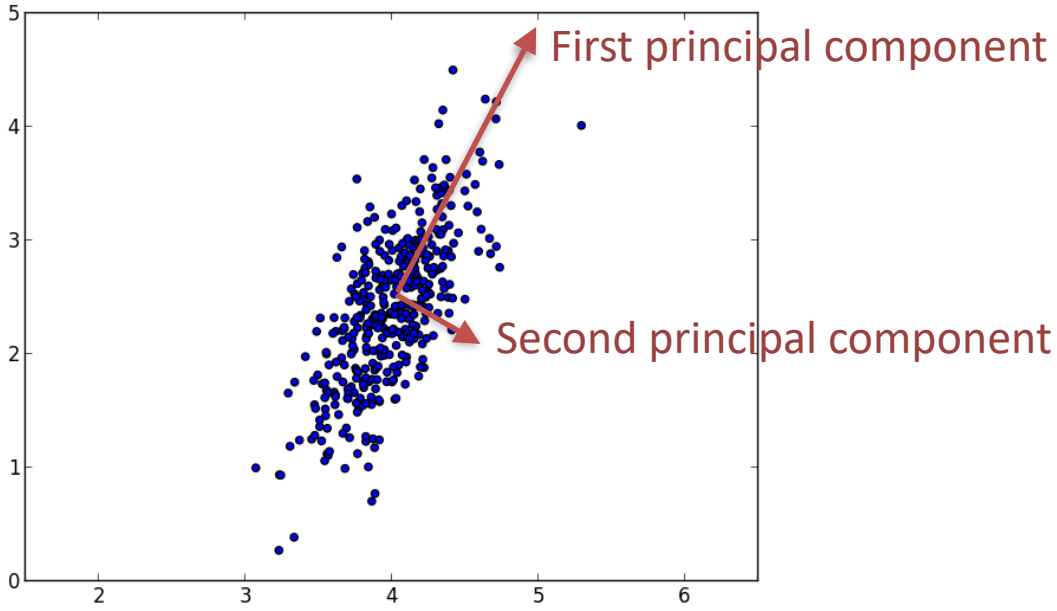


# Reducing image noise

- We can reduce image noise using various filters (e.g., mean, median, Gaussian)
  - These all rely on the fact that nearby pixels in an image tend to be similar
- The mean and Gaussian filters (and many others) are convolutions, and can thus be expressed as multiplications in the frequency domain
  - These denoising filters are *low-pass filters*. They reduce magnitudes of high-frequency coefficients while preserving low-frequency coefficients
  - These filters work because real images have mostly low-frequency content, while noise tends to have a lot of high-frequency content

# Principal component analysis (PCA)

- Basic idea: given a set of points in a multi-dimensional space, we wish to find the linear subspace (line, plane, etc.) that best fits those points.




- PCA provides a way to represent high-dimensional data sets (such as images) approximately in just a few dimensions
- It is thus useful in summarization and classification of images

