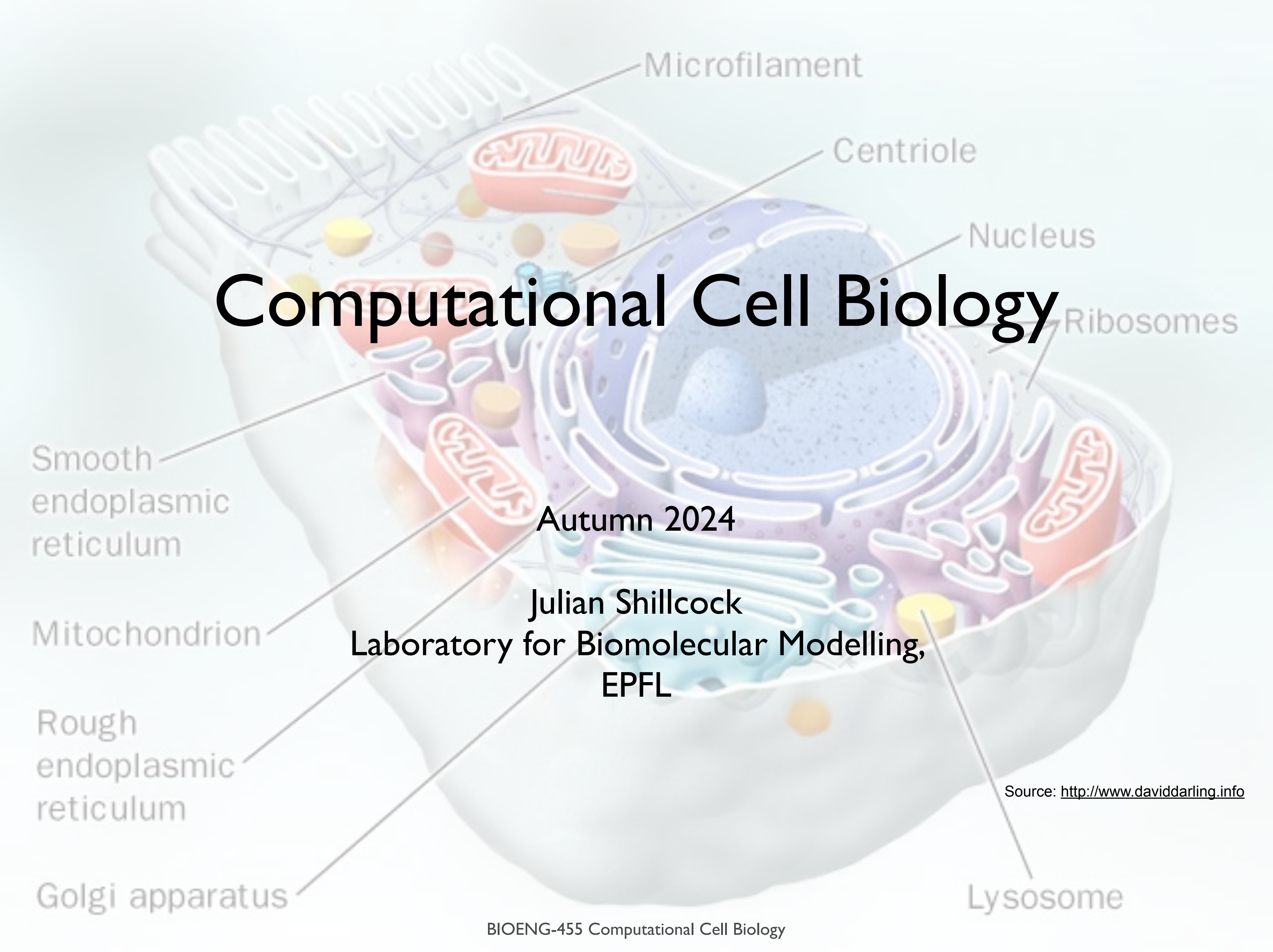


# Computational Cell Biology

Autumn 2024

Julian Shillcock
Laboratory for Biomolecular Modelling,
EPFL

Source: http://www.daviddarling.info

# Test 1

- ~ 1 hour
- no books or googling
- calculator allowed

# Core Concepts

We never simulate a real system…. only a model.

Simulations generate possible futures: we *compare* observables averaged in the simulation with *experiments* to see if the simulation is *good* or not.

Simulations abstract the world into *movies* to give us insight.

# Types of simulation

- Imagining scenarios where you negotiate a pay rise in a job
- Playing computer poker
- Trying to calculate a share price or stock market move in advance
- Integrating stochastic differential equations (Langevin equation)
- Molecular Dynamics
- Dissipative Particle Dynamics
- Monte Carlo simulations

# What is a simulation?

A simulation is not:

- analytically solving a differential equation or pde - e.g., ballistics versus weather
- quadratures - e.g., calculating a Fourier transform of electron density vs. P.F. $Z(\{\mathbf{x}\})$

## What is a simulation?

"a computer *experiment* of the behaviour of a *model* of a physical system in which matter is replaced by mathematical constructs that interact in ways that mimic the interactions in the physical system, and where the model's evolution generates states corresponding to those of the real system."

### Caveat

We never simulate a real system, but only a model of a real system; we first have to construct a model and then adapt it for calculation on a computer.

# Why do we do simulations?

- Experiments are too complicated and theories are too simple
- A computer language allows us to specify a sequence of steps that will solve a problem
- A model captures what we think are the important properties of an experiment, and allow us to ignore irrelevant aspects: if we later find the model is wrong, we can look for the missing important property
- We have almost complete control over all aspects of the simulation; so we can perform thought experiments like changing atmospheric pressure, turn electrostatic interactions on or off, etc
- Simulations are relatively cheap and quick compared to experiments
- No ethical concerns with simulating disease states
- We can visualize aspects of a simulation impossible in an experiment

# Molecular dynamics and Monte Carlo

In cellular biophysics there are two common types of simulation (but with sub-divisions and cross-over):

A) Mechanical types - that integrate more-or-less accurate equations of motion for interacting particles, .e.g., Newton's laws for Molecular Dynamics, Dissipative particle dynamics, Brownian dynamics, …

B) Statistical mechanical types - that calculate observable averages in specific thermodynamic ensembles: $\langle A \rangle = 1/Z \sum A(\{x_i, v_i\})e^{-\beta H(\{\mathbf{x}_i, \mathbf{v}_i\})}$, e.g., Monte Carlo.

