Computación con GPUs: Aplicación a la simulación de membranas biológicas

Enrique Velasco y Teresa Ruiz

Departamento de Física Teórica de la Materia Condensada

Universidad Autónoma de Madrid

Molecular Dynamics (MD) simulations

One of the most widely used techniques for studying condensed systems (fluids, solids,…)

Access to equilibrium and dynamical properties

Some old and recent applications:

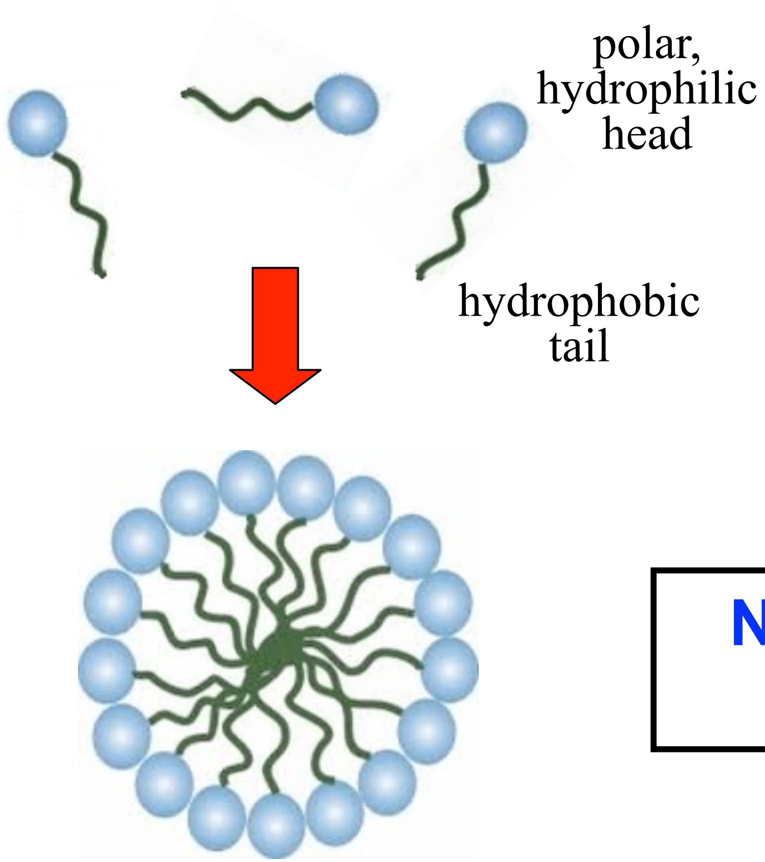
- Liquids (first applications ‘40, ‘60, …)
- Complex fluids (SOFT MATTER):

  liquid crystals, colloids, amphiphilic systems,…

→ Biological systems (macromolecules, proteins,…)

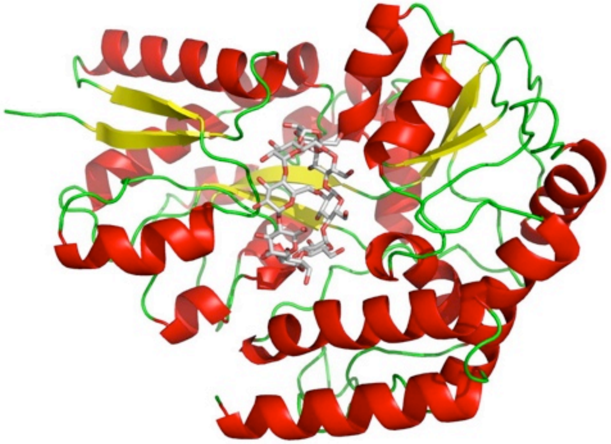
Limiting factors in complex fluid applications: mesoscales

- *Length* scales (not nm but 100 nm – 1 μm)
- *Time* scales (not ps but ns – μs)



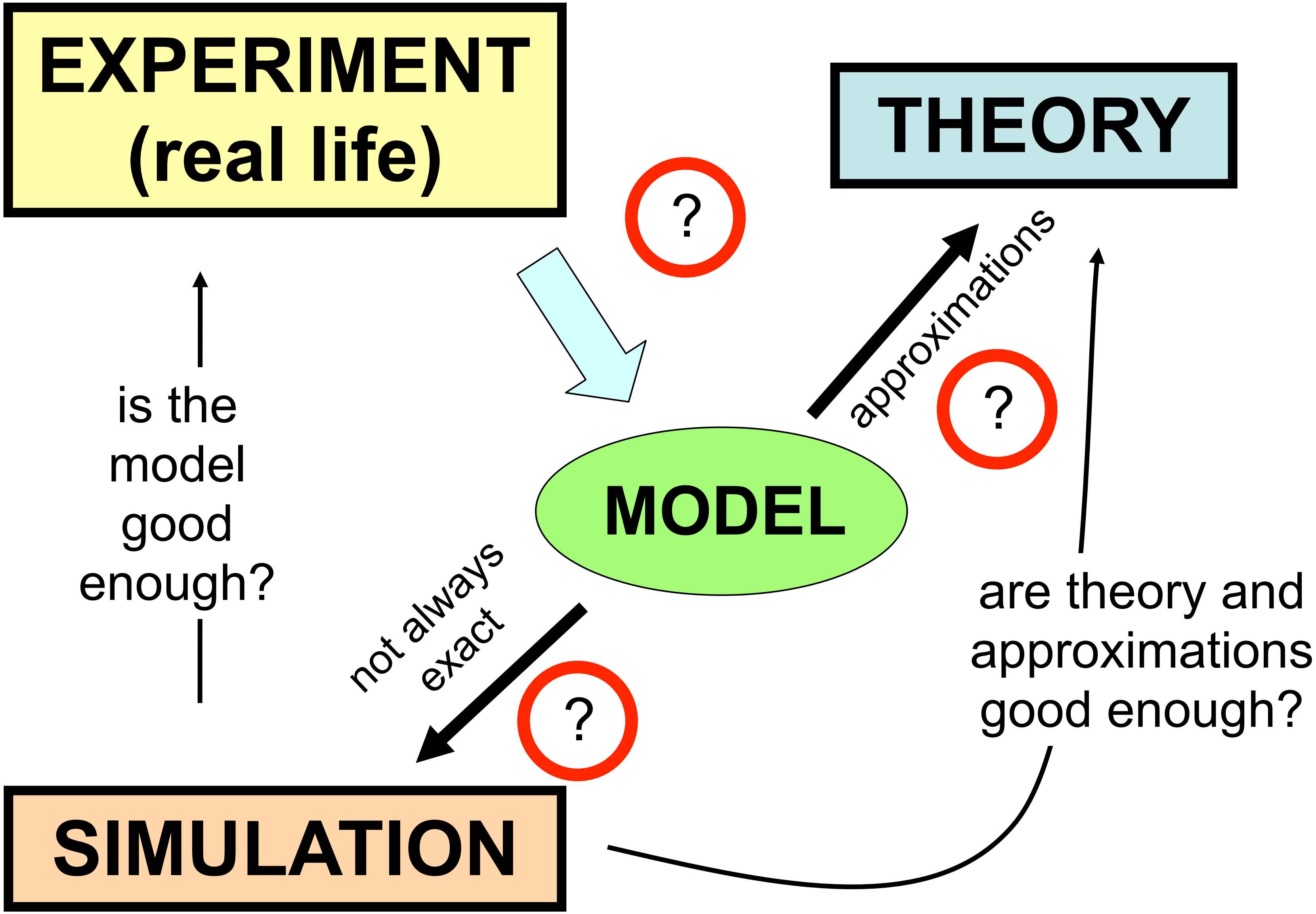
MICELLE FORMATION

PROTEIN DYNAMICS



NEED FOR COARSE GRAINING

MESOSCOPIC MODELS



EXPERIMENT (real life)

THEORY

?

approximations

?

is the model good enough?

MODEL

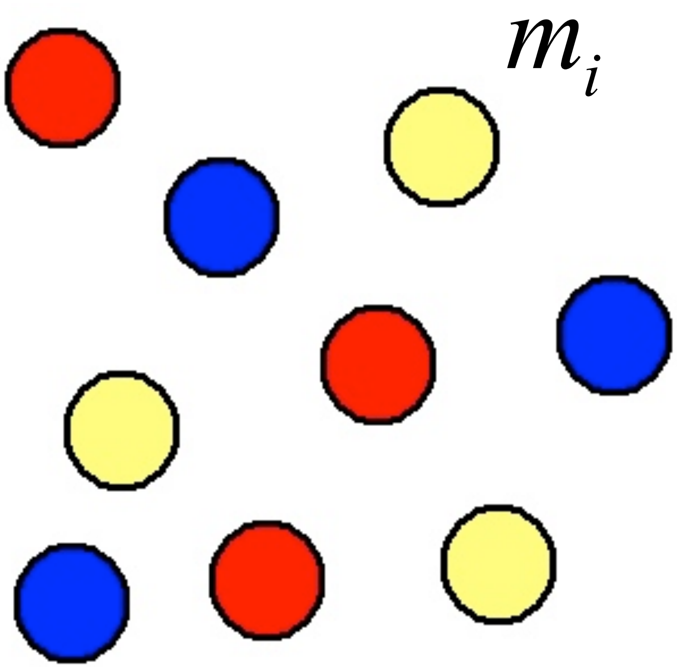
not always exact

?

are theory and approximations good enough?

SIMULATION

MD (classical) simulations: basic facts



We solve classical Newton's equations of motion and obtain dynamical trajectories from the forces:

Hamiltonian system (energy conserved)

$$\begin{cases} \dfrac{d\mathbf{r}_i}{dt} = \mathbf{v}_i, \\ m_i \dfrac{d\mathbf{v}_i}{dt} = \mathbf{F}_i, \end{cases}$$

$$i = 1, 2, \ldots, N$$

$$\mathbf{F}_i = -\nabla_i \Phi$$

potential energy function

- The potential energy Φ has to be specified
- A finite-difference scheme is used

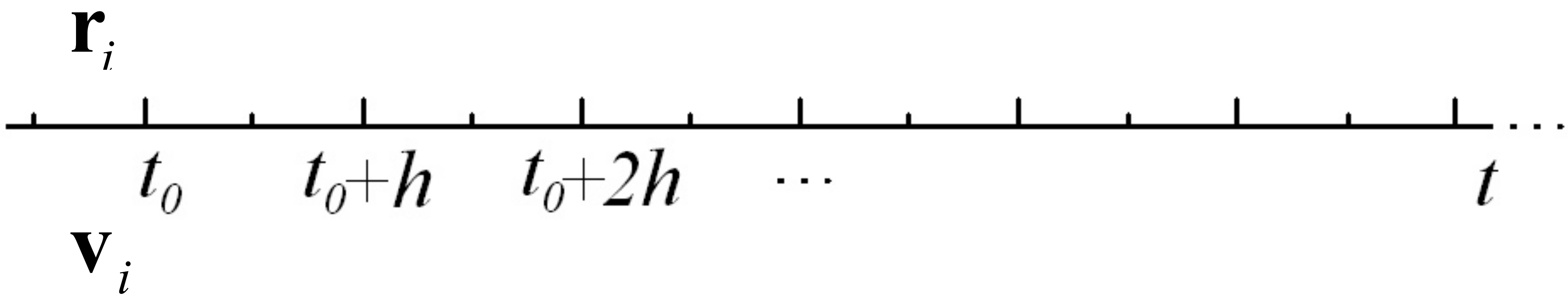
$$\Phi = \Phi\left(\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N\right)$$

Finite-difference scheme: **Verlet (leap-frog) algorithm**

$$t \text{ ® } t_0, t_1, t_2, \ldots \quad t_n = nh, \;\; n = 0,1,2,\ldots$$

$$h: \text{ MD simulation step}$$

$$\begin{cases} \mathbf{v}_i\left(t_n + \dfrac{h}{2}\right) = \mathbf{v}_i\left(t_n - \dfrac{h}{2}\right) + \dfrac{h}{m_i}\mathbf{F}_i(t_n), \\ \\ \mathbf{r}_i(t_n + h) = \mathbf{r}_i(t_n) + h\mathbf{v}_i\left(t_n + \dfrac{h}{2}\right) \end{cases} \qquad i = 1, 2, \ldots, N$$

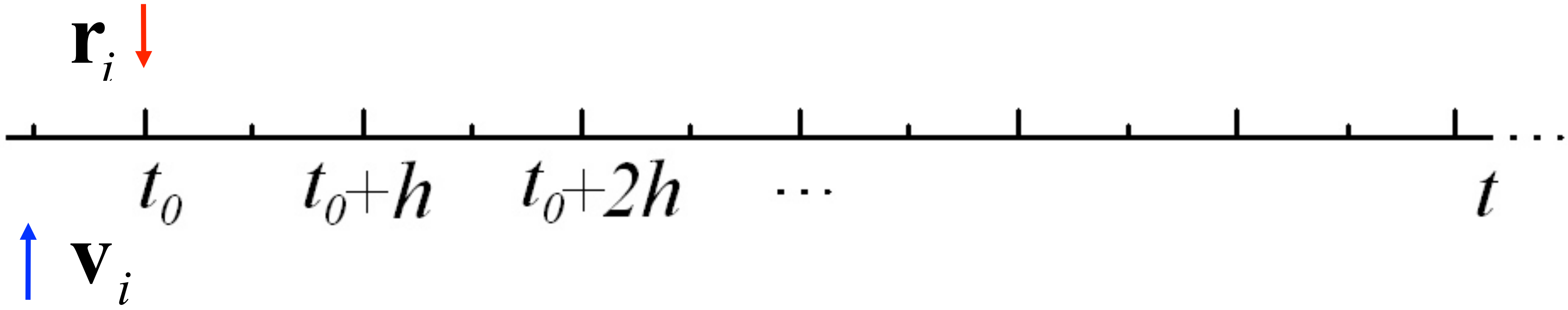


Finite-difference scheme: **Verlet (leap-frog) algorithm**

$$t \circledR \; t_0, t_1, t_2, \ldots \quad t_n = nh, \;\; n = 0,1,2,\ldots$$

$$h: \text{ MD simulation step}$$

$$\begin{cases} \mathbf{v}_i\left(t_n + \dfrac{h}{2}\right) = \mathbf{v}_i\left(t_n - \dfrac{h}{2}\right) + \dfrac{h}{m_i}\mathbf{F}_i(t_n), \\[2em] \mathbf{r}_i(t_n + h) = \mathbf{r}_i(t_n) + h\mathbf{v}_i\left(t_n + \dfrac{h}{2}\right), \end{cases} \qquad i = 1, 2, \ldots, N$$

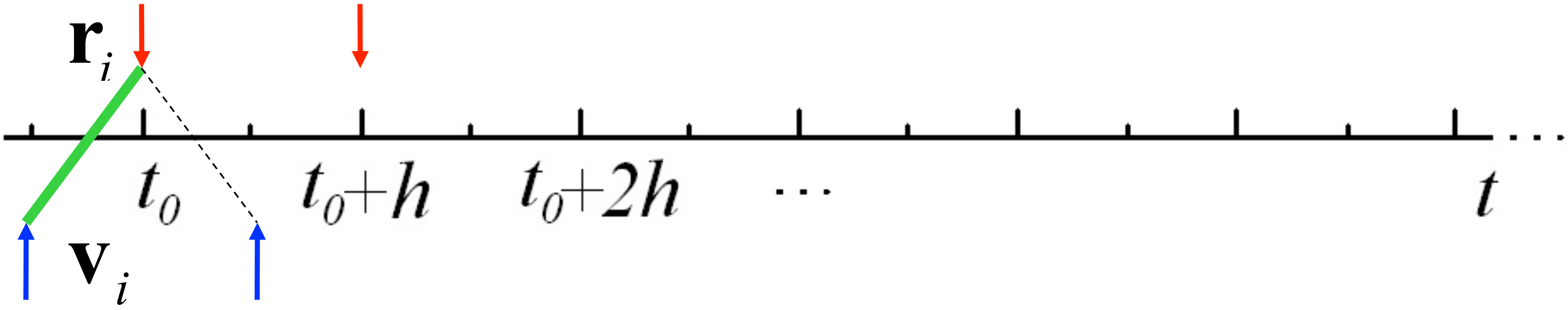


Finite-difference scheme: **Verlet (leap-frog) algorithm**

$$t \rightarrow t_0, t_1, t_2, \ldots \quad t_n = nh, \;\; n = 0,1,2,\ldots$$

$$h: \text{ MD simulation step}$$

$$\begin{cases} \mathbf{v}_i\left(t_n + \dfrac{h}{2}\right) = \mathbf{v}_i\left(t_n - \dfrac{h}{2}\right) + \dfrac{h}{m_i}\mathbf{F}_i(t_n), \\ \\ \mathbf{r}_i(t_n + h) = \mathbf{r}_i(t_n) + h\mathbf{v}_i\left(t_n + \dfrac{h}{2}\right), \end{cases} \qquad i = 1, 2, \ldots, N$$

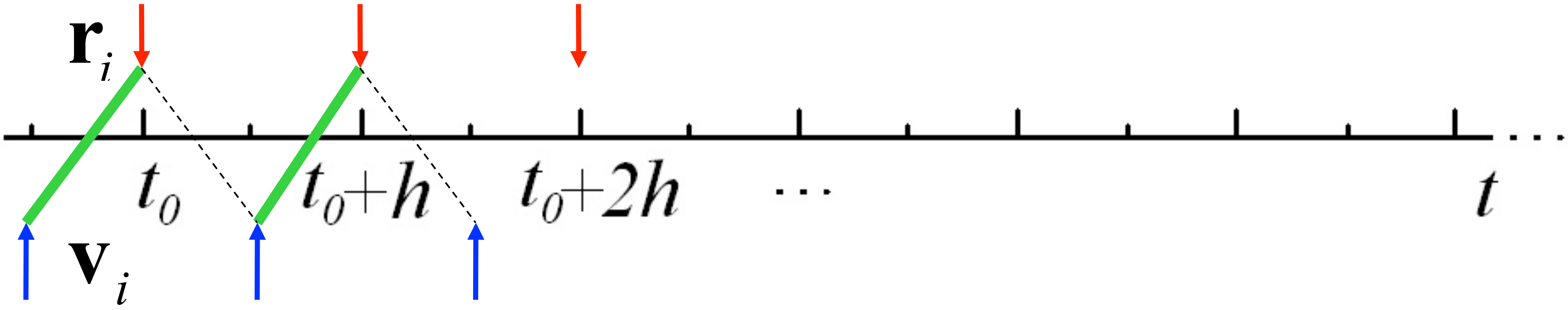


Finite-difference scheme: **Verlet (leap-frog) algorithm**

$$t \text{ ® } t_0, t_1, t_2, \ldots \quad t_n = nh, \;\; n = 0,1,2,\ldots$$

$$h: \text{ MD simulation step}$$

$$\begin{cases} \mathbf{v}_i\left(t_n + \frac{h}{2} \div\right) = \mathbf{v}_i\left(t_n - \frac{h}{2} \div\right) + \frac{h}{m_i}\mathbf{F}_i(t_n), \\ \\ \mathbf{r}_i(t_n + h) = \mathbf{r}_i(t_n) + h\mathbf{v}_i\left(t_n + \frac{h}{2} \div\right) \end{cases} \qquad i = 1, 2, \ldots, N$$