# Coarser-level modeling and imaging-based methods

## Microscopy

# The diffraction limit

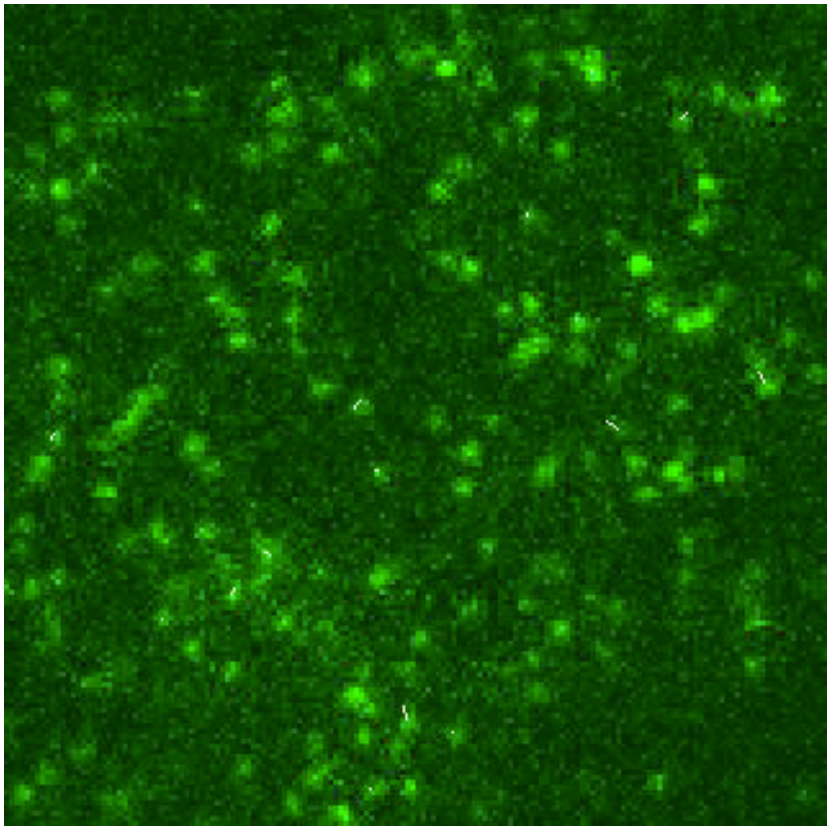
- The image observed under a microscope is always slightly blurred due to fundamental limitations on how well a lens can focus light
  - The observed image is a low-pass filtered version of the ideal image
- This leads to a limit on resolution known as the diffraction limit
  - The achievable resolution scales with wavelength of the radiation used (i.e., a shorter wavelength leads to a smaller minimum distance between resolvable points)
  - X-rays have shorter wavelength than visible light. Electrons have *much* shorter wavelengths.

# Fluorescence microscopy: basic idea

- Suppose we want to know where a particular type of protein is located in the cell, or how these proteins move around
- We can't do this by simply looking through a microscope, because:
  - We (usually) don't have sufficient resolution
  - The protein of interest doesn't look different from those around it
- Solution: Make the molecules of interest glow by attaching fluorophores (fluorescent molecules)
  - When you shine light of a particular wavelength on a fluorophore, it emits light of a different wavelength

# Single-molecule tracking

- If the density of fluorescent molecules is sufficiently low, we can track individual molecules
  - Doing this well is a challenging computational problem



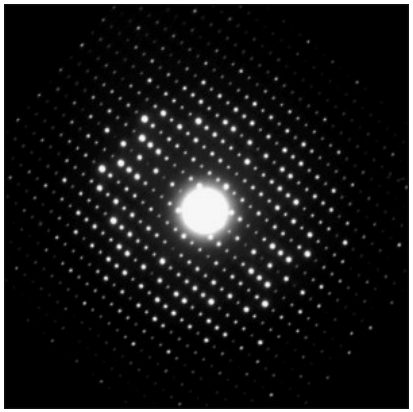
Data: Bettina van Lengerich, Natalia Jura
Tracking and movie: Robin Jia

# Coarser-level modeling and imaging-based methods

## X-ray crystallography

# The basic idea

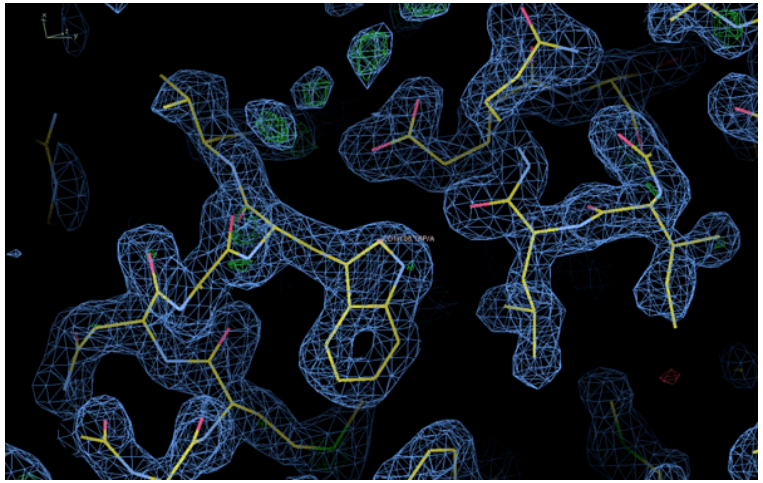
- Get the molecule whose structure you want to determine to form a crystal
- Shine an intense beam of x-rays through the crystal, giving rise to a “diffraction pattern” (a pattern of spots of varying brightnesses)
  - Shine x-rays through the crystal at multiple angles to capture the full 3D diffraction pattern
- From that pattern, infer the 3D structure of the molecule



http://lacasadeloscristales.trianatech.com/wp-content/uploads/2014/09/image005-300x300.jpg