These are mathematically distinct but physically equivalent (where they can both be applied) ways of calculating properties of a system.

Which is more useful depends on the problem of interest.

# Molecular dynamics

## A) Based on integrating some form of Newtonian equations of motion

| | | |
|---|---|---|
| $m.dv/dt = F$ | MD | ↑ |
| $m.dv/dt = F^C + F^D + F^R$ | DPD | Finer |
| $m.dv/dt = F^C - m\gamma.v + \sqrt{(2m\gamma k_B T)}.\zeta(t)$ | Langevin | Coarser |
| $0 = F^C - \gamma.v + \sigma.\zeta(t)$ | Brownian | ↓ |

The difference lies in what constitutes a "particle" and how complex the forces are.

In MD, the particles are atoms but in coarse-grained techniques, the particles are groups of atoms, molecular groups, even groups of molecules.

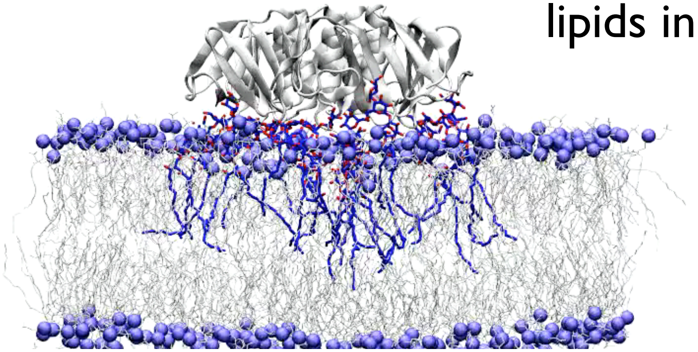
Once the particles are defined (mass, radius), and the forces are given (bonds, non-bonded, electrostatics), we integrate Newton's 2nd law and wait for equilibrium.

Allen, MP, and Tildesley, DJ, Computer Simulation of Liquids, Clarendon Press, Oxford, 1987

Frenkel, D and Smit, B, Understanding Molecular Simulation, Academic Press, 2002

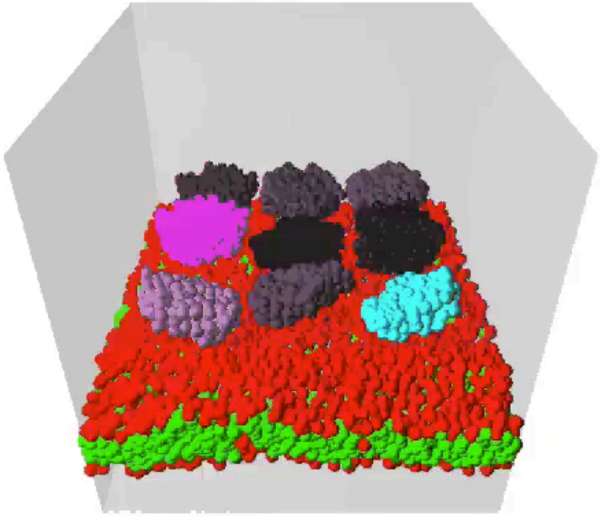
Berendsen, HJC, Faraday Discussions 144:467 (2010)

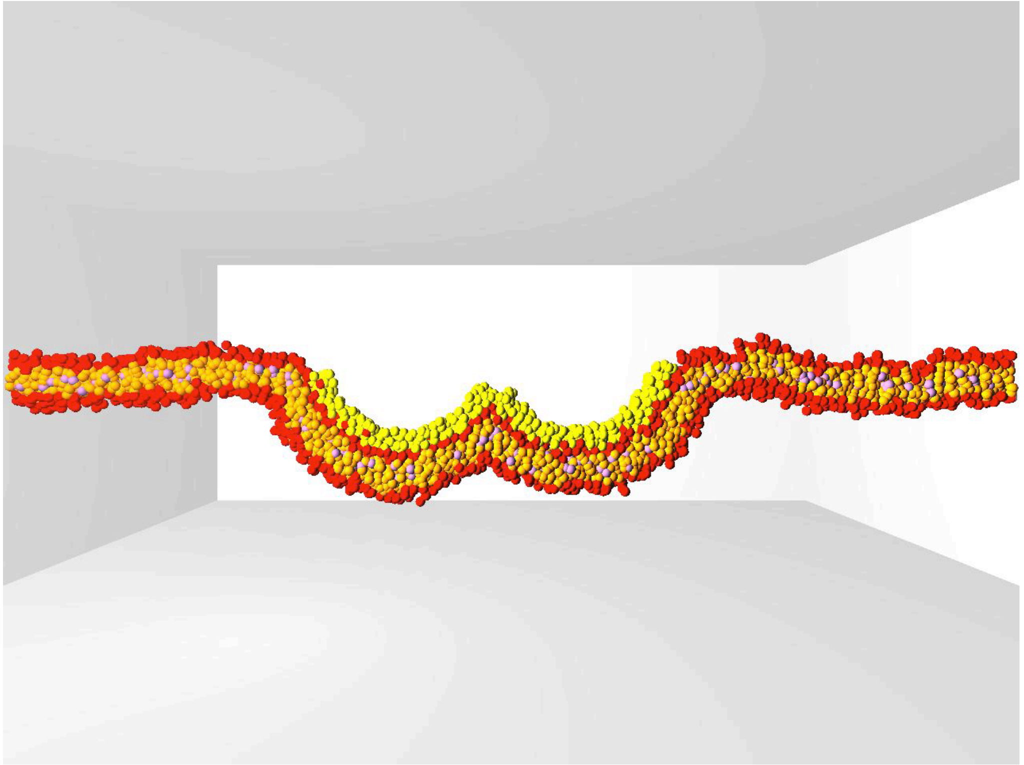
# Molecular dynamics/DPD



Shiga toxin binding to Gb3 lipids in membrane

Repulsive curvature-induced force on adsorbed nanoparticles





Entropic ordering of nanoparticles by membrane fluctuations

Solvent present here

# Brownian dynamics

ADP actin growing in steady-state; no treadmilling

ATP actin treadmilling (red = barbed end, faster growth, green = pointed end, slower growth); ends with ATP or ADP actin have different off rates