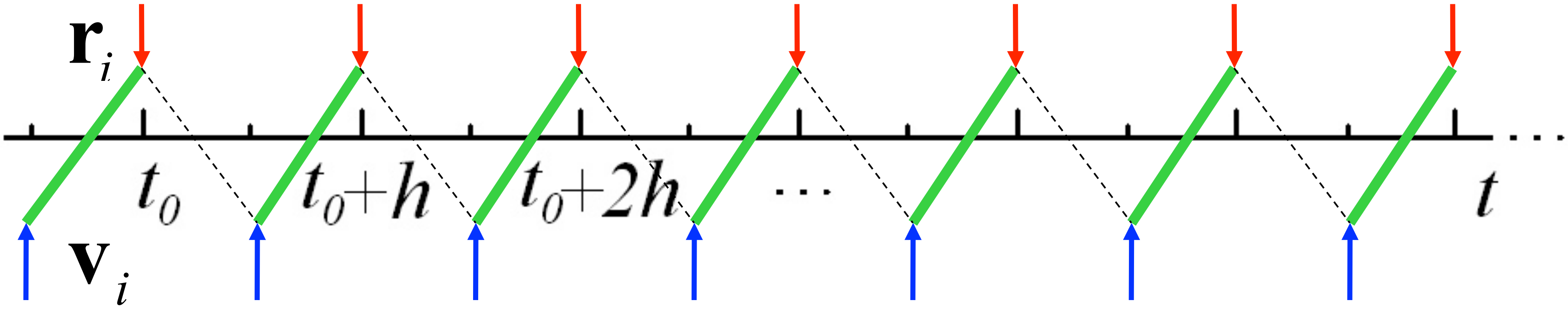


Finite-difference scheme: **Verlet (leap-frog) algorithm**

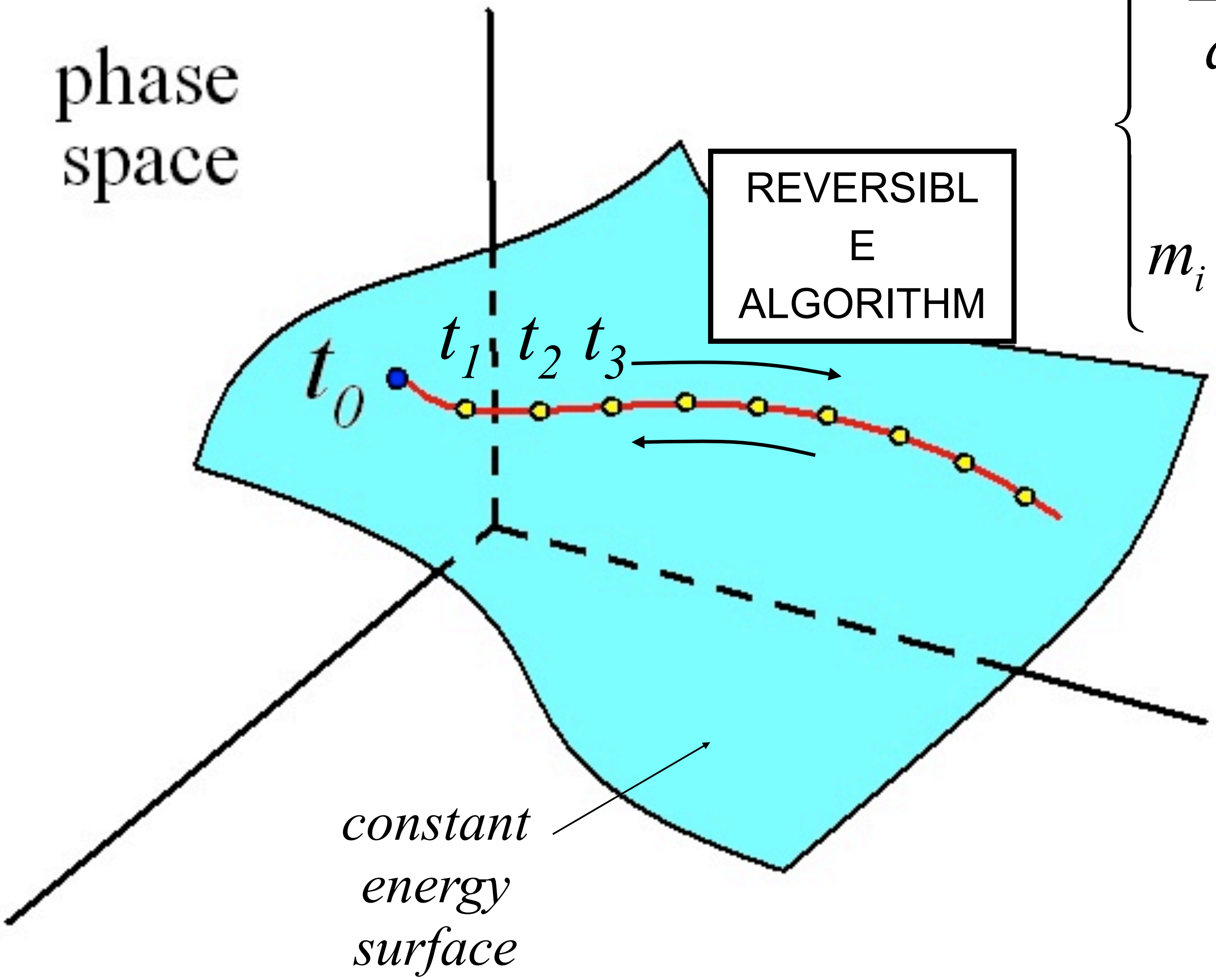
$$t \text{ ® } t_0, t_1, t_2, \ldots \quad t_n = nh, \;\; n = 0,1,2,\ldots$$

h : MD simulation step

$$\begin{cases} \mathbf{v}_i\left(t_n + \dfrac{h}{2}\right) = \mathbf{v}_i\left(t_n - \dfrac{h}{2}\right) + \dfrac{h}{m_i}\mathbf{F}_i(t_n), \\ \\ \mathbf{r}_i(t_n + h) = \mathbf{r}_i(t_n) + h\mathbf{v}_i\left(t_n + \dfrac{h}{2}\right), \end{cases} \qquad i = 1, 2, \ldots, N$$



$$\begin{cases} \dfrac{d\mathbf{r}_i}{dt} = \mathbf{v}_i, \\[2em] m_i \dfrac{d\mathbf{v}_i}{dt} = \mathbf{F}_i, \end{cases}$$



$$\begin{pmatrix} \mathbf{v}_i\left(t_n+\frac{h}{2}\right) \\ \\ \mathbf{r}_i\left(t_n+h\right) \end{pmatrix} = J \begin{pmatrix} \mathbf{v}_i\left(t_n-\frac{h}{2}\right) \\ \\ \mathbf{r}_i\left(t_n\right) \end{pmatrix}$$

$$\det J = 1$$

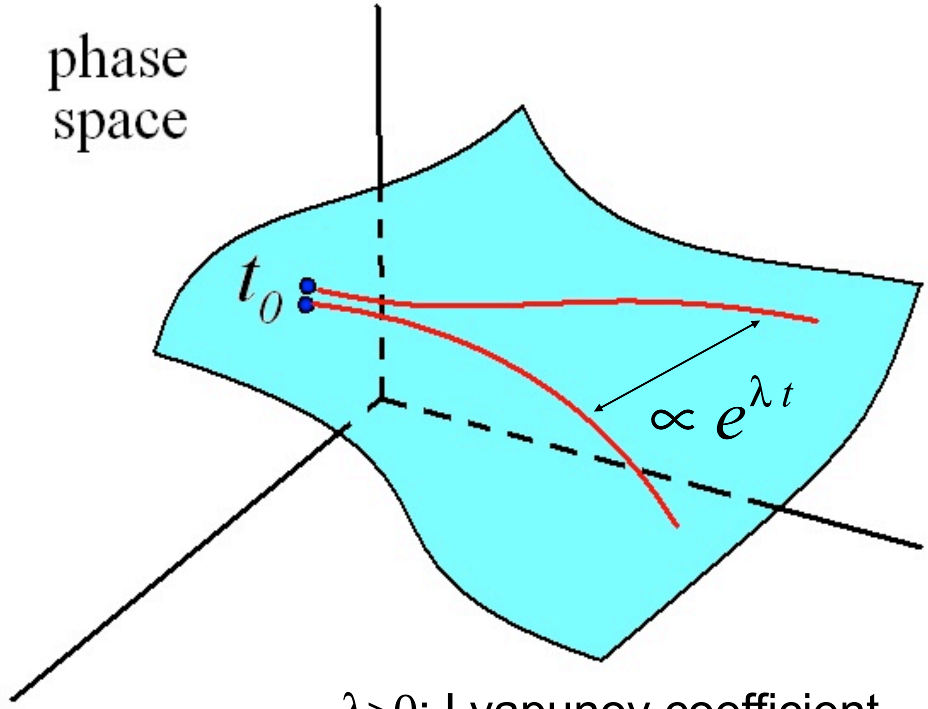
SYMPLECTIC
ALGORITHM

Space volume is conserved (as required by Liouville's theorem of Statistical Mechanics)

Classical many-body systems are intrinsically CHAOTIC

Therefore the algorithm cannot reproduce trajectories exactly

Only requirements are *reversibility* and *symplecticity*



$\lambda>0$: Lyapunov coefficient

Thermodynamic, dynamic, microscopic properties result as TIME AVERAGES over phase space trajectories. For example, the energy:

$$E = \langle H \rangle = \frac{1}{\tau} \sum_n H_n$$

Extensions of constant-energy MD:

- **Isothermal MD:**

the system is coupled to an external heat-bath (thermostat at specific temperature *T*), with which energy is exchanged to maintain a constant temperature

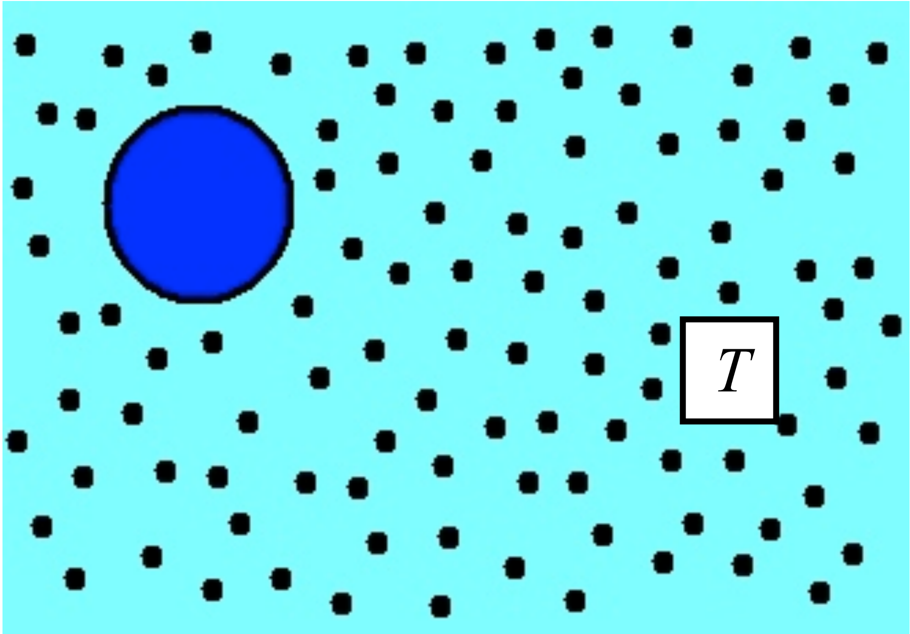
<u>Example</u>: Langevin thermostat

random force

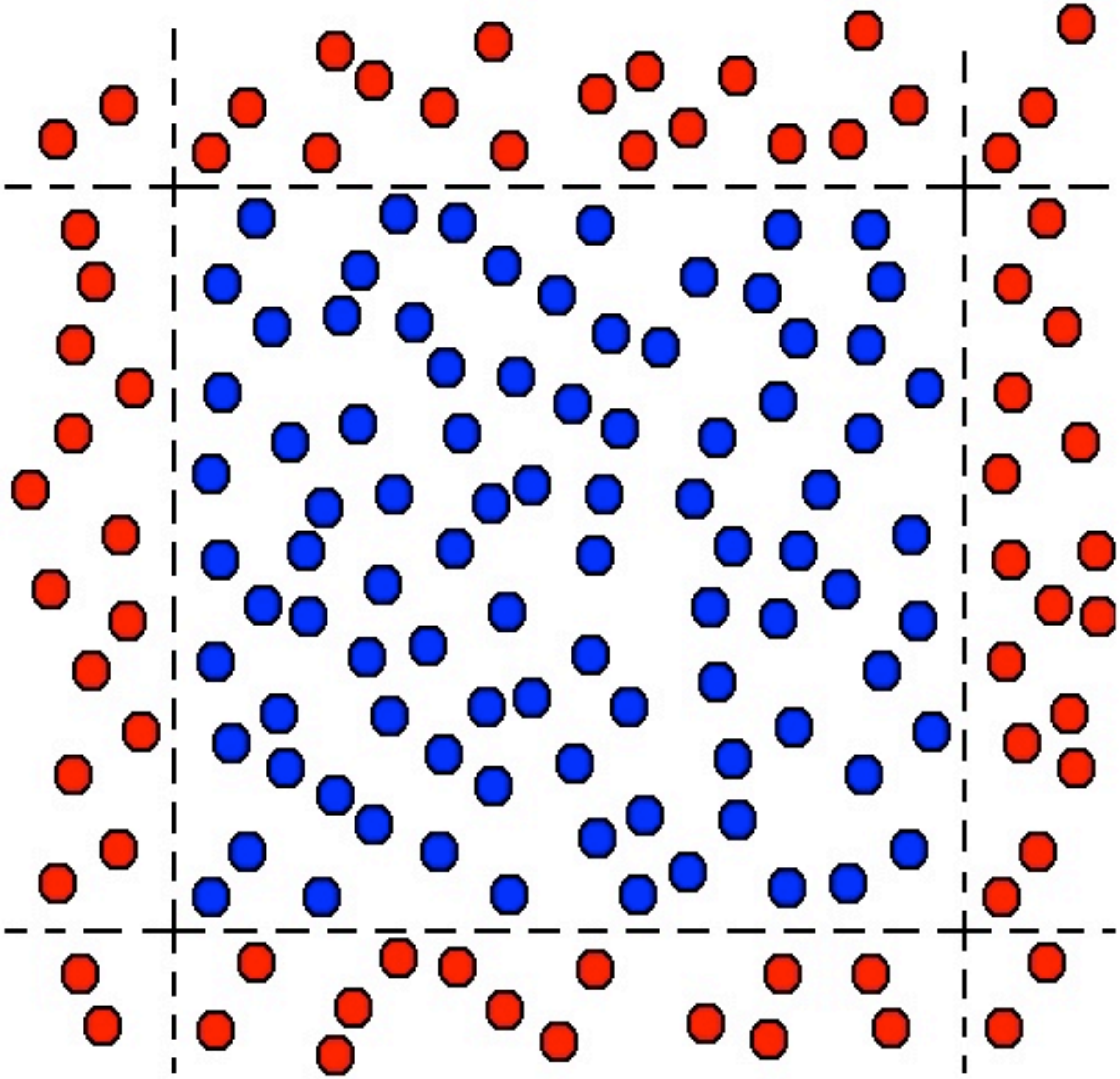
$$m_i \frac{d\mathbf{v}_i}{dt} = -\xi \, \mathbf{v}_i + \mathbf{g}_i + \mathbf{F}_i$$