# How does the diffraction pattern relate to the molecular structure?

- The diffraction pattern is the Fourier transform of the electron density!
  - But only the magnitude of each Fourier coefficient is measured, not the phase
  - The lack of phase information makes solving the structure (i.e., going from the diffraction pattern to a set of 3D atomic coordinates) challenging



Contour map of electron density



http://www.lynceantech.com/images/electron_density_map.png

# Solving for molecular structure

- Step 1: *Initial phasing*
  - Come up with an approximate solution for the structure (and thus an approximate set of phases), often using a predicted structure as a model
- Step 2: *Phase refinement*
  - Search for perturbations that improve the fit to the experimental data (the diffraction pattern), often using simulated annealing
  - Restrain the search to “realistic” molecular structures, usually using a molecular mechanics force field

# Coarser-level modeling and imaging-based methods

## Cryo-electron microscopy

# The basic idea

- We want the structure of a “particle”: a molecule (e.g., protein) or a well-defined complex composed of many molecules (e.g., a ribosome)
- We spread identical particles out on a film, and image them using an electron microscope
- The images are two-dimensional (2D), each representing a *projection* of the 3D shape (density) of a particle.  Each particle is positioned at a different, unknown angle.
- Given enough 2D images of particles, we can computationally reconstruct the 3D shape of a particle

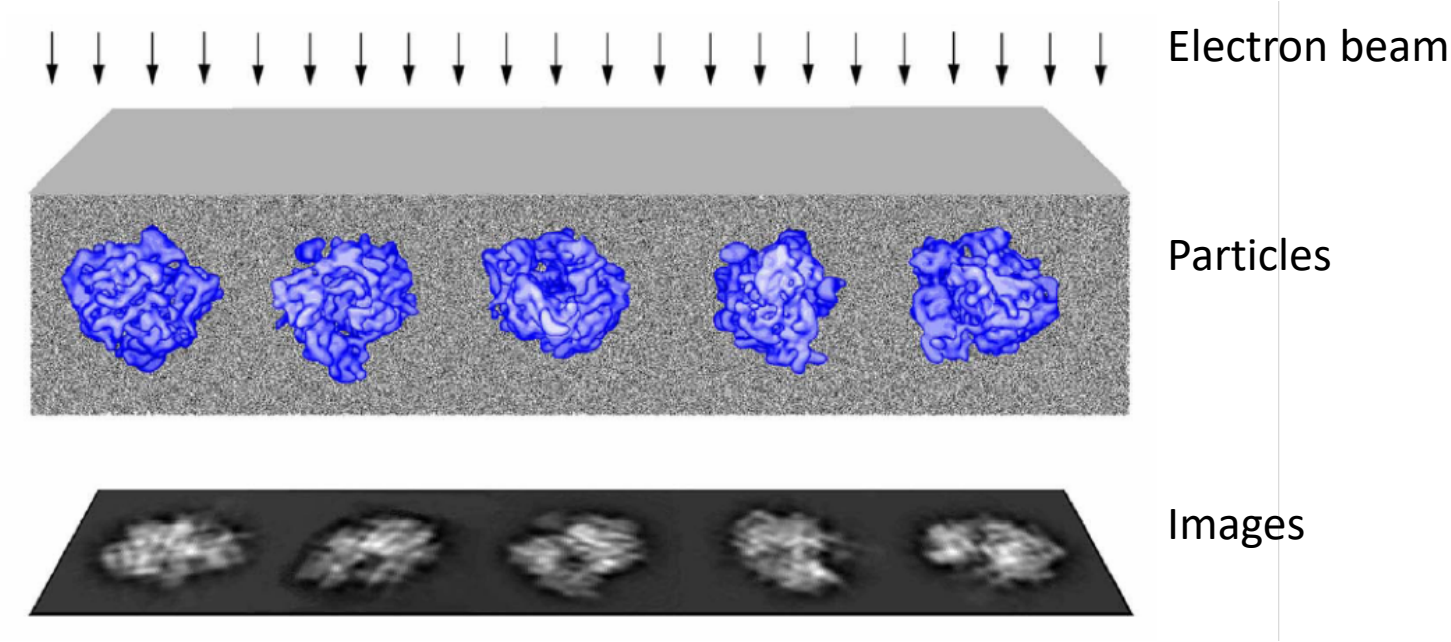





Image from Joachim Frank
http://biomachina.org/courses/structures/091.pdf

# Determining the 3D structure: key steps

- **2D image analysis**: First, go from raw image data to high-resolution 2D projections
  - Image preprocessing
  - Particle picking
  - Image clustering and class averaging: reduce image noise by identifying images with similar view angles, aligning them, and averaging them