No solvent in these simulations

# Monte Carlo

B) Based on defining the Hamiltonian of a system, and performing phase space averages using the Metropolis Monte Carlo (MC) method.

$$<A> = 1/Z \sum A(\{x_i, v_i\})e^{-\beta H(\{\mathbf{x_i}, \mathbf{v_i}\})},$$

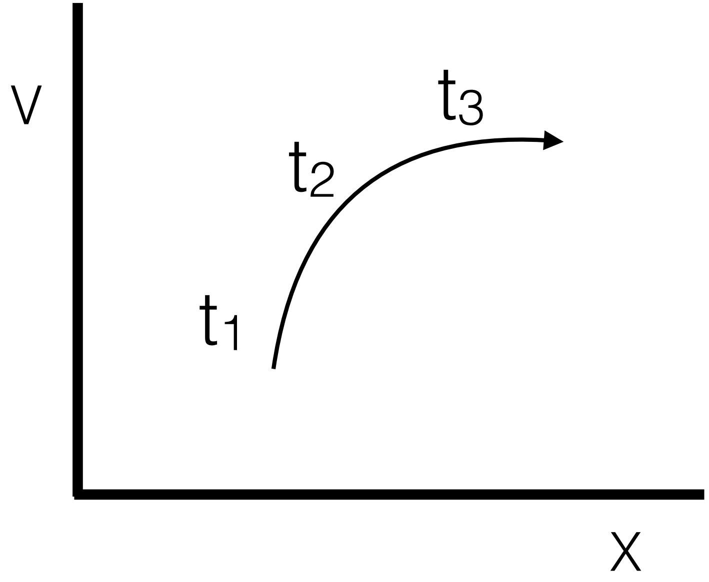
where the Partition Function $Z = \sum e^{-\beta H(\{\mathbf{x_i}, \mathbf{v_i}\})}$.

Monte Carlo simulations are generally simpler, faster, and more useful for calculating general thermodynamic properties of molecular systems than studying the behaviour of specific molecules.

But… the "particles" in MC simulations do not follow Newton's laws. They evolve according to a set of (possibly physically unrealistic) *moves*, in which the relevant degrees of freedom are continually changed so as to sample all allowed values, and observables have their values averaged over a large number of such moves.

Binder, K (ed.) Monte Carlo Methods in Statistical Physics, Topics in Current Physics Vol. 7, Springer Verlag, Berlin, 1986

# Phase Space



Consider a particle moving in a potential in 1d: x(t), v(t) (SHO blackboard)
As time passes, it moves around and its state can be represented as a point in phase space. Newton's 2nd law is:

F = m dv/dt

and the path in phase space is continuous.

Now consider N particles in 3d: there are 6N ( $\mathbf{x}_i(t)$, $\mathbf{v}_i(t)$ ) coordinates, and the phase space is *big*.

Now the state of ALL the particles is represented by a point in this 6N-dimensional space, and as time passes, and all the particles move around, this point moves along a trajectory in the space.

The purpose of Molecular dynamics, Monte Carlo, and other simulations, is to calculate a system's trajectory in phase space and measure the values of observables along that trajectory to make predictions.

# Think - Pair - Share
5 mins.

Pair up and discuss what a trajectory for one of the following systems would look like: identify the relevant degrees of freedom before you draw the trajectory. Or, invent your own system and its trajectory. 5 mins.

1. A pendulum in a grandfather clock
2. A pendulum swinging in water
3. A car on a circular racing track (at constant speed)
4. A cannonball fired from a cannon at 45 degrees upwards.
5. A single water molecule in a glass at room temperature
6. All molecules in a glass of water at room temperature
7. A water molecule in an ice cube

# Trajectories in phase space

## Mechanical simulation techniques: MD, DPD, BD

we integrate F = ma for a set of particles given a force field, and generate a path through phase space along which we calculate averages of measurable quantities, e.g., Temp, Press, order parameters, surface tension, etc.

## Monte Carlo simulations: MC

we define moves, which are discrete changes in the d.o.f of the system, and randomly change the coordinates of particles in the system. This also defines a "trajectory" through the system's phase space but it is not continuous, and does not represent an actual motion of the system.