friction term

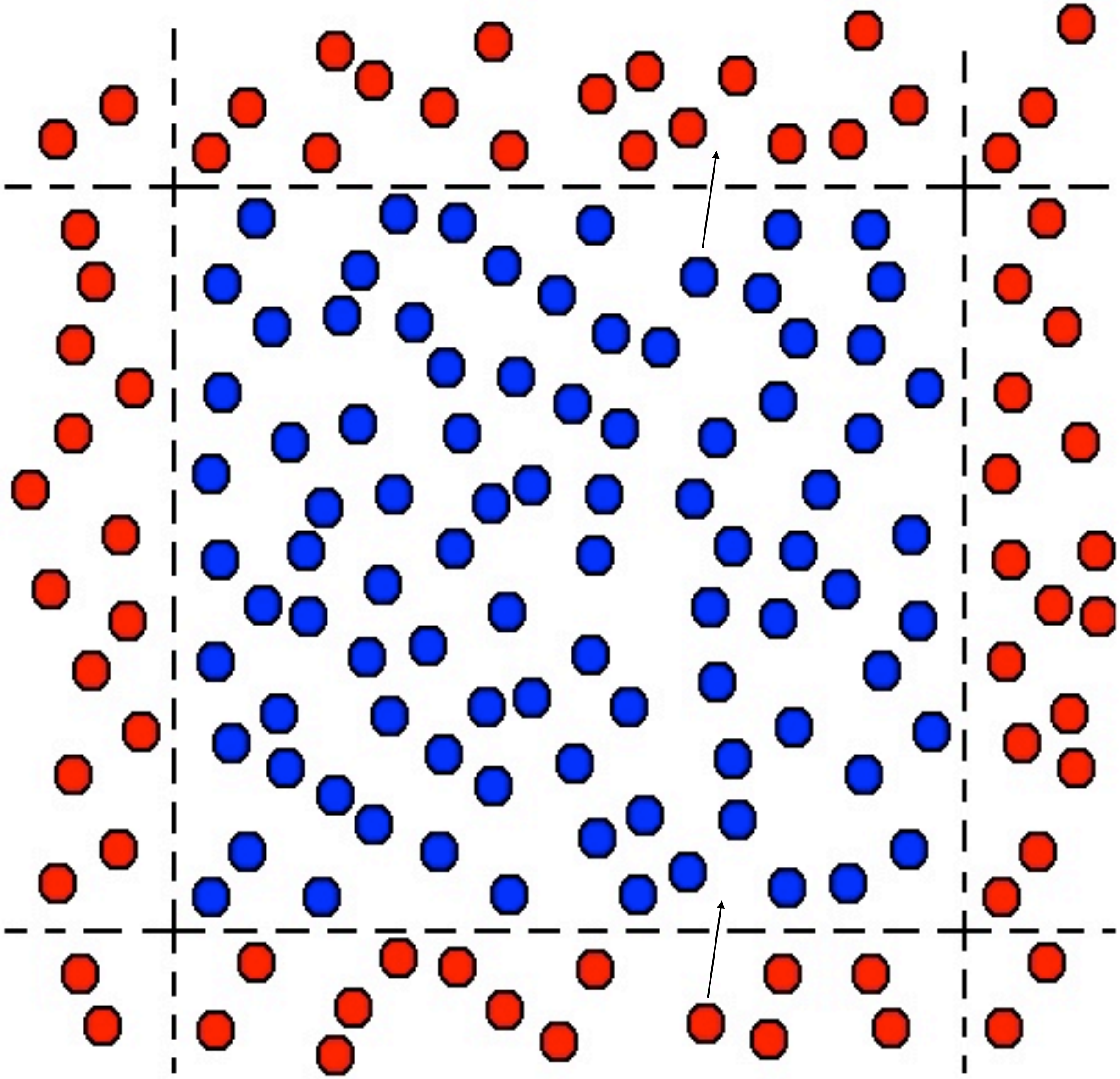
deterministic force



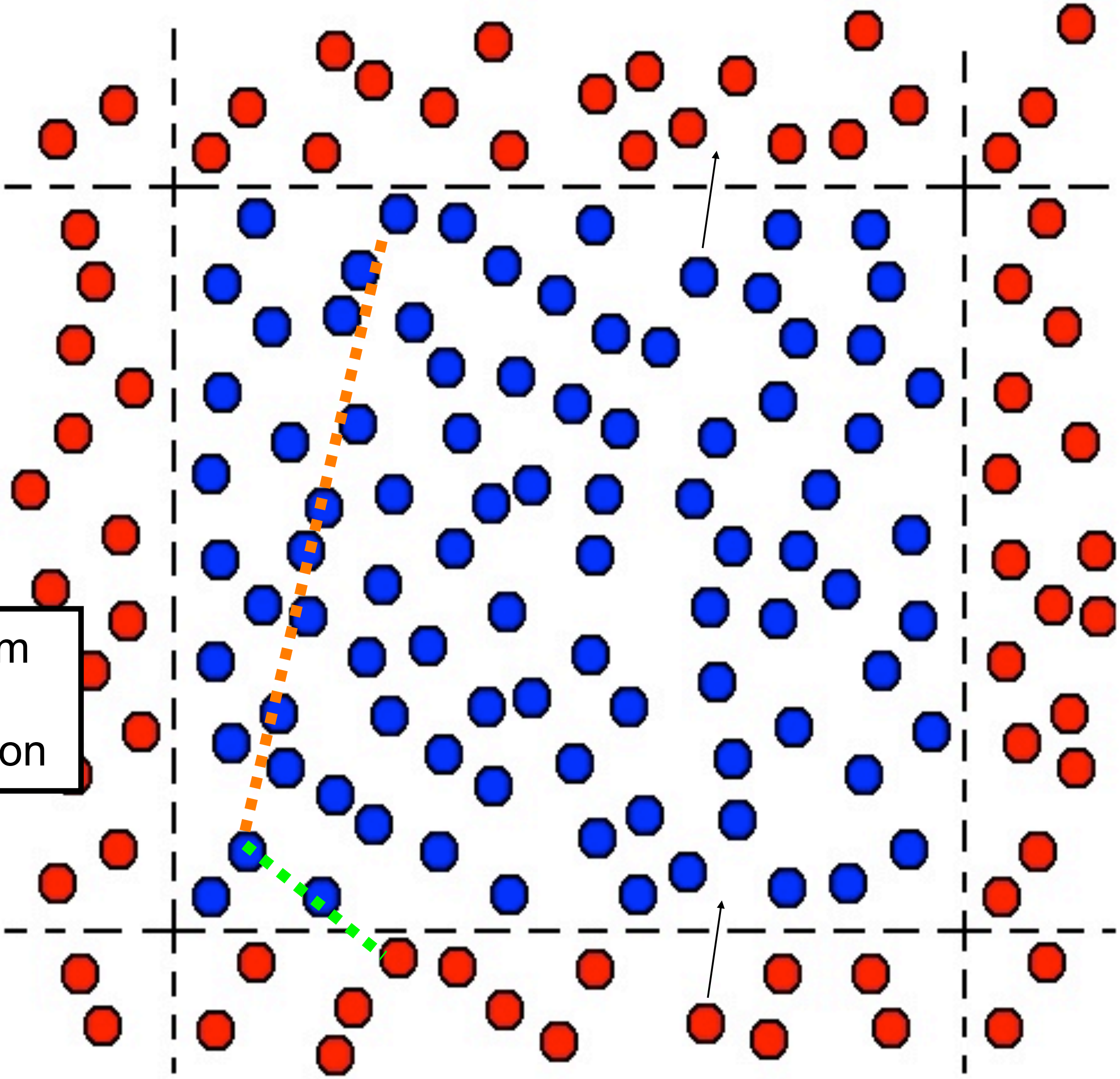
periodic boundary conditions (2D)



periodic boundary conditions (2D)



periodic boundary conditions (2D)



minimum image convention

The time-consuming part of the calculation is the force:

$$\mathbf{F}_{ij} = \mathbf{F}_{ji}$$

$$m_i \frac{d\mathbf{v}_i}{dt} = \mathbf{F}_i \xrightarrow{\text{assuming two-body forces}} \mathbf{F}_i = \sum_{j \neq i} \mathbf{F}_{ij}$$

Force calculation amounts to up to 90% of CPU time required in whole calculation

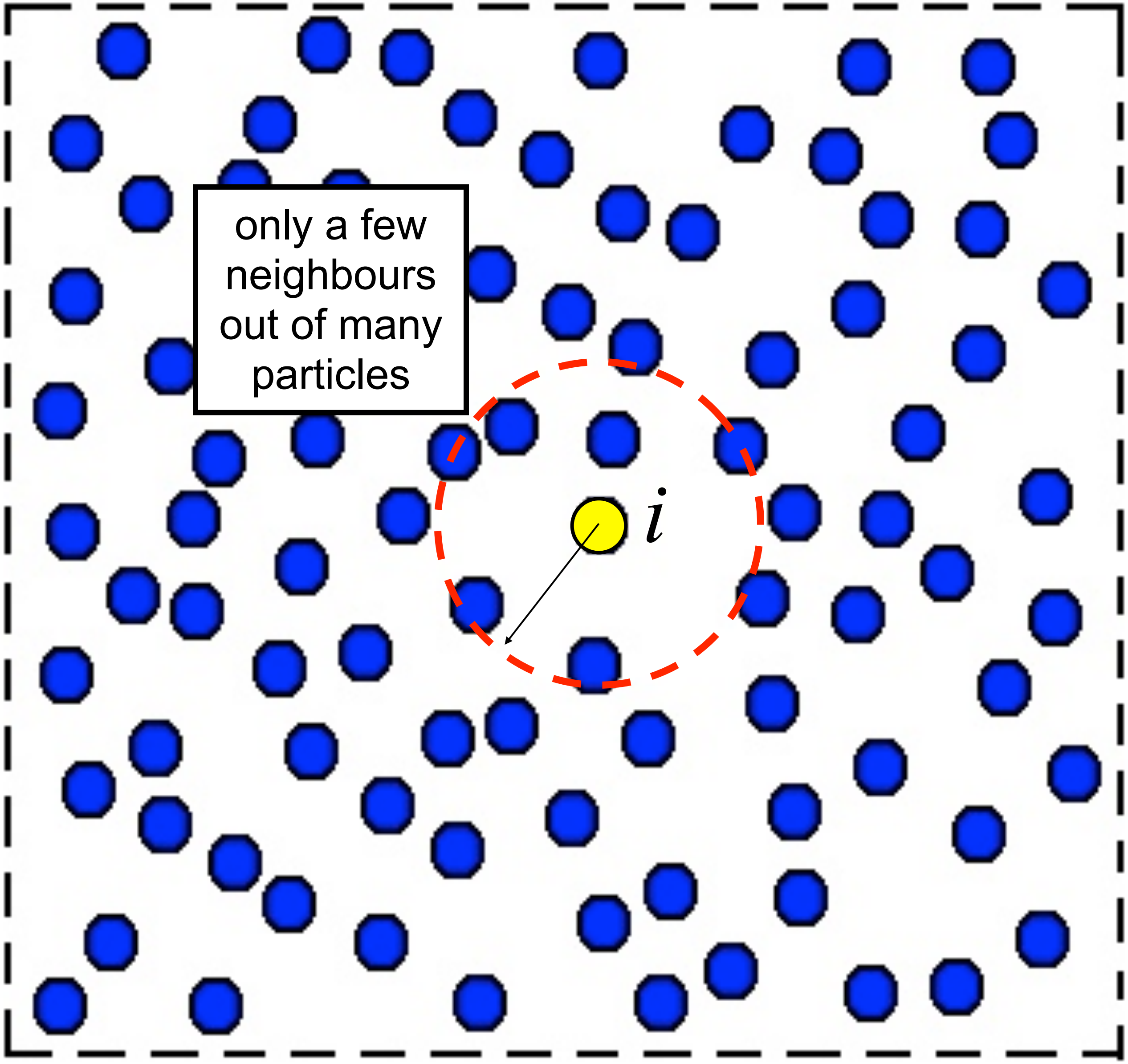
⇨ mayor barrier to large-scale simulations

need for new algorithms and techniques

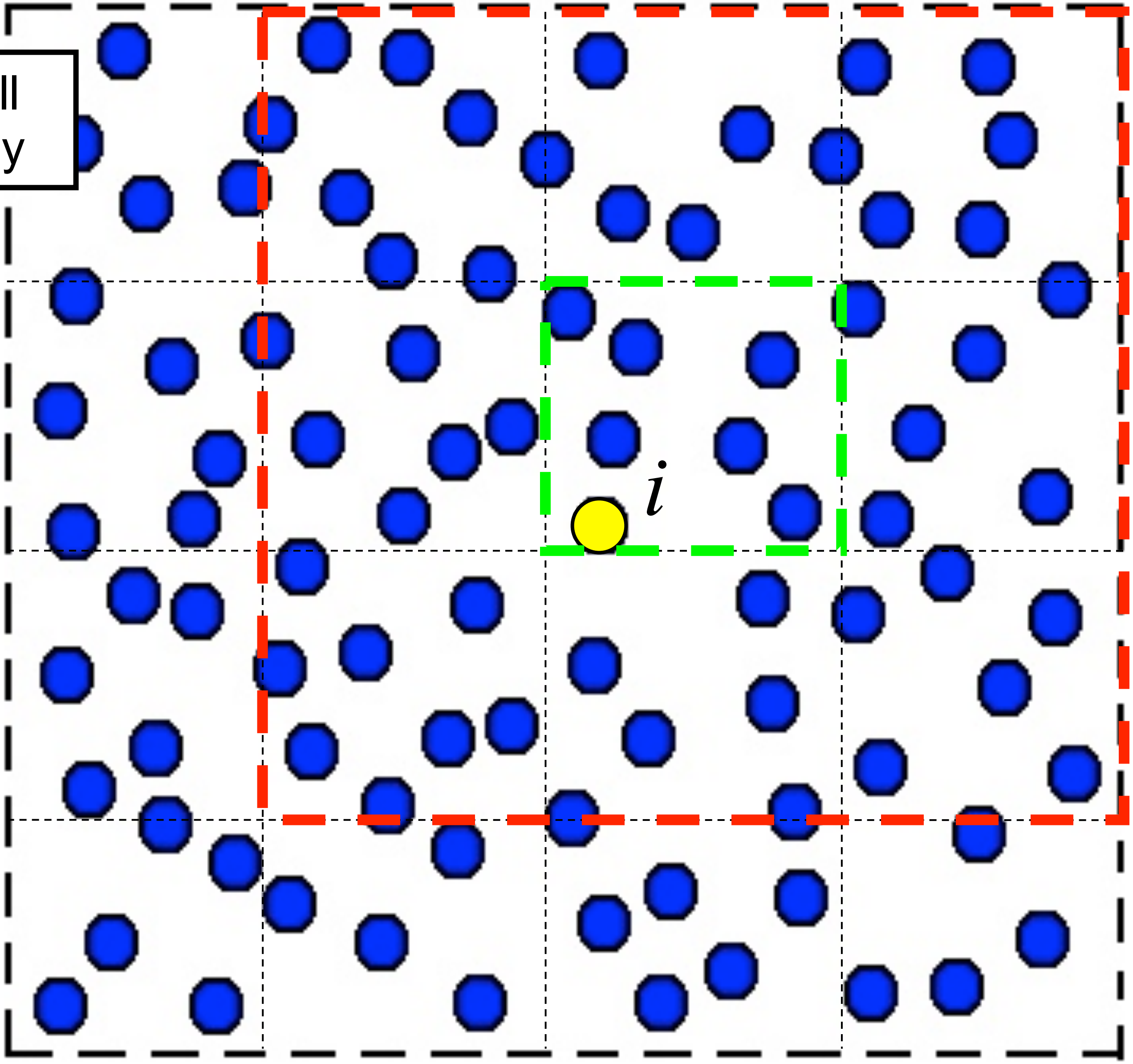
The sum is over all neighbours within the interaction distance. But, which are they?