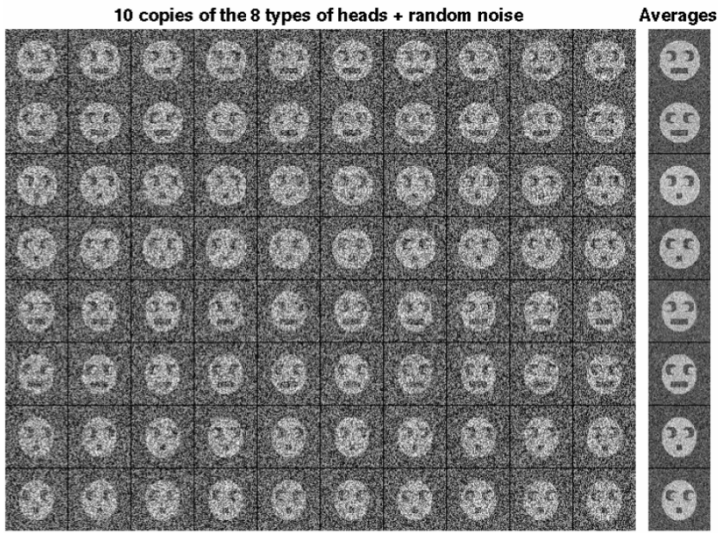
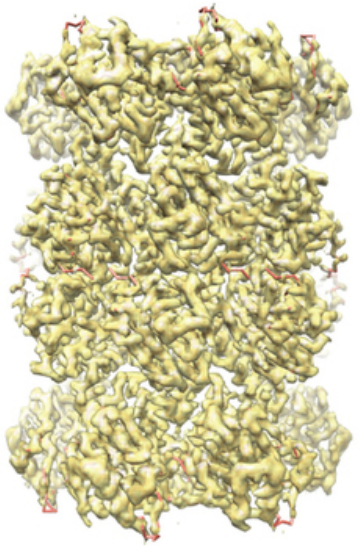



Image from Joachim Frank
http://biomachina.org/courses/structures/091.pdf

# Determining the 3D structure: key steps

- **3D reconstruction**: Then use these high-resolution projections to build a 3D model
  - **Reconstruction with known view angles** is fairly straightforward. Standard algorithm is *filtered back-projection.*
  - **Structure refinement with unknown view angles** (the problem at hand) is harder. Iterate between improving estimates of view angles given a 3D model and building a better 3D model given those view angles.
  - Several methods to "guess" an initial 3D model



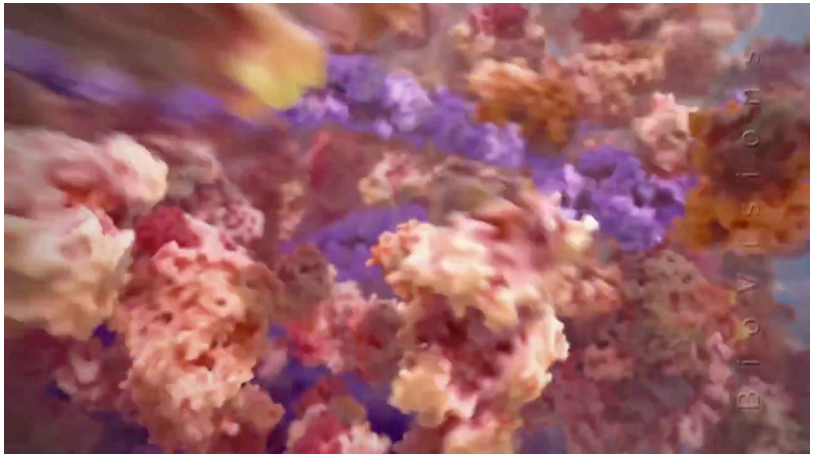
Reconstructed 3D density map

Li et al., Nature Methods 10:584 (2013)