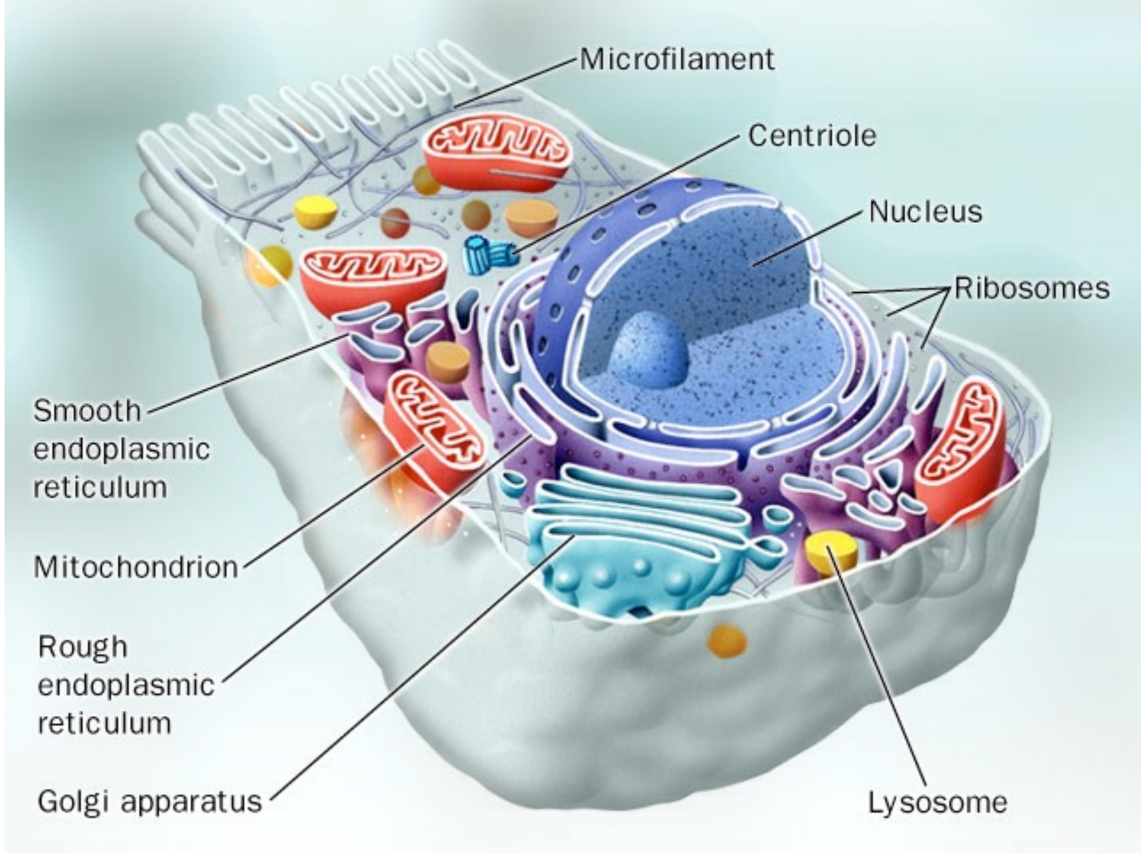
But… when we average over a long enough trajectory, we get the same results as for MD.

We won't cover MC any further in this course.

# Simulations of a cell

What kinds of cellular dynamics can we simulate?

Protein-protein binding
Protein diffusing in cytosol
Ion channel dynamics, pumps
Membrane potential dynamics
Filament self-assembly and collapse
Actin cytoskeleton dynamics
Membrane fluctuations
Vesicle transport
Vesicle fusion
Endo- and exocytosis
Golgi, ER self-assembly and transport
Cell crawling



www.daviddarling.info

We focus here on particle-based simulations as they are widely used for proteins, lipid membranes, macromolecular fluids, etc.

# Anatomy of a simulation

Setting up a simulation requires specifying precisely the **system** and its **surroundings**

| | |
|---|---|
| | What physical system do I want to simulate |
| System | What do I want to learn about it? |
| State | What length and time scales are important? (nm or mm? km for weather sims) |
| Boundary conditions | What are the boundary conditions? |
| Initial conditions | |
| | What are the entities of interest? atoms, molecules, etc. |
| Interactions | How do they interact? |
| Equations of motion | How does the simulation evolve? |
| Observables | What accuracy do I want? (larger system/longer run may be better, but more expensive) |

Usually we are interested in the *thermodynamic* properties of a system, so we connect the simulation to experiments via *thermodynamic coordinates*: temperature, pressure, volume, work, heat, electric field, charge, force, area, length, etc.

We do not simulate a real system, but only a model of a real system; we first construct the model (particles + forces) and second adapt it (discretize Newton's laws) for calculation on a computer.

# Physical System

The **System** is a physical experiment that we want to reproduce in a computer:

Argon gas in a box
Lipid molecules in a membrane
Ferromagnetic atoms on a lattice
Rodlike liquid crystal molecules in solvent

It must be in a measurable **State** with specified **Initial Conditions** and **Boundary Conditions:**

0.05 Mole Argon gas in a closed 1 litre glass bottle at 300 K
0.6 gm DMPC in 1 ml of water at STP

The physical entities must **Interact** in some way with each other and with the container; typically, the system has constant mass that is ensured by a closed container**:**

lipids in water can diffuse around, aggregate and separate, but are constrained by the walls of the container so that their number is constant

We need an **Equation of Motion** for the entities, usually Newton's laws or some artificially-chosen EOM. And we must be able to measure something, viz, **Observables**.

# Mathematical System

We represent physical entities as mathematical objects in well-defined *states*

Argon atoms in a box ⟹ point particles with mass, position, velocity and force field

Lipid molecules in membrane ⟹ ball-and-spring model, Lennard-Jones potential (6-12)

Ferromagnetic atoms on a lattice ⟹ Ising spins with 2 states: up or down

Rodlike liquid crystal molecules ⟹ rigid ellipsoids with non-spherically symmetric potential

Finally: what accuracy is required. Is it enough that atoms are billiard balls? Do we need charge? How many atoms? Do we want to see a phase transition?

NB. more accurate = slow and hard, less accurate = fast and easy.

Summary

A simulation = a physical system + a model + a mathematical algorithm for generating states of the model + observables that correspond to physical properties that can be measured.

# Units

Physical quantities have units (Mass, Length, Time) that define scales of interest in a system.

$k_BT = 4.14\ 10^{-21}$ J ~ 0.026 eV ~ 1/40 eV

4 pN.nm ~ 1 $k_BT$

mass of $e^-$ ~ 0.511 MeV

mass of $CH_4$ ~ 16 gm/mol ~ 2.6 $e^{-23}$ gm

1 Mole / litre ~ 0.6 molecules/$nm^3$

Computers know nothing of units; all quantities are dimensionless; all equations are discrete; all numbers are integers even when they're real, the same program run on different platforms (Windows, Linux, Mac) *will* produce different results.

This means that in a simulation we are explicitly (or implicitly) converting all dimensional quantities into dimensionless ones by multiplying/dividing by some standard M, L, T scales.

*Implicit* because if you forget the units, or get them wrong, the simulation will often happily continue, and produce rubbish, but it won't tell you.

# Reduced units

In MD (and, in fact, all particle-based simulations), once we have values for a mass $m_0$ (e.g., one atom), length $r_0$ (e.g., diameter of one atom) and energy (or temperature $k_B T$), we can make all other physical quantities dimensionless:

Reduced quantity = function of physical quantities

Mass $m = M/m_0$
Time $t = T/t_0$
Length $l = L/r_0$
Area $a = A/r_0^2$
Volume $v = (L/r_0)^3$
Density $\rho = \text{Density} . r_0^3/m_0 = N / l^3$
Diffusion constant $D' = (D. t_0/r_0^2)$
Area per lipid $a_{Lipid} = A/(N. r_0^2)$

The benefit of this is that we reduce the range of physical quantities (imagine simulating H with a mass $1.67\ 10^{-27}$ Kg) and can represent many physical systems by one simulation.