⇩

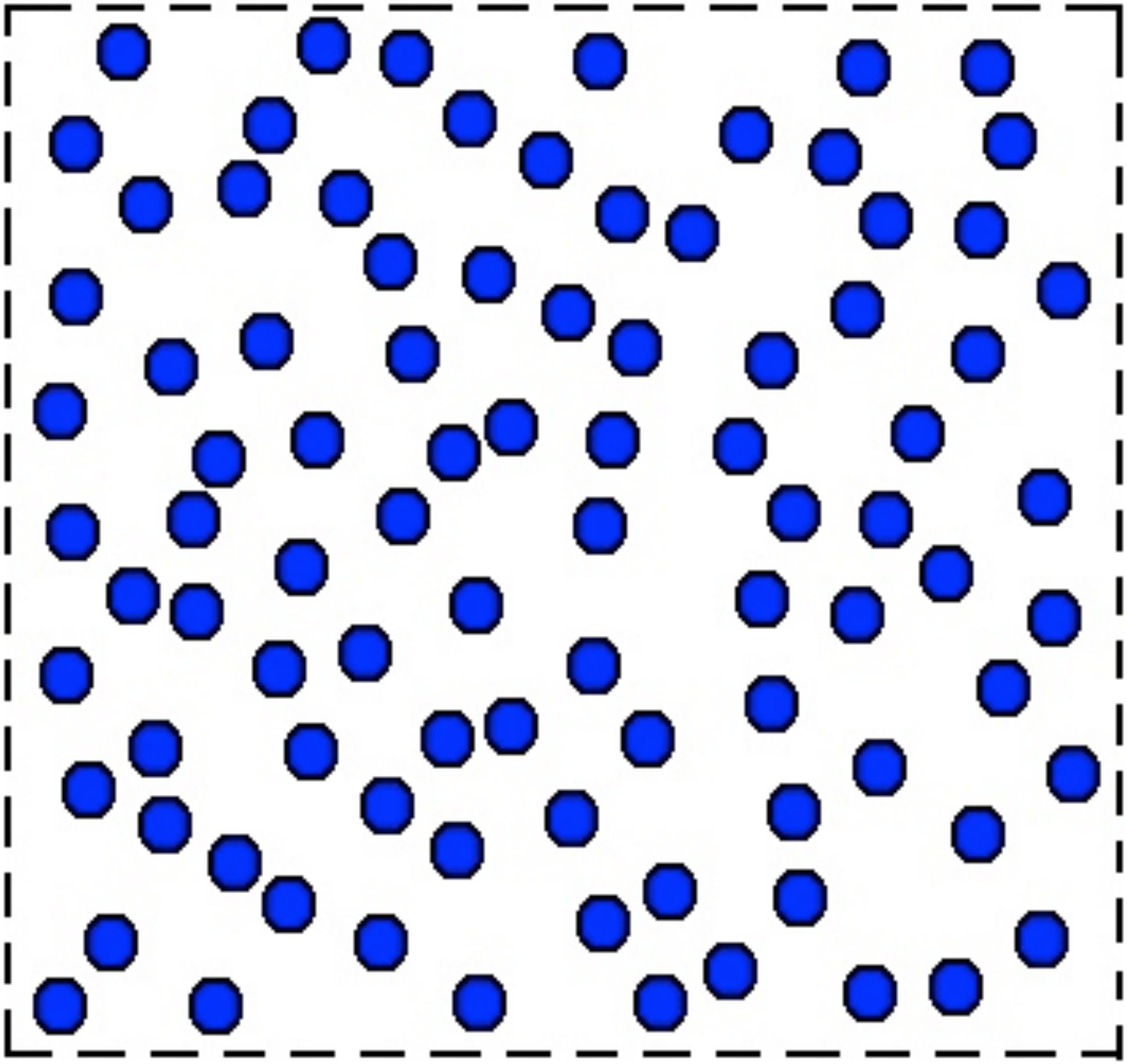
Identify all distinct pairs



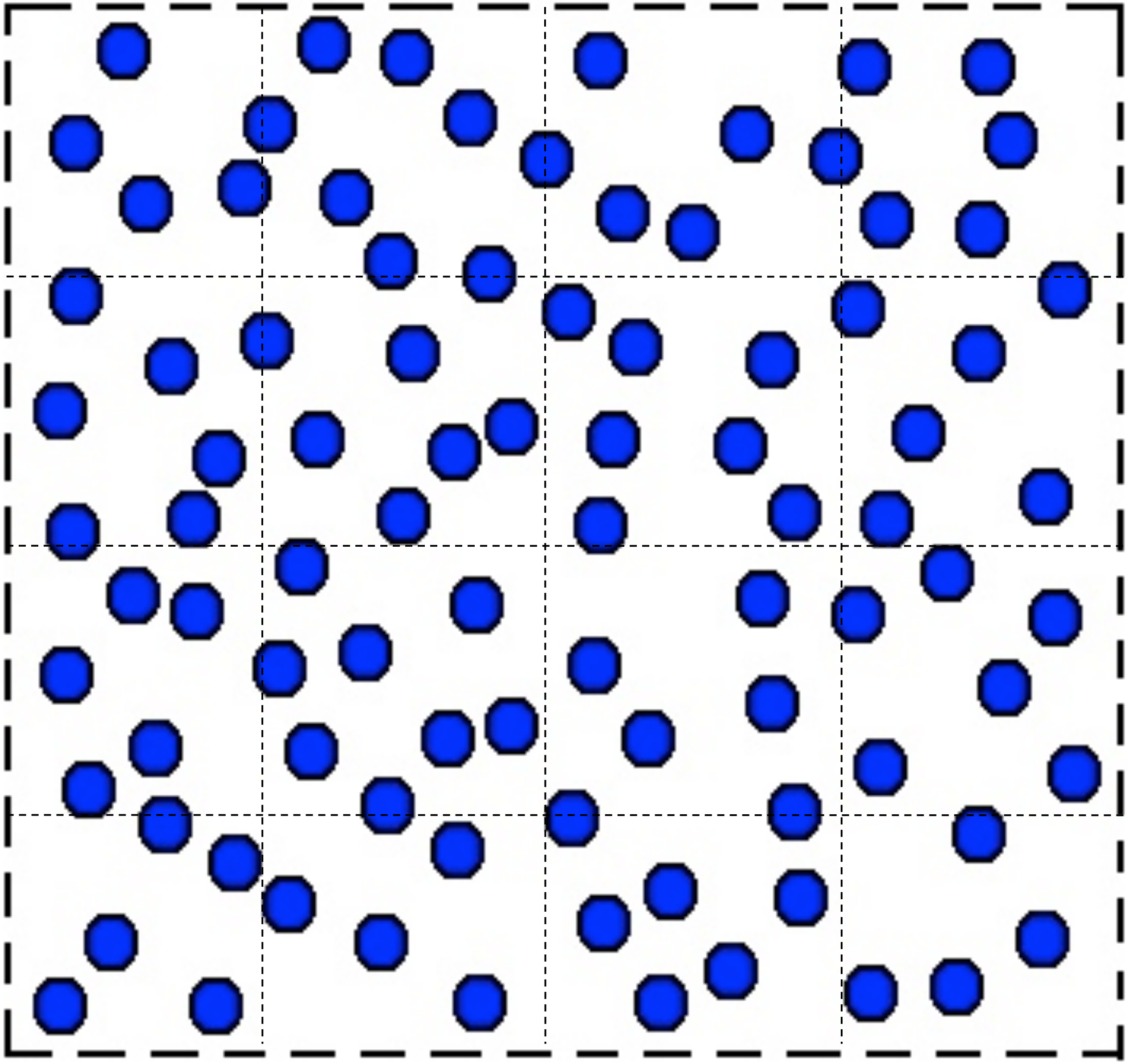
only a few neighbours out of many particles link-cell strategy



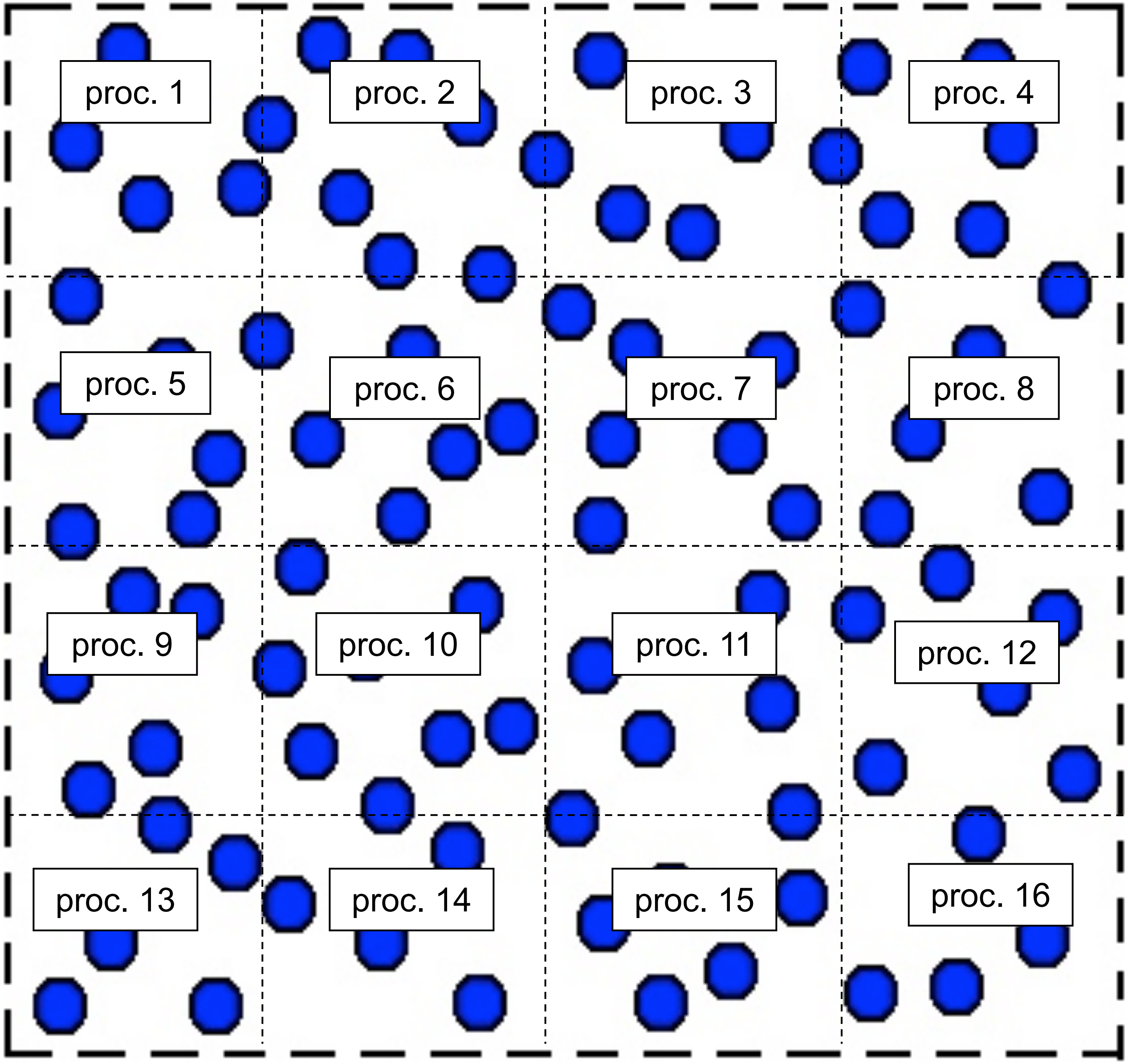
parallelisation: domain decomposition



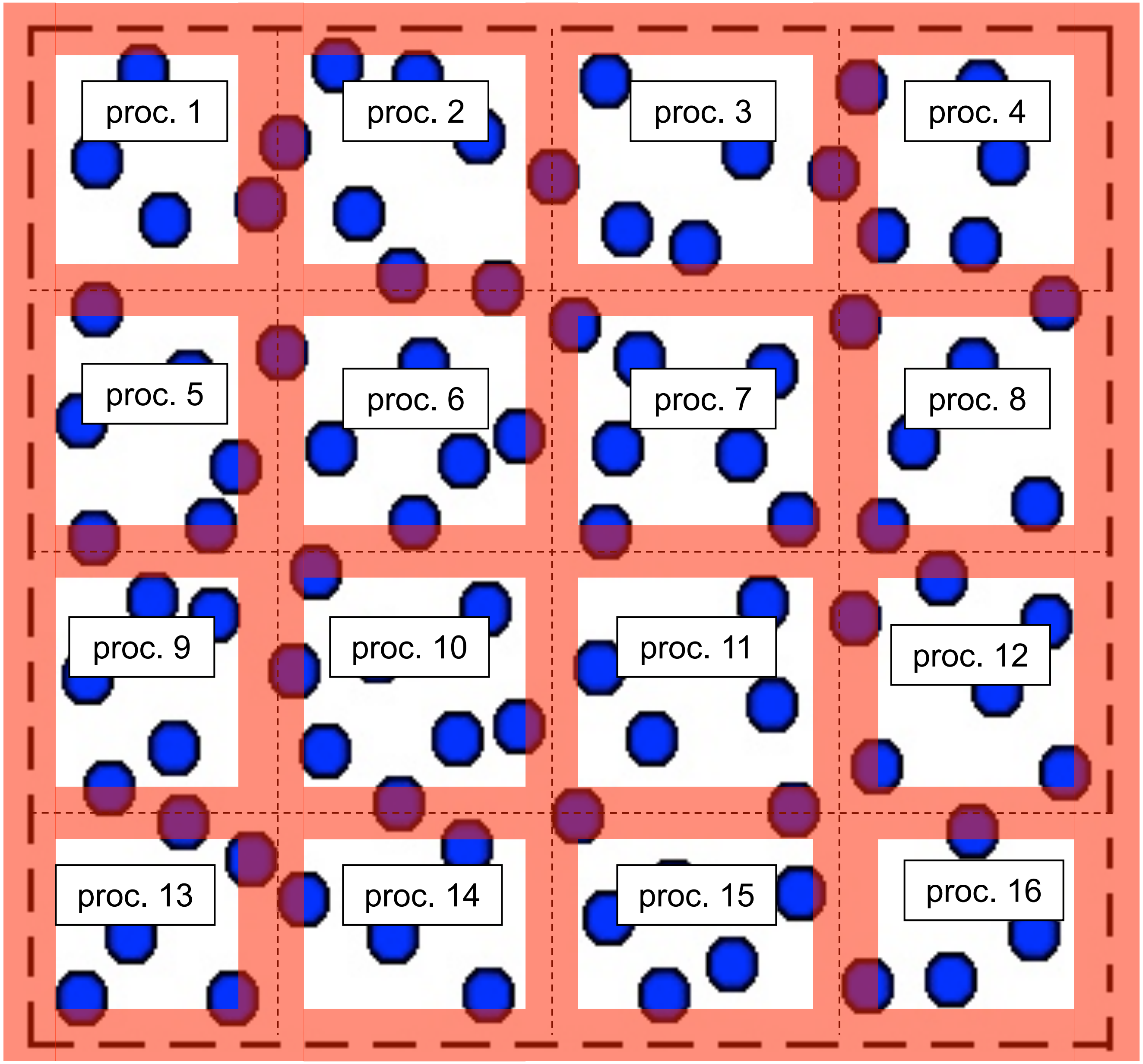
parallelisation: domain decomposition



parallelisation: domain decomposition



parallelisation: domain decomposition



- **GPU (Graphics Processing Unit)**

Specialised processor for graphics rendering

(multithreaded, data parallel co-processor)

GPGPU: General purpose computing on GPU

GeForce 8800 GTX (128 cores)

Tesla C1060 (240 cores)

- **CUDA (Compute Unified Device Architecture)**

Arquitectura para la programación en paralelo de la empresa NVIDIA

Permite acceder a la GPU para calcular como si fuera una CPU

Programables inicialmente en OpenGL (orientado a gráficos)

CUDA C: lenguaje C extendido

Actualmente también admite códigos en C++, FORTRAN, Java…

Página web de CUDA en NVIDIA:

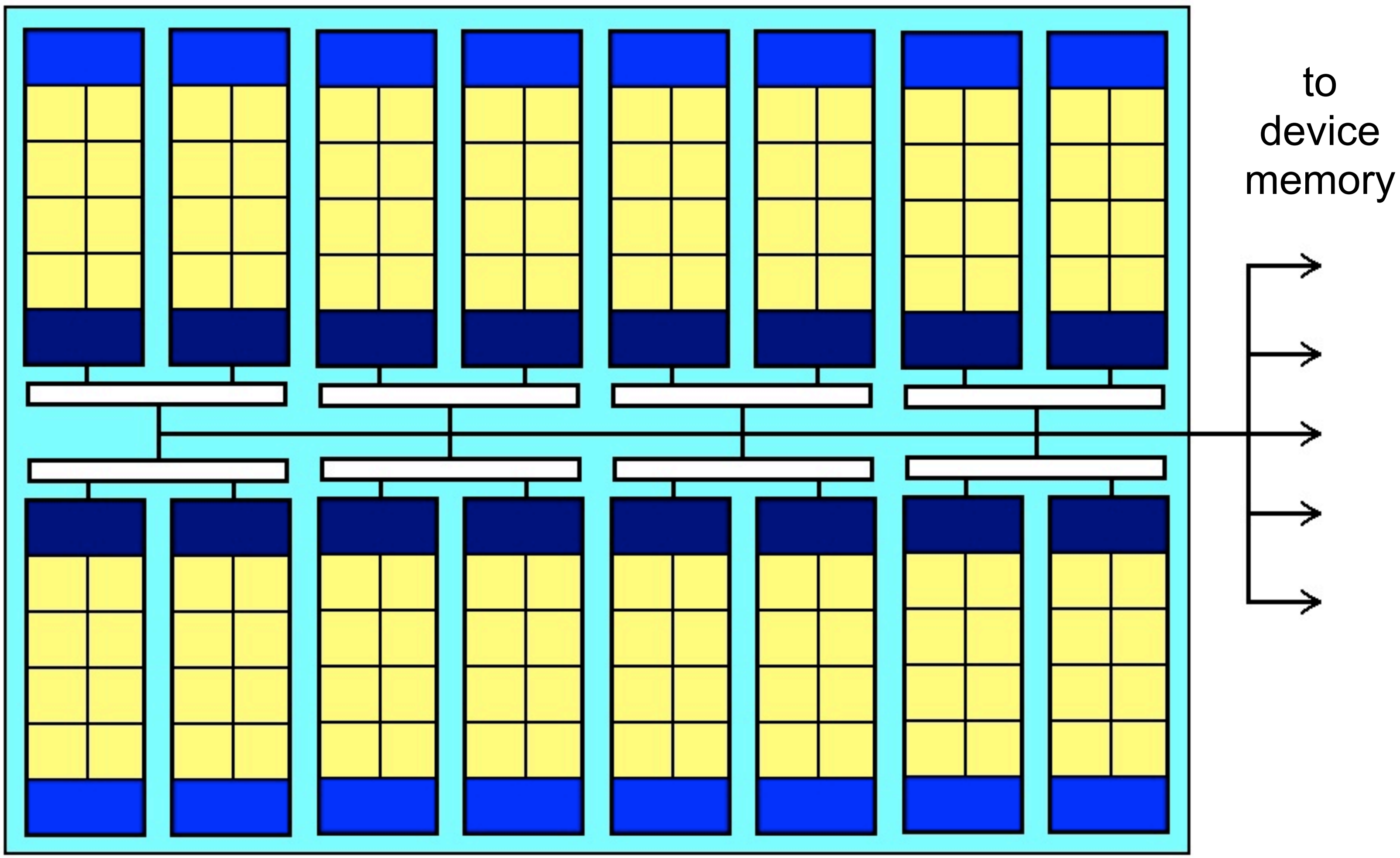
`http://www.nvidia/com/object/cuda_home.html`

Curso de programación en CUDA en UIUC:

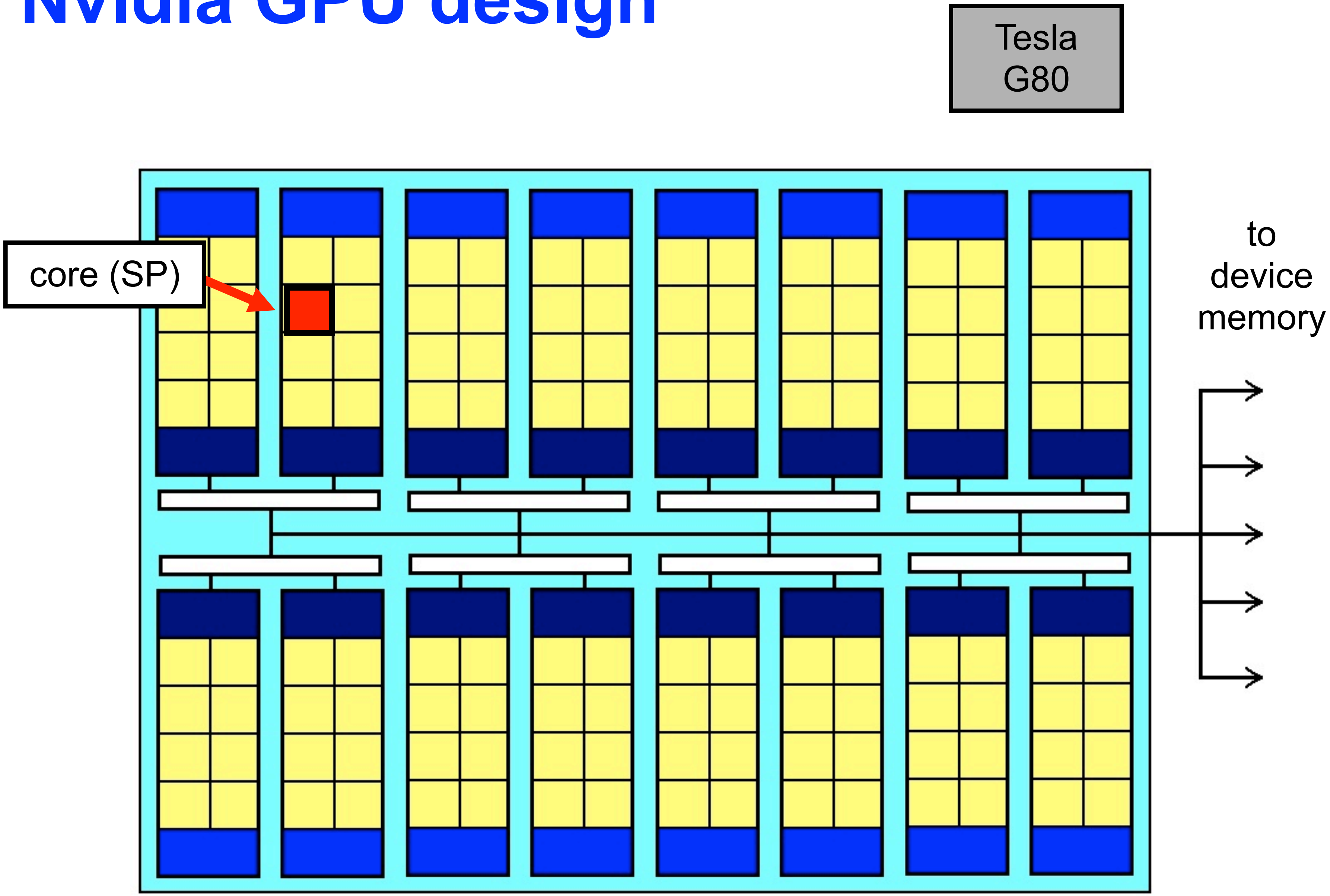
`http://courses.ece.illinois.edu/ece498/al`