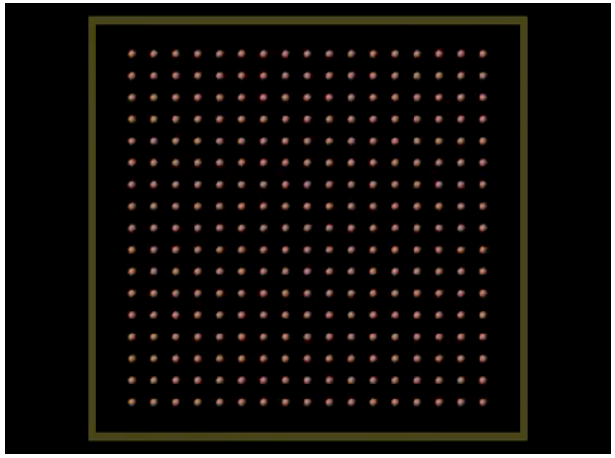
Coarser-level modeling and imaging-based methods

Diffusion and cellular-level simulation

# How do molecules move within a cell?




From *Inner Life of the Cell* | *Protein Packing*, XVIVO and Biovisions @ Harvard



https://www.youtube.com/watch?v=1jYabtziQZo

- Molecules jiggle about because other molecules keep bumping into them
- Individual molecules thus follow a random walk
- Diffusion = many random walks by many molecules
  - Substance goes from region of high concentration to region of lower concentration
  - Aggregate behavior is deterministic

# Particle-based perspective

- In the basic case of random, unconfined, undirected motion:
  - Individual molecules follow a random walk, leading to Brownian motion
  - Mean squared displacement is proportional to time
  - The proportionality constant is specified by the diffusion coefficient
    - Faster-moving molecules have larger diffusion coefficients

# Continuum view of diffusion

- If enough molecules are involved, we can predict their aggregate behavior
- The rate at which the concentration of a molecule changes with time is given by the diffusion equation
  - This rate is determined by the second derivative of concentration with respect to each spatial coordinate

$$\frac{\partial c}{\partial t} = D\left(\frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial y^2} + \frac{\partial^2 c}{\partial z^2}\right)$$

*D* is the diffusion coefficient

# Reaction-diffusion simulation

- A common way to model how molecules move within the cell involves *reaction-diffusion simulation*
- Basic rules:
  - Molecules move around by diffusion
  - When two molecules come close together, they have some probability of reacting to combine or modify one another
- Two implementation strategies:
  - Particle-based models
  - Continuum models (based on the diffusion equation)

# Recurring Themes

# Physics-based vs. data-driven approaches

- Physics-based approaches: modeling based on first-principles physics
- Data-driven approaches: machine learning based on experimental data
- Examples:
  - Physics-based vs. knowledge-based energy functions
  - Molecular dynamics vs. protein structure prediction
- Most methods fall somewhere on the continuum between these two extremes
  - Examples: ligand docking or solving x-ray crystal structures

# Energy functions

- Energy functions (representing either potential energy or free energy) play a key role in many of the techniques we've covered, including:
  - Molecular dynamics
  - Protein structure prediction
  - Protein design
  - Ligand docking
  - X-ray crystallography

# Computation is required not only for prediction of structure and dynamics but also for structural interpretation of experimental data

- Computational prediction
  - Protein structure prediction and design
  - Molecular dynamics simulation
  - Ligand docking
  - Reaction-diffusion simulations
- Structural interpretation of experimental data
  - X-ray crystallography
  - Cryo-electron microscopy
  - Image analysis for fluorescence microscopy data

# Recurring math concepts

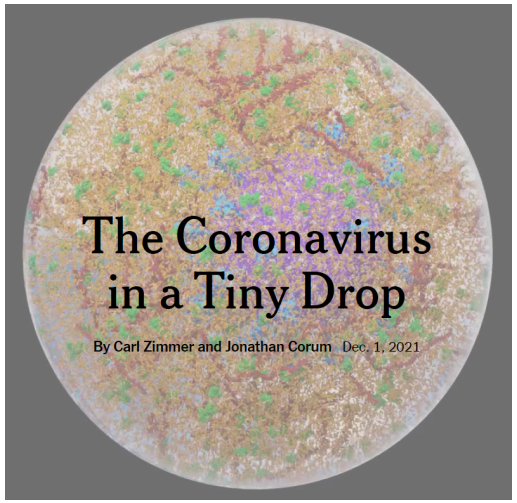
- **Fourier transforms** and **convolution** play important roles in:
  - Image analysis
  - X-ray crystallography
  - Cryo-electron microscopy
  - Some methods for docking, molecular dynamics simulation, and machine learning (especially deep learning)
- Another recurring math concept: **Monte Carlo methods**