# System, State, Boundary Conditions

Suppose our *system* is a fluid, a *state* is defined by giving each particle a mass, x(t), v(t), F(t), … in a fixed volume.

But what happens at the walls? We need *boundary conditions*

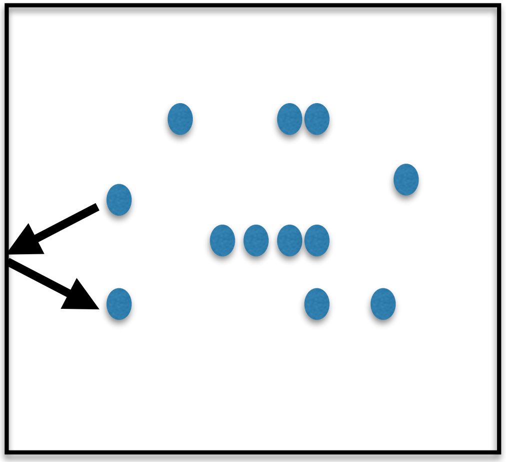
We would like to simulate a macroscopic system ($10^{22}$ particles) so we can apply thermodynamics, but we only have ~ $10^6$ - $10^8$ particles in a typical particle-based simulation.

There are two choices for the boundaries:

Hard boundaries - isolated system, large influence of walls on bulk

Periodic boundaries - infinitely periodic system, no walls at all!

For hard walls, particles just bounce off and have their normal velocity reversed. Easy to implement, but leads to artifacts due to finite system size.



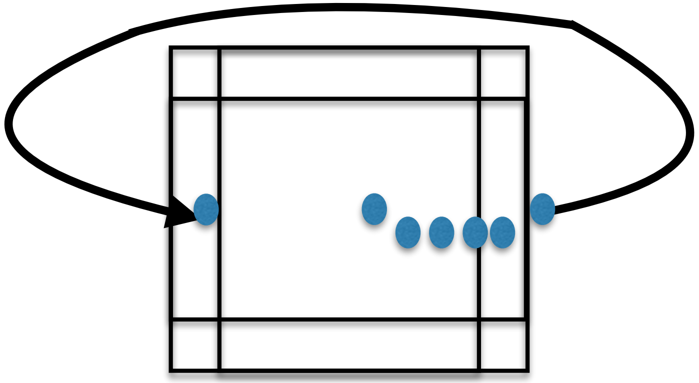
# Periodic boundary conditions

For periodic boundary conditions (PBC), there are no walls, and particles that leave the simulation volume by crossing a face are removed and replaced at the opposite face.

If $x(t + dt) > L$ then $x(t + dt) = x(t + dt) - L$

else if $x(t + dt) < 0$ then $x(t + dt) = x(t + dt) + L$

and for $y(t), z(t)$



This works because Newton's laws have translational invariance for central forces:

$$F(x+L) = m\, d^2(x+L)/dt^2 = m\, d^2x/dt^2 = F(x)$$

But we must only calculate the force between two particles **once** for the nearest pair (minimum image convention).

PBCs can lead to spurious effects if the range of F(x) is comparable to the box size, or there is periodic motion that spans the whole box.

# Ensembles in simulations

Common ensembles are those familiar from classical thermodynamics because we can apply the formulae of TD to the simulation: ensemble = set of observables that are constant over trajectory

## Microcanonical - N, V, E constant

Atomistic Molecular Dynamics simulations are usually done here - a fixed number of particles moving in a box of constant volume, and constant total energy. In practise, the integration scheme is not perfect, and the energy drifts over time and must be corrected.

Typically, atoms are placed in suitable positions subject to physical constraints (i.e., bonded atoms in a molecule should be separated by less than their bond length and not overlap); they are given a velocity drawn randomly from a Maxwell-Boltzmann distribution.

## Canonical - N, V, T constant

DPD, BD, MC and most coarse-grained simulations are done here as they involve random forces or noise, so the energy cannot be constant but the average energy, i.e., temperature, is constant.

## Grand Canonical - μ, V, T const

Uncommon because it is hard to change the particle number in a dense fluid, but sometimes useful.

# Initial conditions

The simulation also has to **start** somewhere - we must specify *initial conditions.*

We must set up the computer experiment correctly, otherwise our measurements are meaningless (compare doing an experiment on a cell and the room temperature changes unpredictably during the experiment.)

What properties are constant during the simulation?

Is the total mass constant (i.e., number of particles)?

Constant volume or pressure?

Constant temperature or energy?

The set of constant properties defines the thermodynamic **Ensemble** of the simulation.

# Types of initial condition

Particles in any ensemble must be assigned initial values of position, velocity, etc.

Can be assigned randomly or in a specific configuration:

- random initial state can be followed to see the appearance of an ordered state
- an initially ordered state can be followed as it relaxes to its equilibrium state

If we are interested in equilibrium, the simulation must be run for a time to ***forget*** the influence of the non-equilibrium initial state before we can start to sample observables.

This time must be determined for each system - it cannot usually be predicted in advance.

But sometimes, the initial phase of relaxation to equilibrium is also of interest.

See DPD user guide for examples of initial states

# Artifacts of initial conditions

Whatever ensemble we choose, a system may be unable to relax to equilibrium, or it relaxes very slowly, because of:

A) Conservation of some quantity - if total momentum is non-zero, Newton's laws will result in it remaining constant so the whole system may translate for ever.

B) Energy barriers between states - if the initial configuration is badly chosen, particles may be stuck in a region of phase space and unable to move.

C) Influence of the (artificial) boundary conditions - if we choose a cubic box, we cannot see an hexagonal packing of particles.