Nvidia GPU design

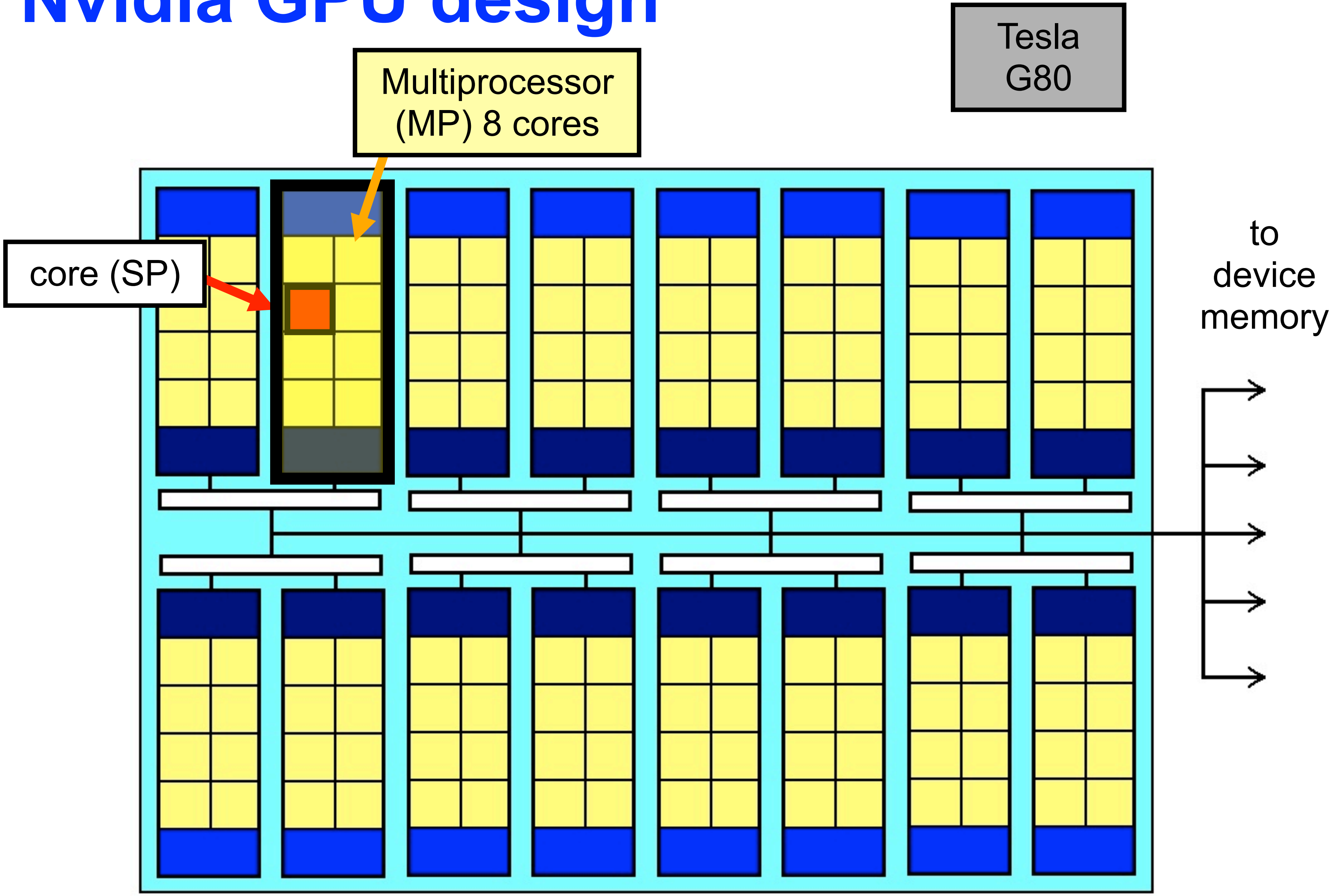
Tesla
G80



Nvidia GPU design

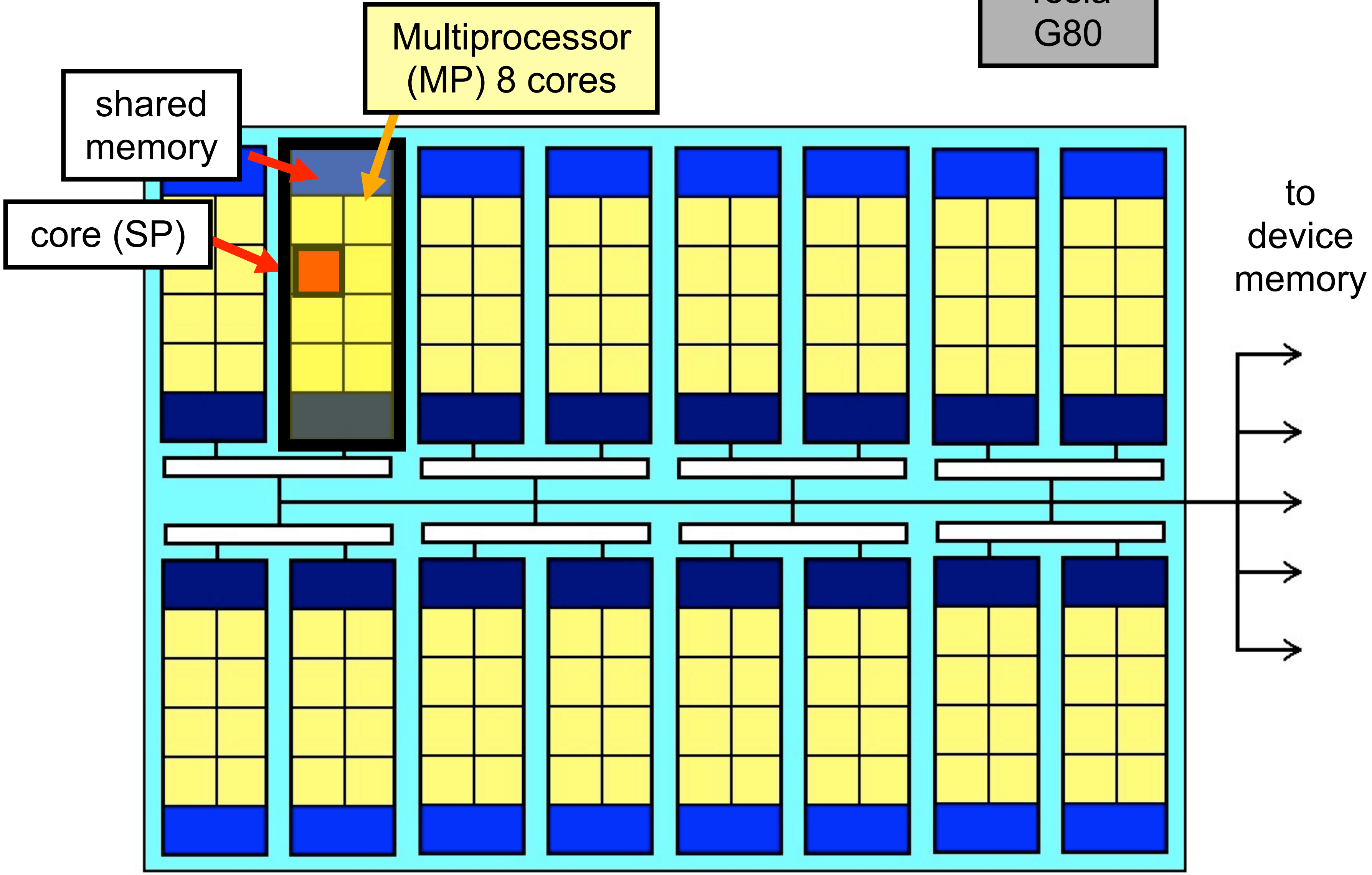


Nvidia GPU design



Nvidia GPU design

Tesla
G80



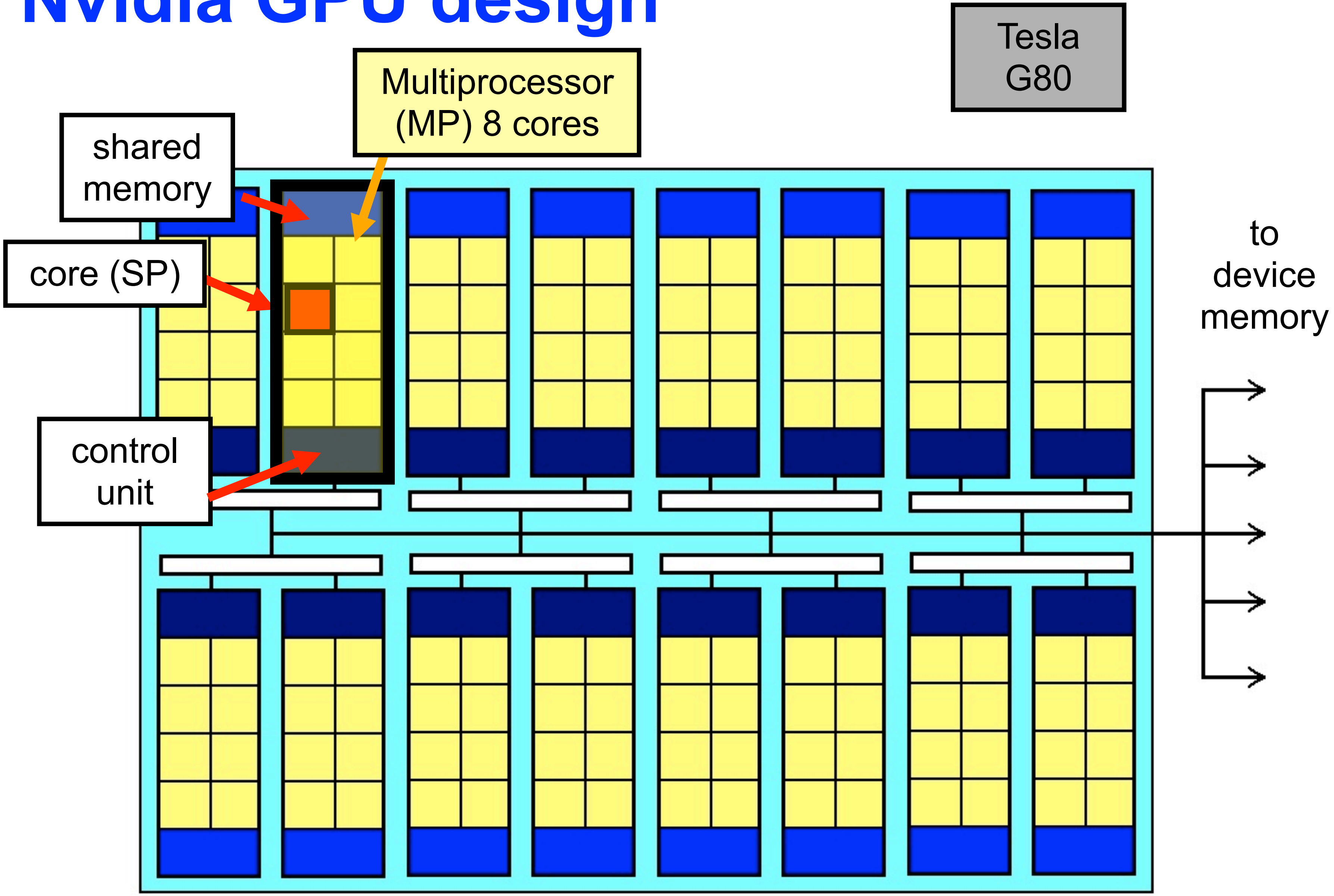
Multiprocessor
(MP) 8 cores

shared
memory

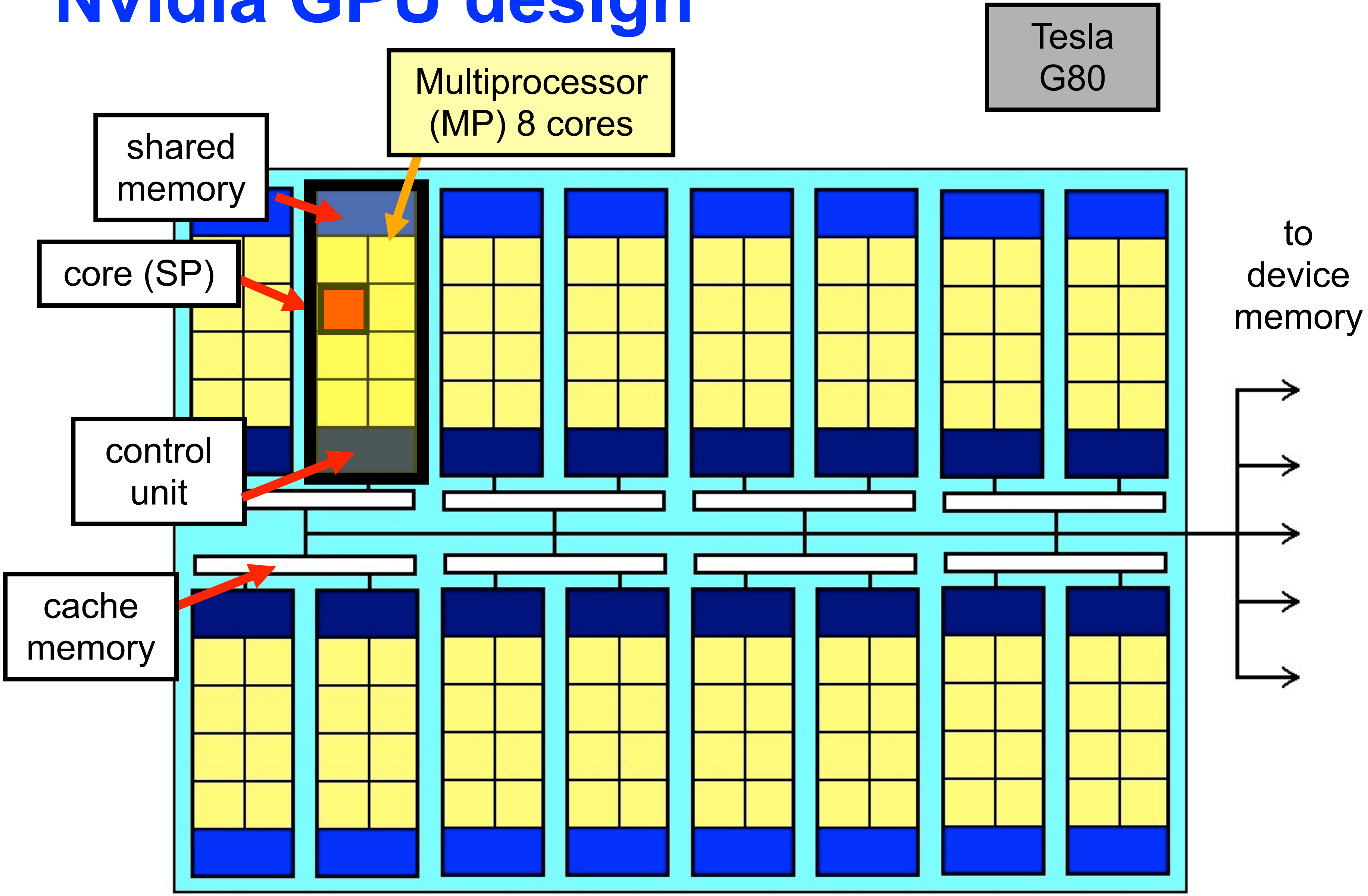
core (SP)