# Similarities and differences in methods employed at different spatial scales

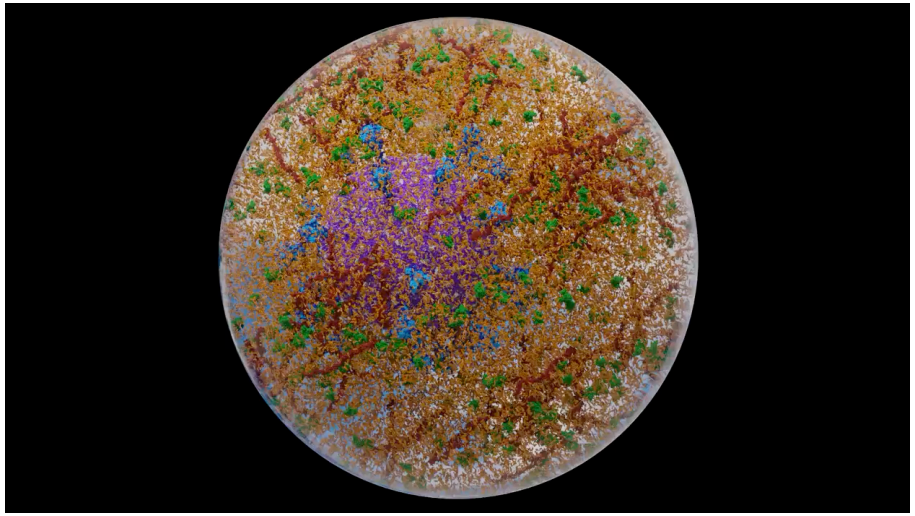
- Atomic-level modeling of single proteins vs. coarser-level modeling of complexes and cells
- Experimental methods: x-ray crystallography vs. single-particle electron microscopy vs. fluorescence microscopy
- Simulation methods: molecular dynamics vs. reaction-diffusion simulations

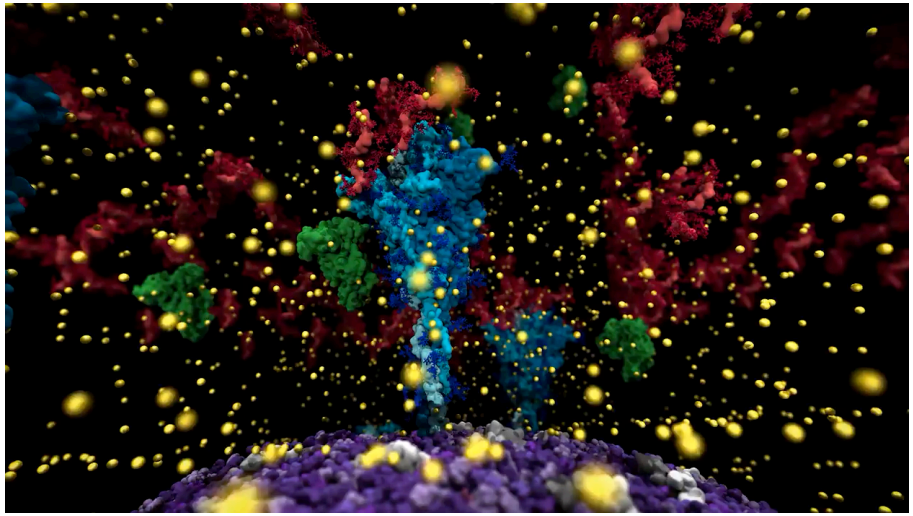
Although we've mostly used proteins as examples, the concepts and methods we've covered apply to other biomolecules as well (RNA, DNA, carbohydrates, small molecules, etc.)

# Simulations on the front page of yesterday's *New York Times*




MD simulations of COVID-19 virus in an aerosol drop
(Rommie Amaro and collaborators)





How did people do these things before they had powerful computers?

How did people do these things before they had powerful computers?

# Large-scale simulation in 1971

- Excerpt from “Protein synthesis: an epic on the cellular level”
- Performed at Stanford!



https://www.youtube.com/watch?v=u9dhO0iCLww