# Interactions

Interactions are defined by a *force field* that specifies the force between any two particles as a function of their position (and, sometimes, velocity)

Force fields can have many different terms depending on the interactions between the particles and the length and time scale of the simulation. In terms of their complexity, we have:

all-atom MD > coarse-grained MD >> DPD ~ Brownian Dynamics > MC

Hard spheres = billiard balls

Ising spin model of ferromagnet

Ball and spring model of a membrane

Lennard-Jones particles

Bond forces within polymers

Bending potential "

Torsion potential "

Hydrogen bonds, polarization, dipolar forces, ..

# Interactions

Lennard-Jones potential
(non-bonded particles)

$$V_{\mathrm{LJ}} = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]$$

Electrostatic force between
charged particles

$$F = k_e \frac{q_1 q_2}{r^2},$$

where $k_e$ is Coulomb's constant ($k_e = 8.99 \times 10^9$ N m$^2$ C$^{-2}$),

General Amber MD force field
(GAFF)
J. Comput. Chem. 25:1157-1174 (2004)

$$E_{\mathrm{pair}} = \sum_{\mathrm{bonds}} k_r(r - r_{\mathrm{eq}})^2 + \sum_{\mathrm{angles}} k_\theta(\theta - \theta_{\mathrm{eq}})^2 + \sum_{\mathrm{dihedrals}} \frac{v_n}{2}$$

$$\times [1 + \cos(n\phi - \gamma)] + \sum_{i<j}\left[\frac{A_{ij}}{R_{ij}^{12}} - \frac{B_{ij}}{R_{ij}^{6}} + \frac{q_i q_j}{\varepsilon R_{ij}}\right]$$

Martini coarse-grained MD force field
J. Phys. Chem. B 111:7812-7824 (2007)

$$U_{\mathrm{LJ}}(r) = 4\epsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r}\right)^{12} - \left(\frac{\sigma_{ij}}{r}\right)^{6}\right] \qquad U_{\mathrm{el}}(r) = \frac{q_i q_j}{4\pi\epsilon_0\epsilon_r r}$$

$$V_{\mathrm{bond}}(R) = \frac{1}{2}K_{\mathrm{bond}}(R - R_{\mathrm{bond}})^2 \qquad V_{\mathrm{id}}(\theta) = K_{\mathrm{id}}(\theta - \theta_{\mathrm{id}})^2$$

DPD has simpler forces, we'll cover these later

# MD equations of motion

Molecular Dynamics is a method for solving Newton's equation of motion $F = ma$ for a set of interacting particles (usually small spheres) to get their trajectories over time.

Observable properties that are functions of position and velocity $O(x, v)$ are then calculated by averaging over trajectories.

The differential equation is approximated by a difference equation, which is the Taylor series for position or velocity:

$$x(t + dt) \sim x(t) + dx/dt * dt + d^2x/dt^2 * dt^2/2 + d^3x/dt^3 * dt^3/6 + \dots$$

Finite difference schemes differ in:

- how many terms $dt^n$ are kept
- how many values of $x(t)$, $v(t)$ are needed
- how many times $F(x)$ must be calculated

these affect the accuracy, the computational cost and the memory required.

# Euler algorithm

Simplest method of solving $F = ma$: a first-order approximation.

Consider the Taylor series for the position $x(t)$ and velocity $v(t)$ of a particle in 1d:

$$x(t + dt) \sim x(t) + dx/dt * dt + d^2x/dt^2 * dt^2/2 + \ldots$$

$$v(t + dt) \sim v(t) + dv/dt * dt + d^2v/dt^2 * dt^2/2 + \ldots$$

Now $dx/dt = v(t)$ and $dv/dt = a(t) = F(t)/m$, and assume the $dt^2$ term is so small compared to the x, dx/dt terms that we can ignore it during the time step dt (equivalent to velocity being constant during dt):

$$\boxed{\begin{aligned} x(t + dt) &\sim x(t) + v(t) * dt + O(dt^2) + \ldots \\ v(t + dt) &\sim v(t) + (F(t)/m) * dt + O(dt^2) + \ldots \end{aligned}}$$

Algorithm: given x(t), v(t) at one time use these equations to iterate forwards in time.

# Verlet algorithm

Euler's algorithm has an error of order $dt^2$, because we neglected this term in the Taylor series. If we keep more terms, we get a more accurate trajectory.

Consider the same Taylor series for x(t) but expand it for x(t + dt) and x(t - dt):

$$x(t + dt) \sim x(t) + dx/dt * dt + d^2x/dt^2 * dt^2/2 + d^3x/dt^3 * dt^3/6 + O(dt^4)$$

$$x(t - dt) \sim x(t) - dx/dt * dt + d^2x/dt^2 * dt^2/2 - d^3x/dt^3 * dt^3/6 + O(dt^4)$$

now add them together (and write $d^2x/dt^2 = a(t) = F(t)/m$):

$$x(t + dt) \sim 2 * x(t) - x(t - dt) + (F(t)/m) * dt^2 + O(dt^4) + \ldots$$

$$v(t) \sim ( x(t + dt) - x(t - dt) )/2*dt$$

Algorithm: this is more accurate than Euler as the truncation error is $O(dt^4)$, and given x(t) at two times we use these equations to iterate forwards in time. Small problem is that v(t) has error $O(dt^2)$, but method does not need v(t) to calculate trajectory.

# Considerations for integration schemes

- How many times must we evaluate F(x)? Computational cost
- At how many time points must x(t), v(t) be known? Memory cost
- How big can $\Delta t$ be? Stability
- How accurate is the integration scheme? Truncation and round-off errors
- Are the forces truncated in space (e.g., electrostatics)?

- $\Delta t$ must be small enough so that neglected terms in the Taylor series of x(t) are small compared to the terms kept: for Euler the first neglected term is $O(\Delta t^2)$ so the local error is 2nd order; for Verlet, the local error is 4th order.

- Many other methods exist of higher order or different types (Haile, Sect. 4.4, pp157 ff)

- Euler/Verlet require only one calculation of force for each particle - this is the best we can do

- Some forces in the "force field" may require much smaller dt than others, making it inefficient to calculate all forces every time step.