to
device
memory

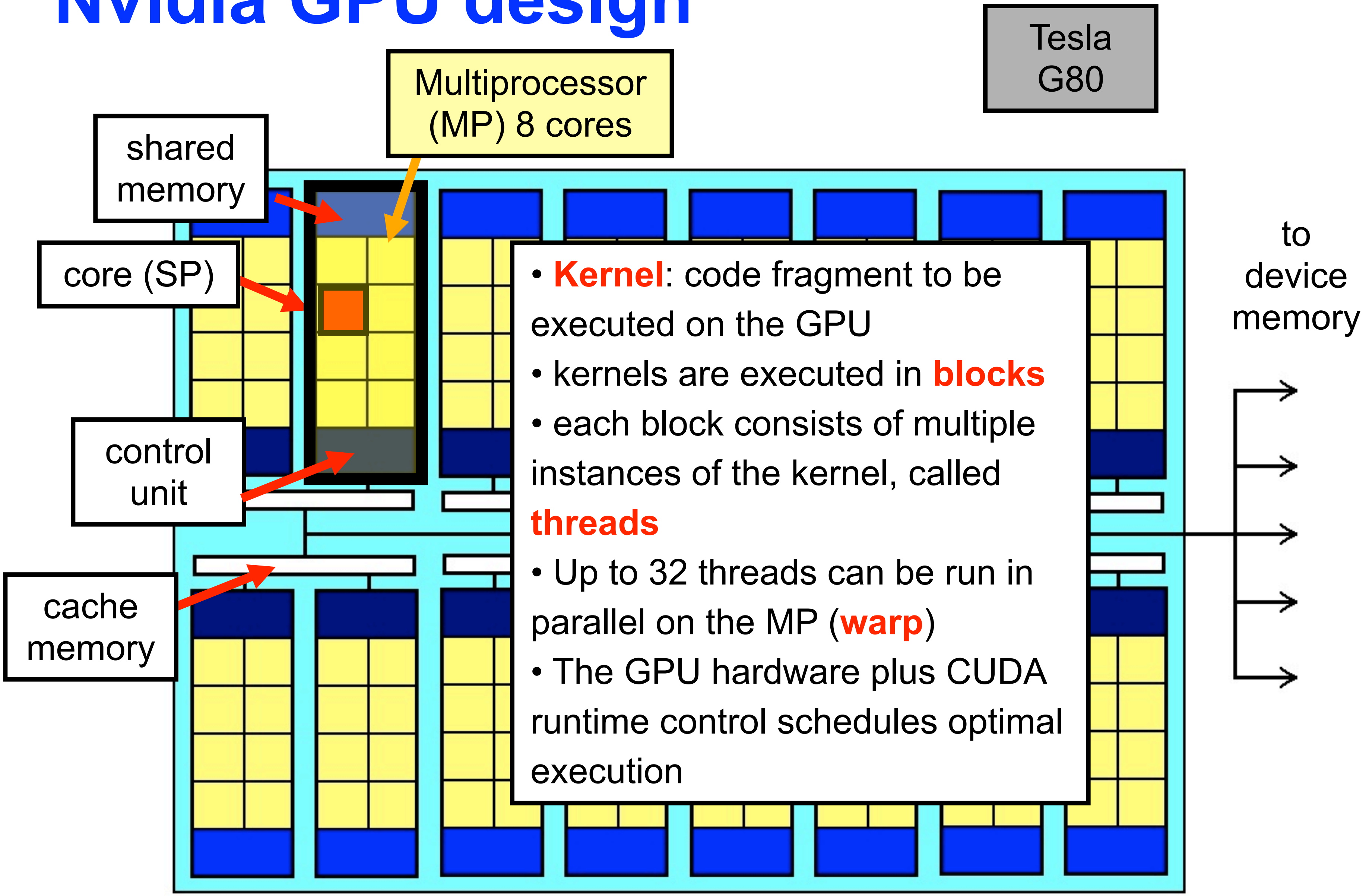
Nvidia GPU design



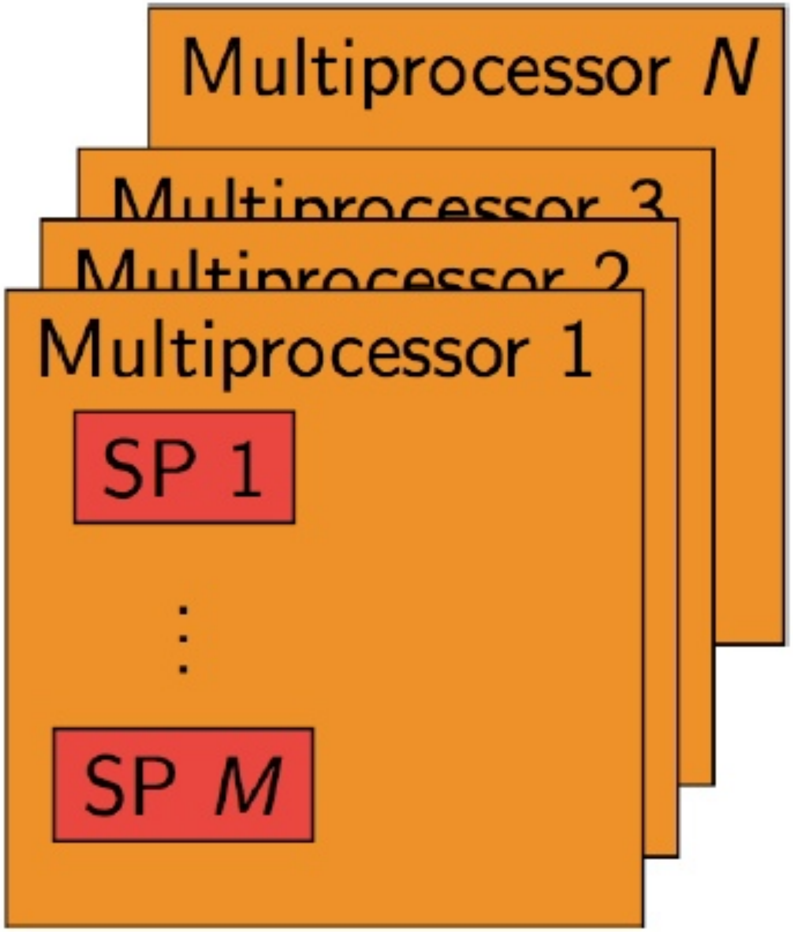
Nvidia GPU design



Nvidia GPU design



- **Kernel**: code fragment to be executed on the GPU
- kernels are executed in **blocks**
- each block consists of multiple instances of the kernel, called **threads**
- Up to 32 threads can be run in parallel on the MP (**warp**)
- The GPU hardware plus CUDA runtime control schedules optimal execution



GeForce GTX580

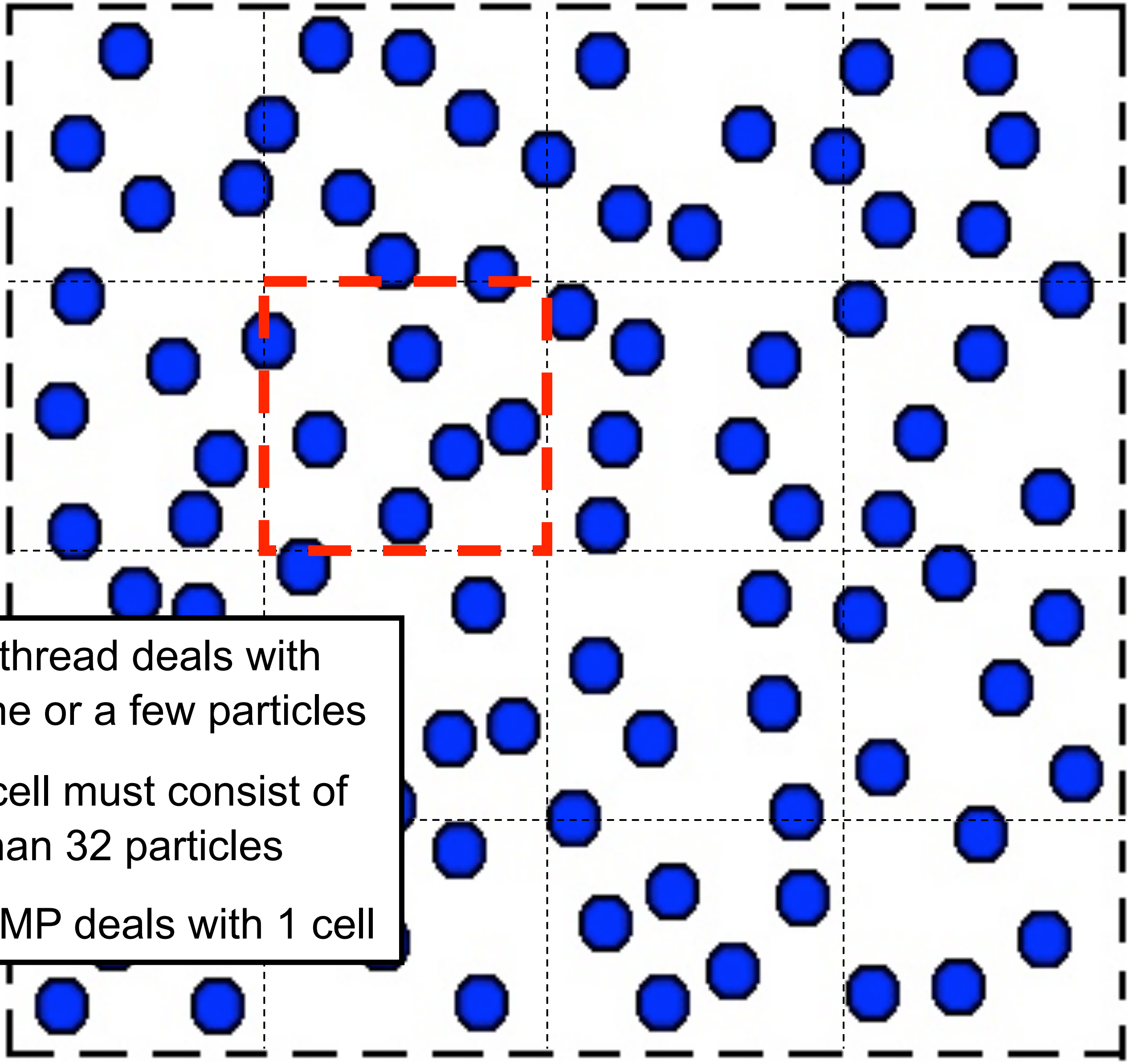
$N = 64$

$M = 8$

512 CUDA SP cores

64k shared memory

Teraflop performance (10^{12} floating point operations per second)

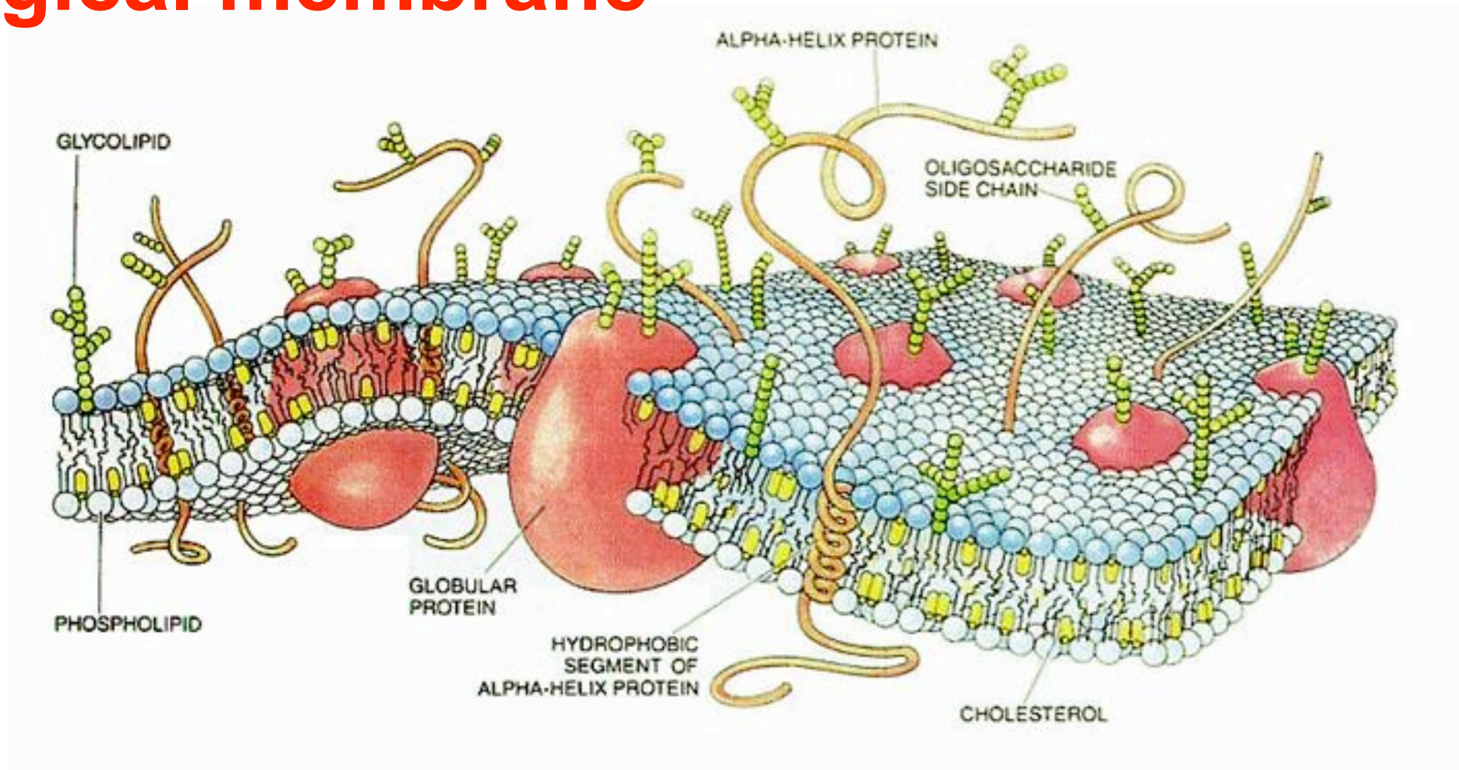


every thread deals with just one or a few particles

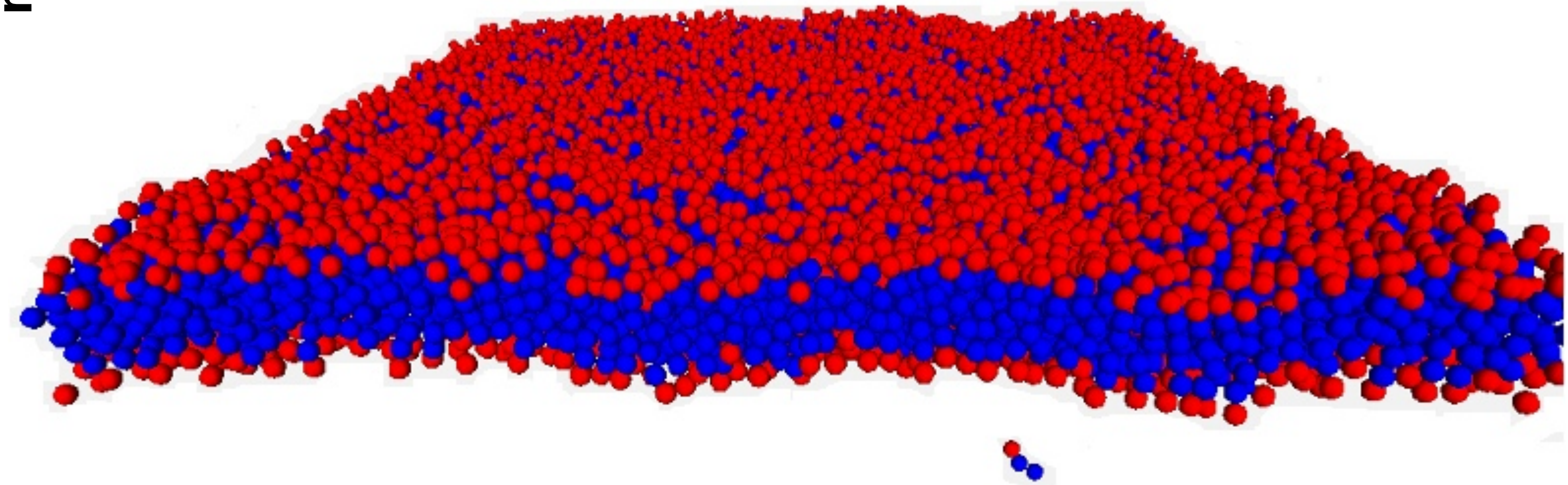
each cell must consist of less than 32 particles

every MP deals with 1 cell

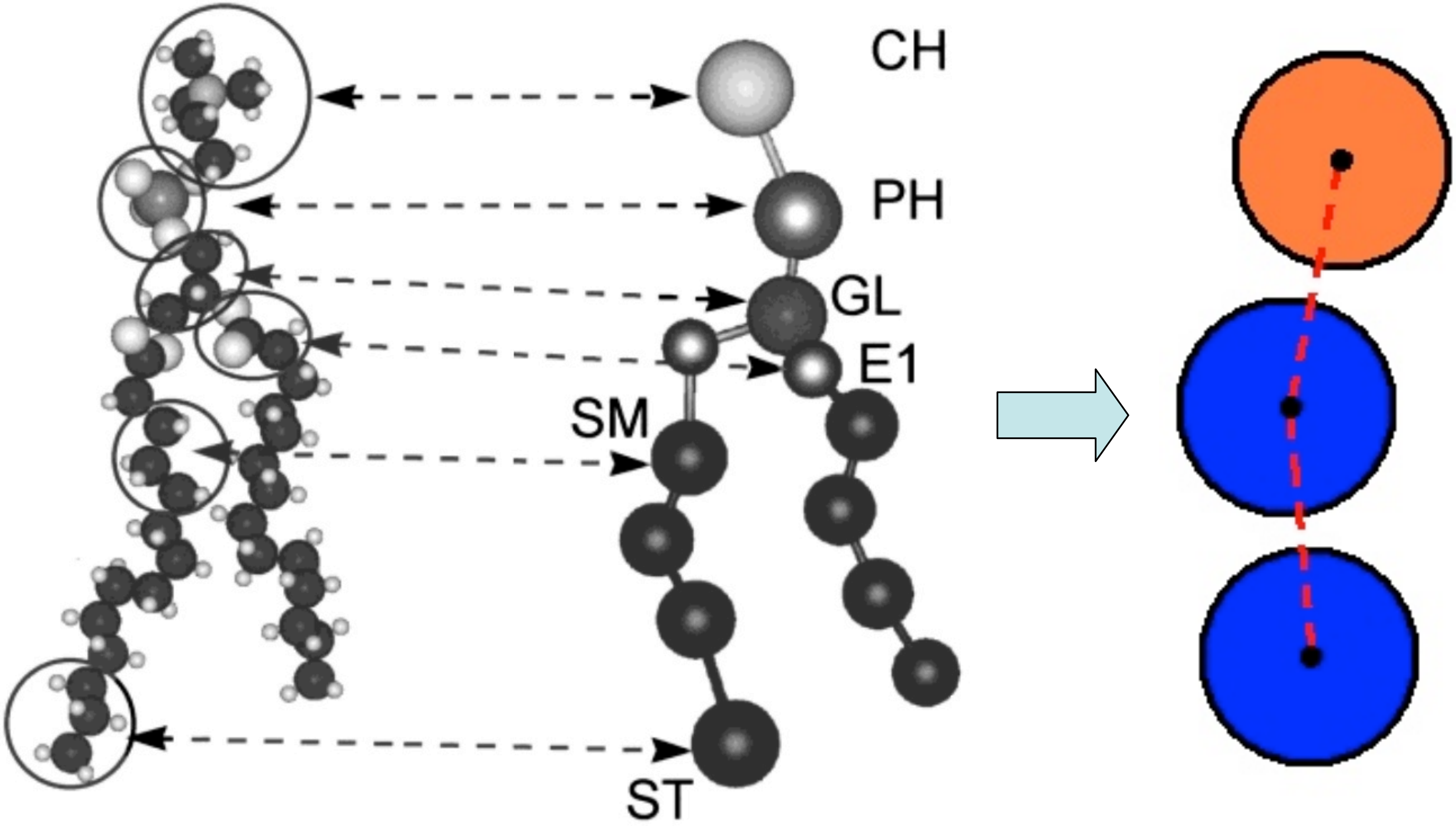
Biological membrane


ALPHA-HELIX PROTEIN
GLYCOLIPID
OLIGOSACCHARIDE SIDE CHAIN
GLOBULAR PROTEIN
PHOSPHOLIPID
HYDROPHOBIC SEGMENT OF ALPHA-HELIX PROTEIN
CHOLESTEROL

mesoscopic model

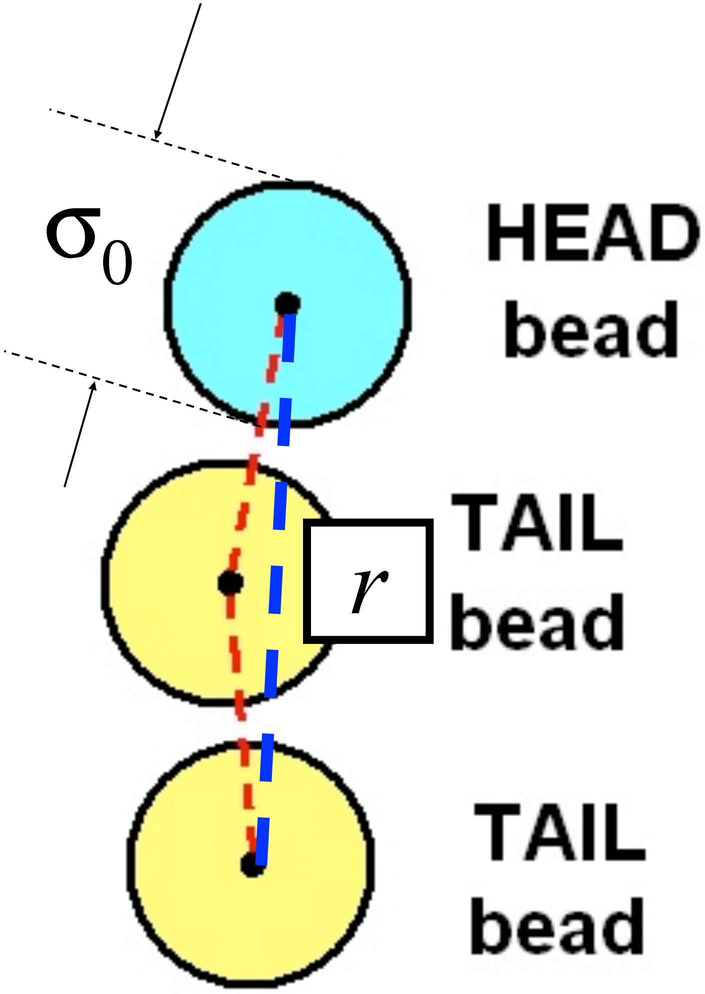


Coarse-grained molecular model



Molecular model

Three beads: one head, two tail, no solvent (water)



INTRAMOLECULAR:

- **bend**

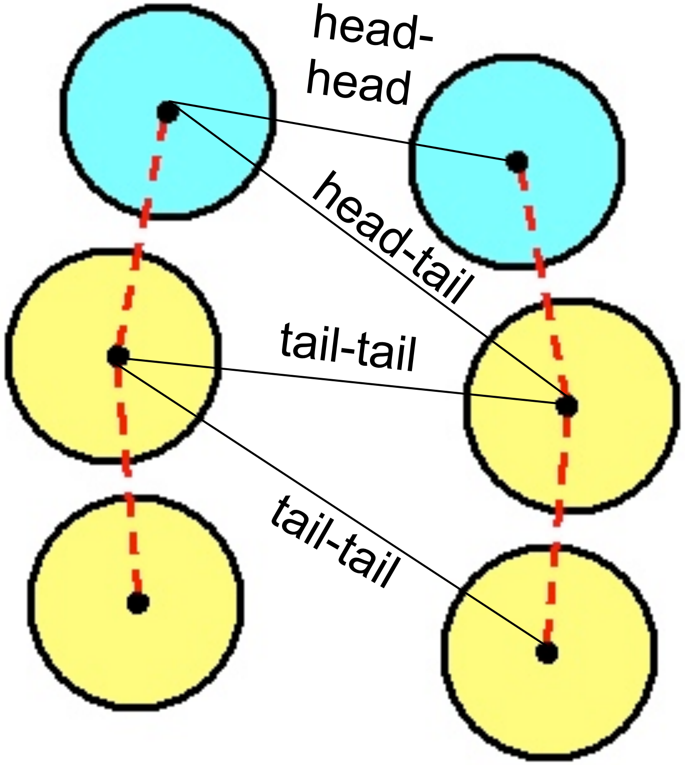
$$V_{bend}(r) = \frac{1}{2} k_{bend} (r - 3\sigma_0)^2$$

$$k_{bend}\sigma_0^2 = 10\varepsilon_0$$

- **bond**

$$V_{bond}(r) = -\frac{1}{2} k_{bond} r_\infty^2 \log\left[1 - (r / r_\infty)^2\right]$$

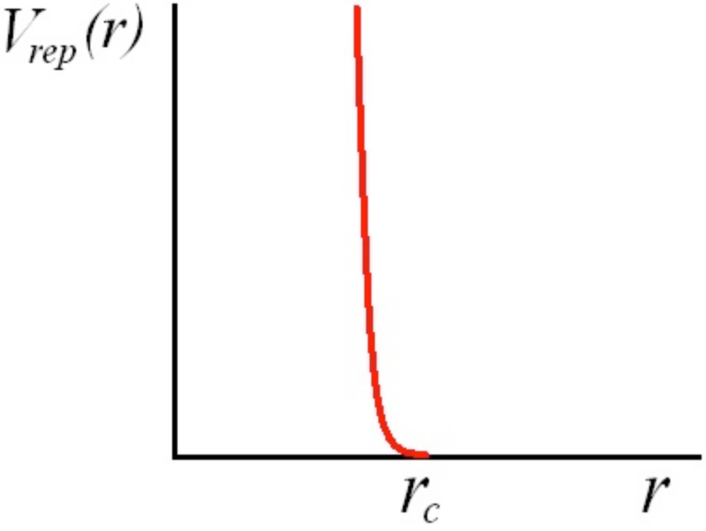
$$k_{bond}\sigma_0^2 = 30\varepsilon_0, \; r_\infty = 1.5\sigma_0$$



INTERMOLECULAR:

all beads of different molecules interact via the Weeks-Chandler-Andersen potential:

$$V_{rep}(r) = \begin{cases} 4\varepsilon_0\left[\left(\frac{b}{r}\right)^{12} - \left(\frac{b}{r}\right)^{6}\right] + \varepsilon_0, & r \le r_c, \\ 0, & r > r_{c.} \end{cases}$$



ε_0 is the unit of energy

$b_{head\text{-}head} = 0.95\ \sigma_0$

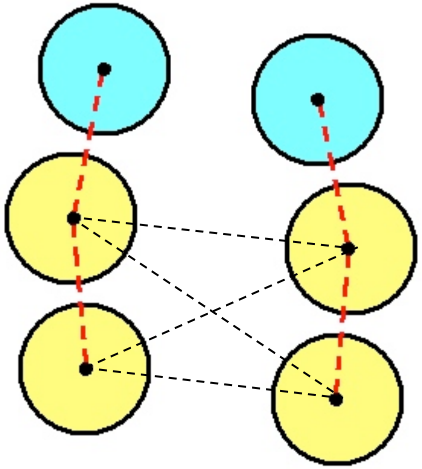
$b_{hear\text{-}tail} = 0.95\ \sigma_0$

$b_{tail\text{-}tail} = \sigma_0$

$$r_c = 2^{1/6}\sigma_0$$

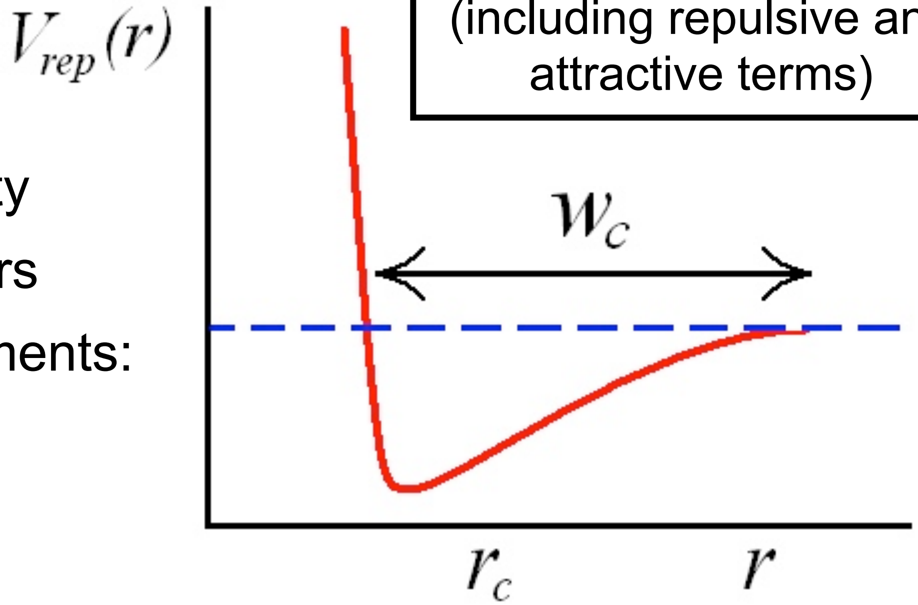
PLUS tail attraction (to mimic hydrophobic effect):

$$V_{attr}(r) = \begin{cases} -\varepsilon, & r < r_c, \\ -\varepsilon_0 \cos^2\left[\dfrac{\pi(r-r_c)}{2w_c}\right], & r_c \le r \le r_c + w_c, \\ 0, & r > r_c + w_c. \end{cases}$$



Advantages of model:

- Broad range of membrane fluidity
- Easily tunable via few parameters
- Good agreement with measurements: rigidity, diffusion, density



Total tail-tail potential (including repulsive and attractive terms)

Features of our simulations

- $N = 7164 - 28800$ molecules (x 3 beads)

- $\tau = 2 - 3$ ns

- NPT ensemble (constant pressure = constant $\gamma = 0$)
(fluctuating system volume)

- Langevin thermostat

- To model typical phospholipid:

$d = 10$ nm (membrane thickness) $\longrightarrow$ $\sigma_0 = 1.7$ nm

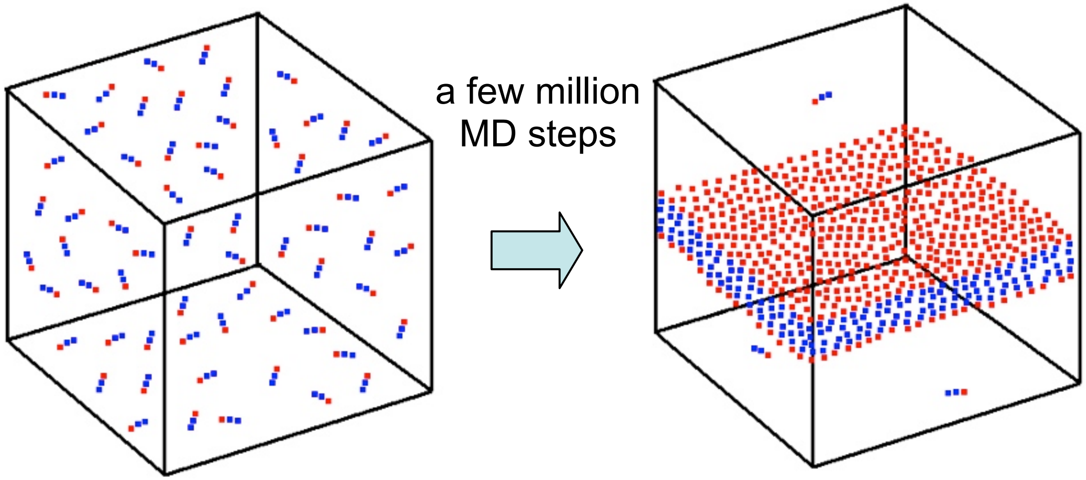
$\varepsilon_0 = 3.72 \times 10^{-21}$ J

$m_0 = 220$ g/mol

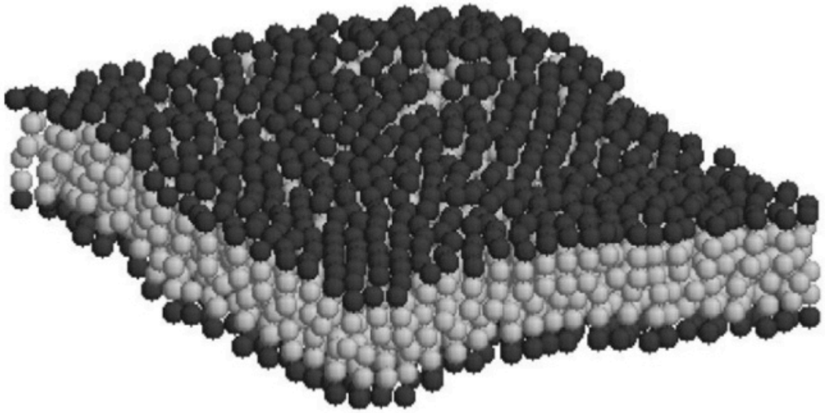
$\rightarrow$ $\tau_0 = \sigma_0 \sqrt{\dfrac{m_0}{\varepsilon_0}} = 9.85$ ps (time scale of MD simulation)

molecules self-assemble spontaneously into planar membranes

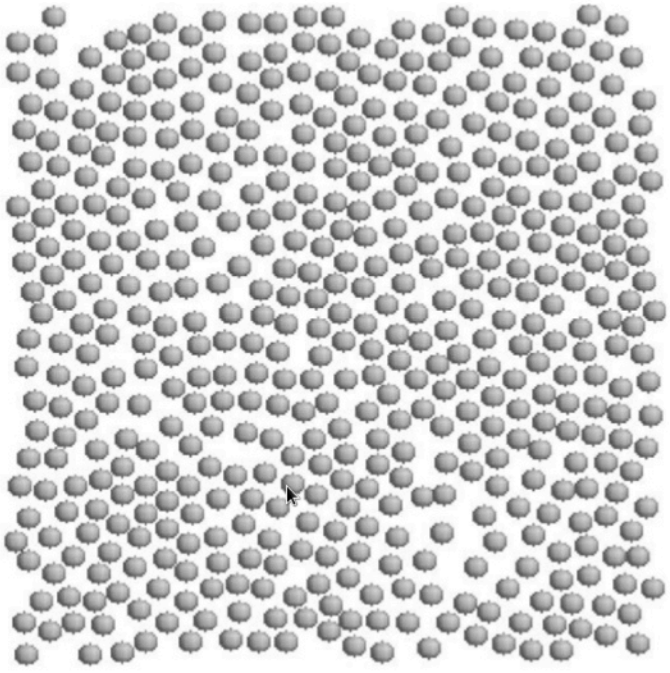
special MD techniques (ensuring $\gamma=0$) are needed (fluctuating box)