# Observables

Given the set $\{\mathbf{x}_i(t), \mathbf{v}_i(t)\}$ for all the particles, we calculate observable properties of the system by integrating over the phase space trajectory of the particles.

e.g. Temperature = $k_B T \sim < 1/2 m.v_i^2 >$

But adjacent states are typically highly correlated so we have to simulate for a long time to get good statistics, and allow long gaps between samples to have independent measurements.

How do we know if our samples are independent?

We can calculate the auto-correlation function (= 2-point correlation function) of the observable of interest

$$C_2(\tau) = (\langle O(t+\tau).O(t)\rangle - \langle O(t)\rangle^2) \;/\; (\langle O(t)^2\rangle - \langle O(t)\rangle^2)$$

The number of time steps between samples must be at least as large as the *longest* correlation time of the observables of interest. Note that different observables can have different correlation times, so we cannot measure just one time period and use it for all observables.

Collective properties (CM of a membrane, surface tension) have longer correlation times than single particle properties (lipid tail length, velocity)

# Precision, error and truth

Experiments have external influences, e.g., temperature fluctuations, dirt, admixtures, etc.

A simulation has **systematic** errors and **statistical** errors

| **Systematic Errors** | **Statistical Errors** |
|---|---|
| Initial state | Too few samples |
| Finite system size | Samples too close together |
| Approximations in forces | Correlations |
| Truncation error (missing terms in Taylor series) | Stuck in metastable state |
| Round-off error (machine precision, sqrt, order of calculations) | |
| Random number generator isn't | |
| Bugs in the code | |

Any simulation that does not quantify and discuss the errors due to finite system size, finite run length, the influence of the initial state, boundary conditions, etc, is useless.

Reproducible ≠ Right.

# Truth?

Simulations are *approximations* to the truth.

It may be hard to define precisely what is a **State** or to quantify the effects of the **BCs**, and once they are defined it may be hard to represent them accurately on a computer.

The degree to which our *Model* corresponds to reality (or, how accurate our simulation is) follows from how accurately the mathematical properties of our model represent the behaviour of the real entities; or, how much of the physics of an experiment is captured in the simulation:

e.g., a RW captures a coin tossing experiment very well

an MD simulation of a drug molecule binding to a protein is not so accurate.

# Pros and Cons of Simulations

## Advantages

- Keeps only “relevant” properties
- Exact knowledge of microstates
- Equations of motion can be chosen
- Effects of each force term can be isolated and studied
- Measurements don’t perturb the system
- We can always repeat a simulation with exactly the same, or carefully-different conditions
- Usually cheaper than experiments

## Disadvantages

- What is a relevant property?
- We lose direct connection to experiments
- We see only what we expect to see
- One simulation gives one data point, and we may need many points
- Force fields can be hard to choose and not *transferable* between similar systems

# How do you choose a simulation type for a given soft matter problem?

Ask yourself:

- What are the length and time scales? ns and nm or microns or metres? Local or global properties?
- Am I interested in trends or absolute values?
- Do I need to study a *particular* molecule? e.g., DOPC vs DOPG or a generic "lipid"
- Are H-bonds (or C=C, or aromatic, etc) important?
- Do I have accurate values for interaction parameters between atoms/molecules?
- Am I interested only in equilibrium states?
- If not, what part of the dynamics do I need to get right?
- What computer resources do I have?

Often, there is not much choice about which technique to use as it is forced on you by the system you want to study: if the system of interest is large compared to atomic scale, you have to coarse-grained it.

# Summary

- Essentially only two types of simulation in cell biology:

  **Newtonian type** - Molecular dynamics, Brownian dynamics, DPD

  **Phase space integration** - Monte Carlo

- You never simulate a *real* system but only a *model* of a system
- Simulations have artifacts, approximations, errors - any simulation-based work that does not quantify/estimate the errors (statistical and systematic) is useless
- Reproducible does not mean Right
- When done well, simulations give us access to information that is hard or impossible to get from experiments or theoretical models

Break
10 mins.

# Exercise Today

1) Entropic spring exercise

2) How to run simulations on the HPC cluster helvetios.epfl.ch. See the folder on the moodle homepage that contains the executable, script, and instructions.

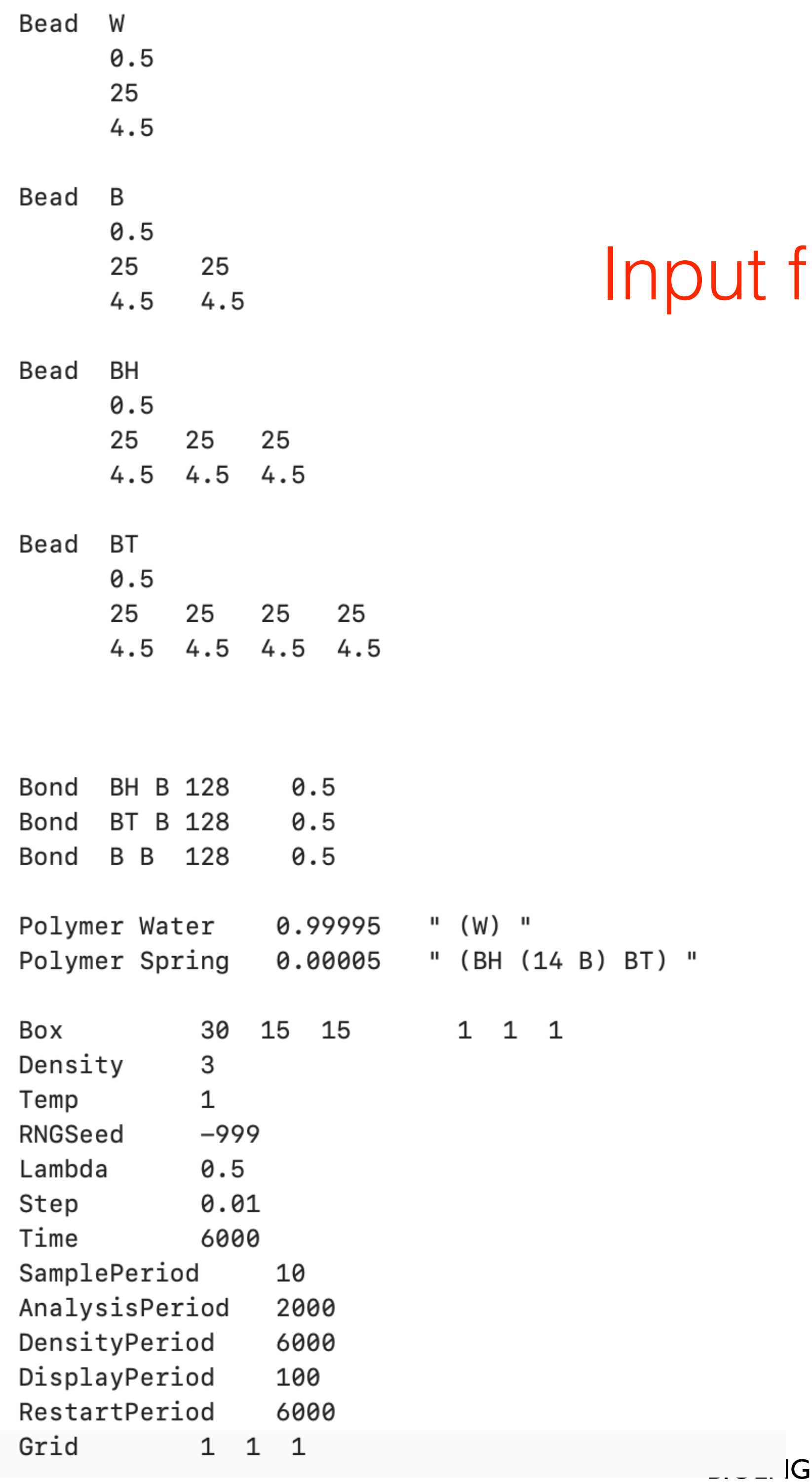
# Homework Ex. 1 Entropic spring

**Simulation of an entropic spring -** apply a stretching force to both ends of a single, long polymer in a DPD simulation and measure the end-end length as a function of the force (it probably has to be a very small force).

Then invert it to get F(L) and plot it including error bars of the statistical errors. How do you convert results to physical units?

Now make a fraction of the beads sticky (so that the polymer tends to stick to itself) and see how this changes the F(L) curve. You will need to vary the number of sticky beads to find an interesting regime (too few and nothing will happen, too many and the polymer will just stick together in a tight ball). Interesting means that the system shows some unusual, non-linear behaviour.

Needs commands in DPD to solve - see Section 8 of the User Guide.



```
Bead  W
      0.5
      25
      4.5

Bead  B
      0.5
      25    25
      4.5   4.5

Bead  BH
      0.5
      25   25   25
      4.5  4.5  4.5

Bead  BT
      0.5
      25   25   25   25
      4.5  4.5  4.5  4.5



Bond  BH B 128    0.5
Bond  BT B 128    0.5
Bond  B B  128    0.5

Polymer Water    0.99995   " (W) "
Polymer Spring   0.00005   " (BH (14 B) BT) "

Box         30  15  15       1  1  1
Density     3
Temp        1
RNGSeed     -999
Lambda      0.5
Step        0.01
Time        6000
SamplePeriod     10
AnalysisPeriod   2000
DensityPeriod    6000
DisplayPeriod    100
RestartPeriod    6000
Grid        1  1  1
```

Input file on moodle page for today:
dmpci.ex1

# Simulating an entropic spring under tension

We create command targets for the two ends of a molecule and apply equal and opposite forces to stretch it.

```
Command  SelectBeadTypeInSimBox 1  head  BH
Command  SelectBeadTypeInSimBox 1  tail  BT

Command  Comment  1000  // Apply a constant force to the first and last beads in
the +X and -X directions  //

Command ConstantForceOnTarget        1000  head  fh   1 0 0   5.0
Command ConstantForceOnTarget        1000  tail  ft   1 0 0   -5.0



 Command  Comment  5000  // Delete the applied forces  //

 Command RemoveCommandTargetActivity  5000   fh
 Command RemoveCommandTargetActivity  5000   ft
```

# Measuring the strain

## To Do:

1. Pick a box size of 30 x 15 x 15; adjust the number fractions to have 1 polymer of type (BH (14 B) BT), i.e., distinct head and tail beads so they can be selected.
2. Turn force on at T = 1000 steps. How long should you keep it on?
3. How can you measure the extension?
4. Next, change the backbone to contain a new bead type that is "sticky". Try (BH B B B S S S S B B B BT), and give S the same interactions as B except for its self interaction that is reduced to make it sticky. Vary the number of S beads until you find a value that makes an observable difference.

## Questions to answer

What is the stress/strain relation F(L) for the "molecular spring"?

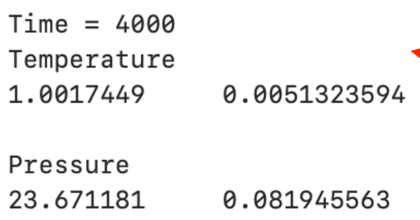
Does it have different regimes for F(L) under different tensions? Why?

With sticky beads there are two new parameters: the number of sticky beads and their self-interaction. How can you reasonably select values for these?

# dmpcas file contains time-averaged observables

## Typically there are 2 columns:   mean and standard deviation

dmpcas.ex1 has <Lee> averaged between 2000 - 4000 steps

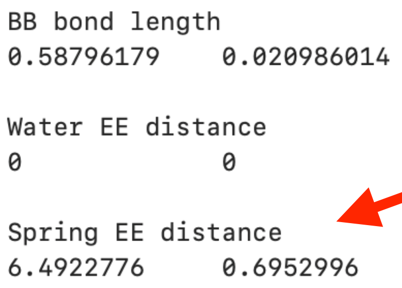


```
Time = 4000
Temperature
1.0017449      0.0051323594

Pressure
23.671181      0.081945563
```

Use the time period while force is applied, which is at the end time of the period

. . .



```
BB bond length
0.58796179     0.020986014

Water EE distance
0              0

Spring EE distance
6.4922776      0.6952996
```

This is the mean and std. dev. of polymer end-to-end length