side view

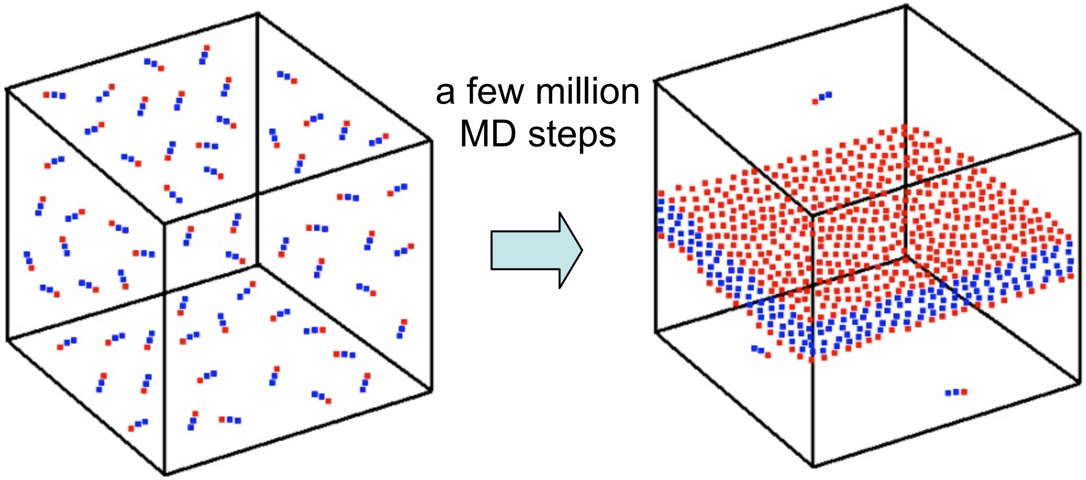


top view

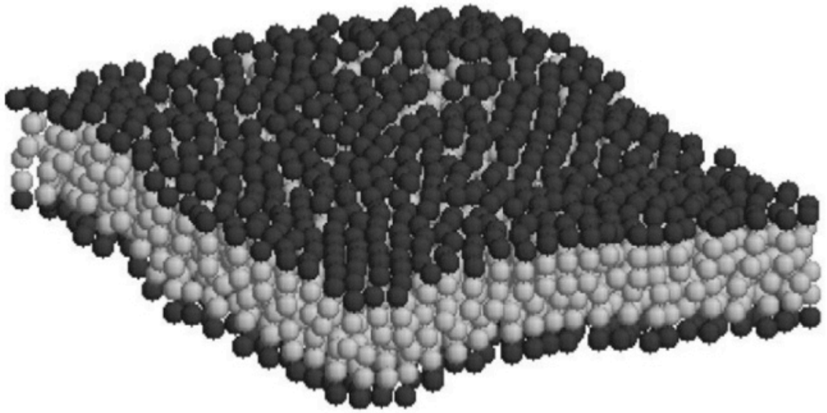


molecules self-assemble spontaneously into planar membranes

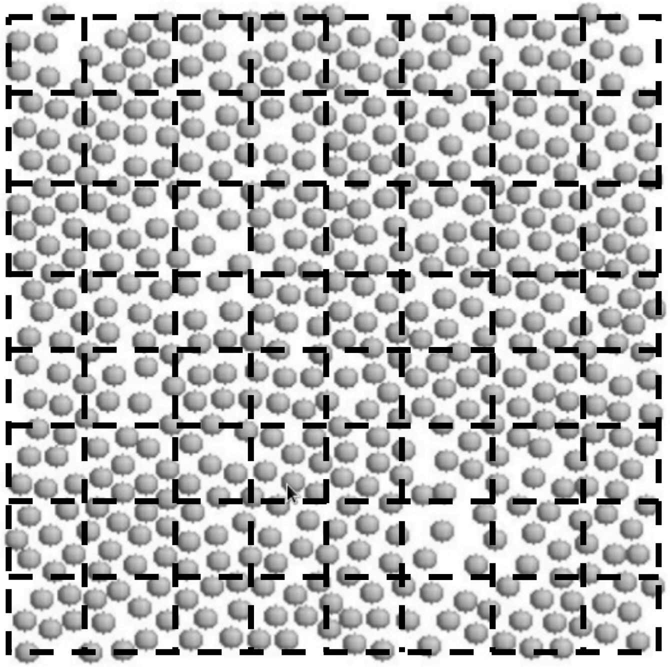
special MD techniques ($\gamma=0$) are needed (fluctuating box)



side view

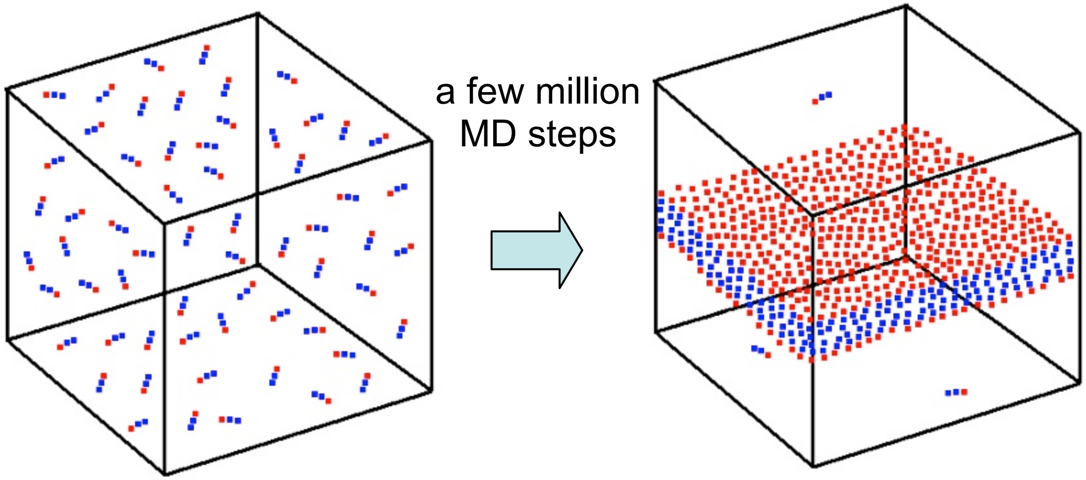


top view

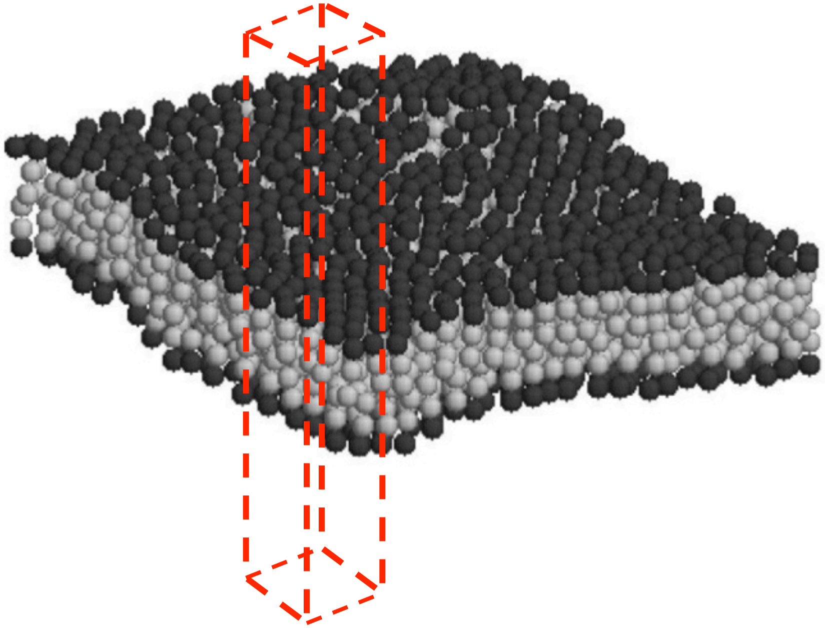


molecules self-assemble spontaneously into planar membranes

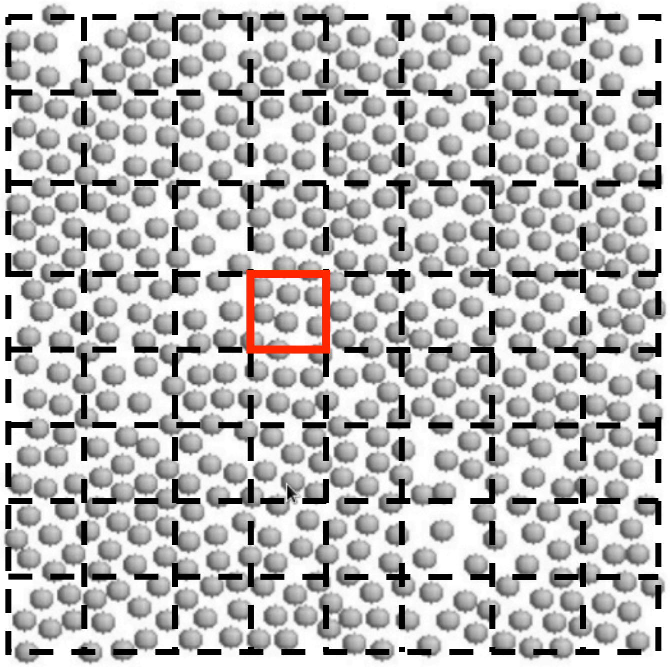
special MD techniques (ensuring $\gamma=0$) are needed (fluctuating box)



side view



top view



Properties of a mathematical membrane

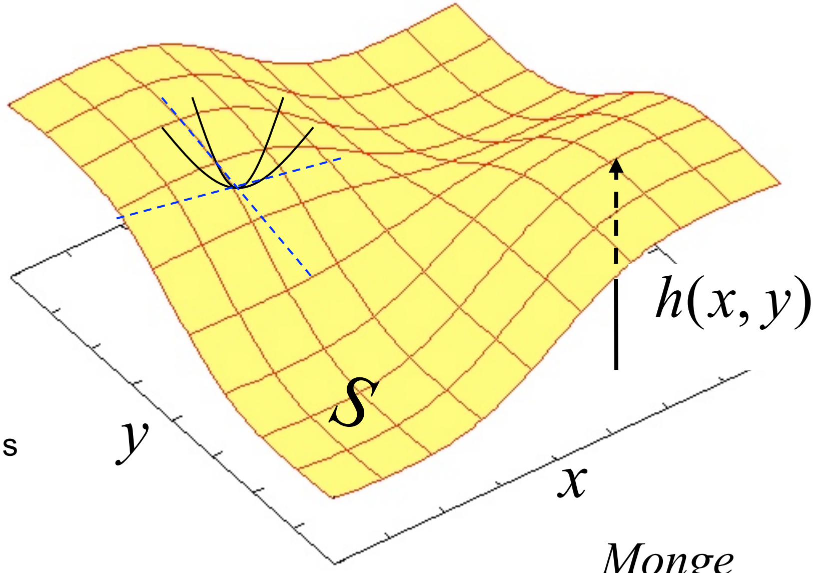
Fluctuating sheet of zero thickness

Described by Helfrich free energy:

$$F = \int_S dA \left[\gamma + \frac{\kappa}{2} \big(\underbrace{c_1 + c_2} - 2c_0 \big)^2 \right]$$

spontaneous curvature

mean curvature $H = \dfrac{c_1 + c_2}{2}$



Monge representation

For our membranes c_0=0 and

$$F = \int_S dA \left(\gamma + 2\kappa H^2 \right) \longrightarrow \frac{1}{2} \iint dxdy \left[\gamma \left| \nabla_\perp h \right|^2 + \kappa \left(\nabla^2 h \right)^2 \right]$$

Properties of a mathematical membrane

Fluctuating sheet of zero thickness

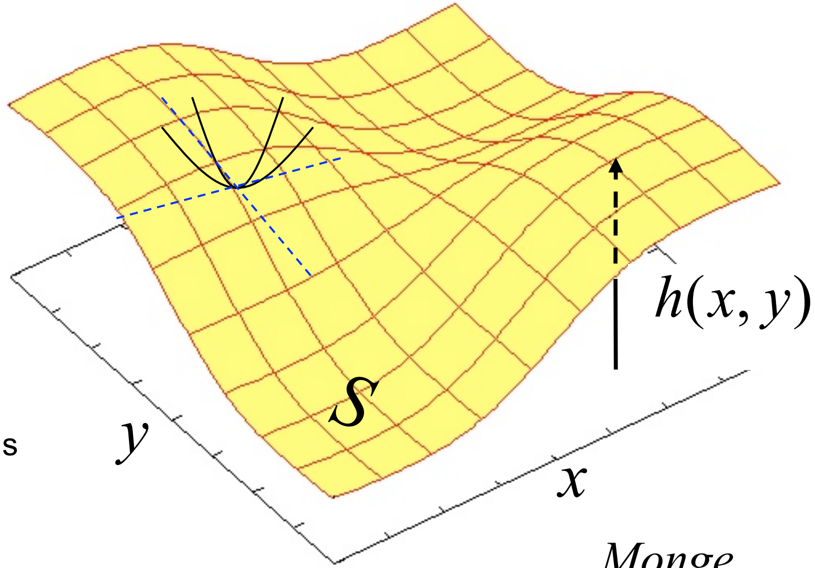
Described by Helfrich free energy:

$$F = \int_S dA\left[\gamma + \frac{\kappa}{2}\underbrace{\left(c_1 + c_2\right.}_{}\left. - 2c_0\right)^2\right]$$

surface tension

mean curvature $H = \frac{c_1 + c_2}{2}$

spontaneous curvature



Monge representation

For our membranes c_0=0 and

$$F = \int_S dA\left(\gamma + 2\kappa H^2\right) \longrightarrow \frac{1}{2}\iint dxdy\left[\gamma\left|\nabla_\perp h\right|^2 + \kappa\left(\nabla^2 h\right)^2\right]$$

Properties of a mathematical membrane

Fluctuating sheet of zero thickness

Described by Helfrich free energy:

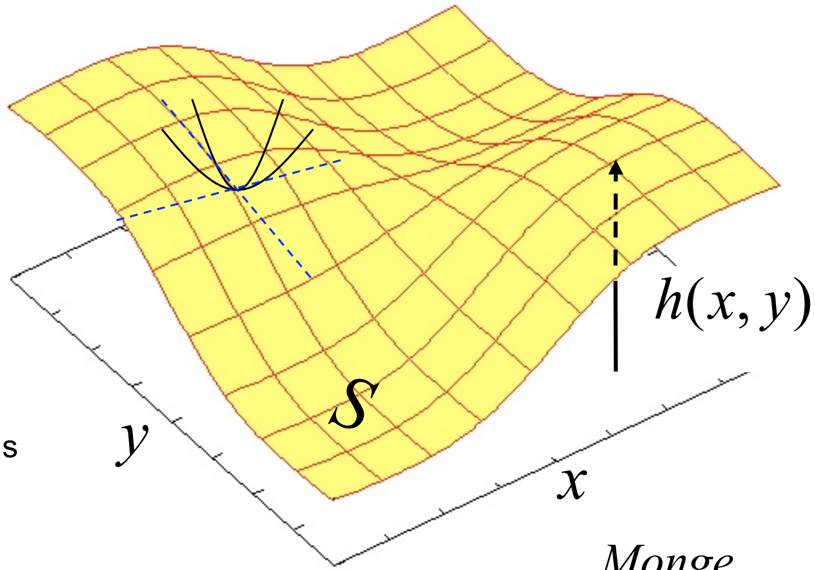
$$F = \int_S dA \left[\gamma + \frac{\kappa}{2} \left(\underbrace{c_1 + c_2}_{} - 2c_0 \right)^2 \right]$$

bending constant

surface tension

mean curvature $H = \frac{c_1 + c_2}{2}$

spontaneous curvature



Monge representation

For our membranes c_0=0 and

$$F = \int_S dA\left(\gamma + 2\kappa H^2\right) \longrightarrow \frac{1}{2}\iint dxdy\left[\gamma\left|\nabla_\perp h\right|^2 + \kappa\left(\nabla^2 h\right)^2\right]$$

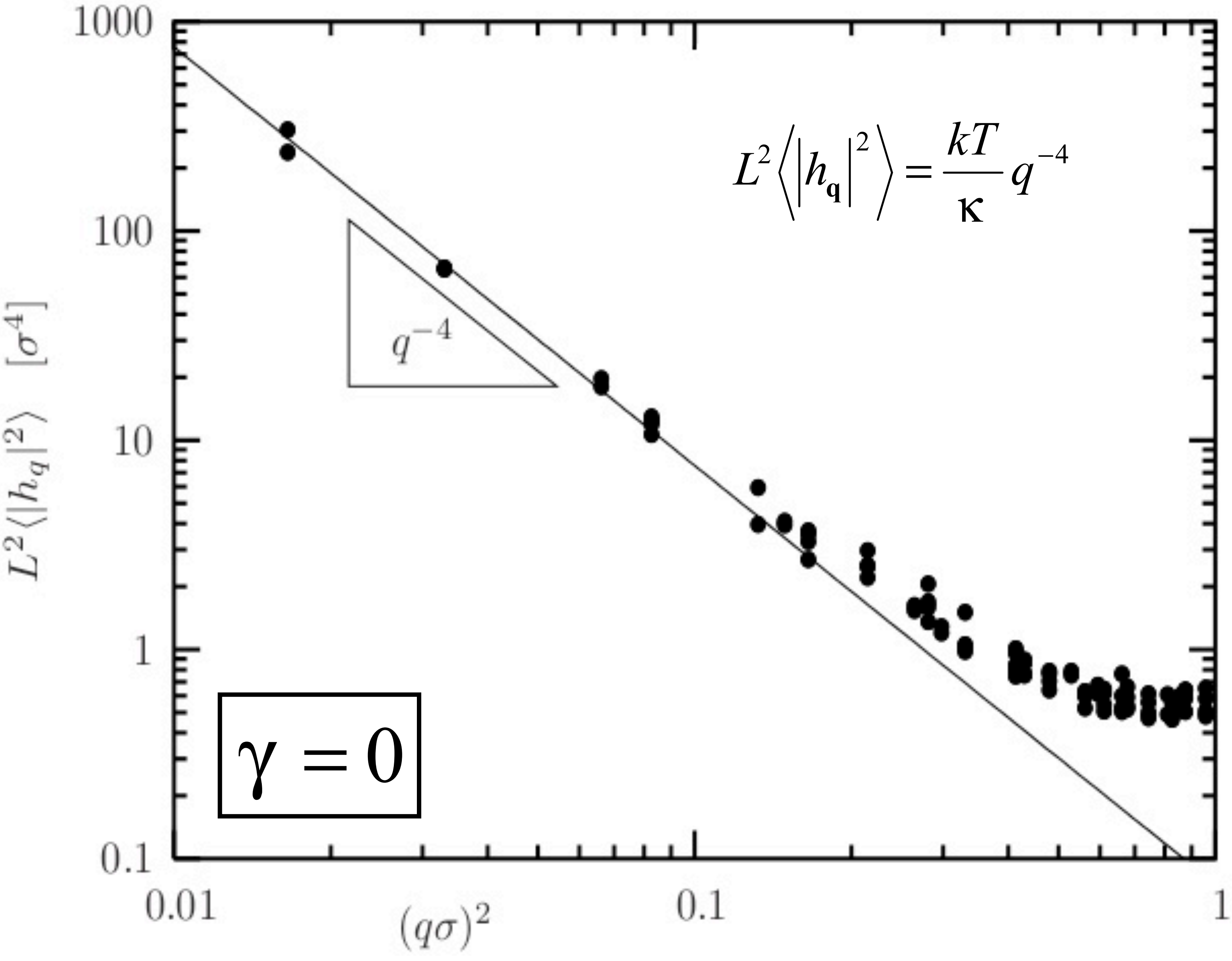
How can γ and κ be computed from a MD simulation?

Taking Fourier transform:

$$F = \frac{L^2}{2}\sum_{\mathbf{q}}\left(\gamma|\mathbf{q}|^2 + \kappa|\mathbf{q}|^4\right)|h_{\mathbf{q}}|^2$$

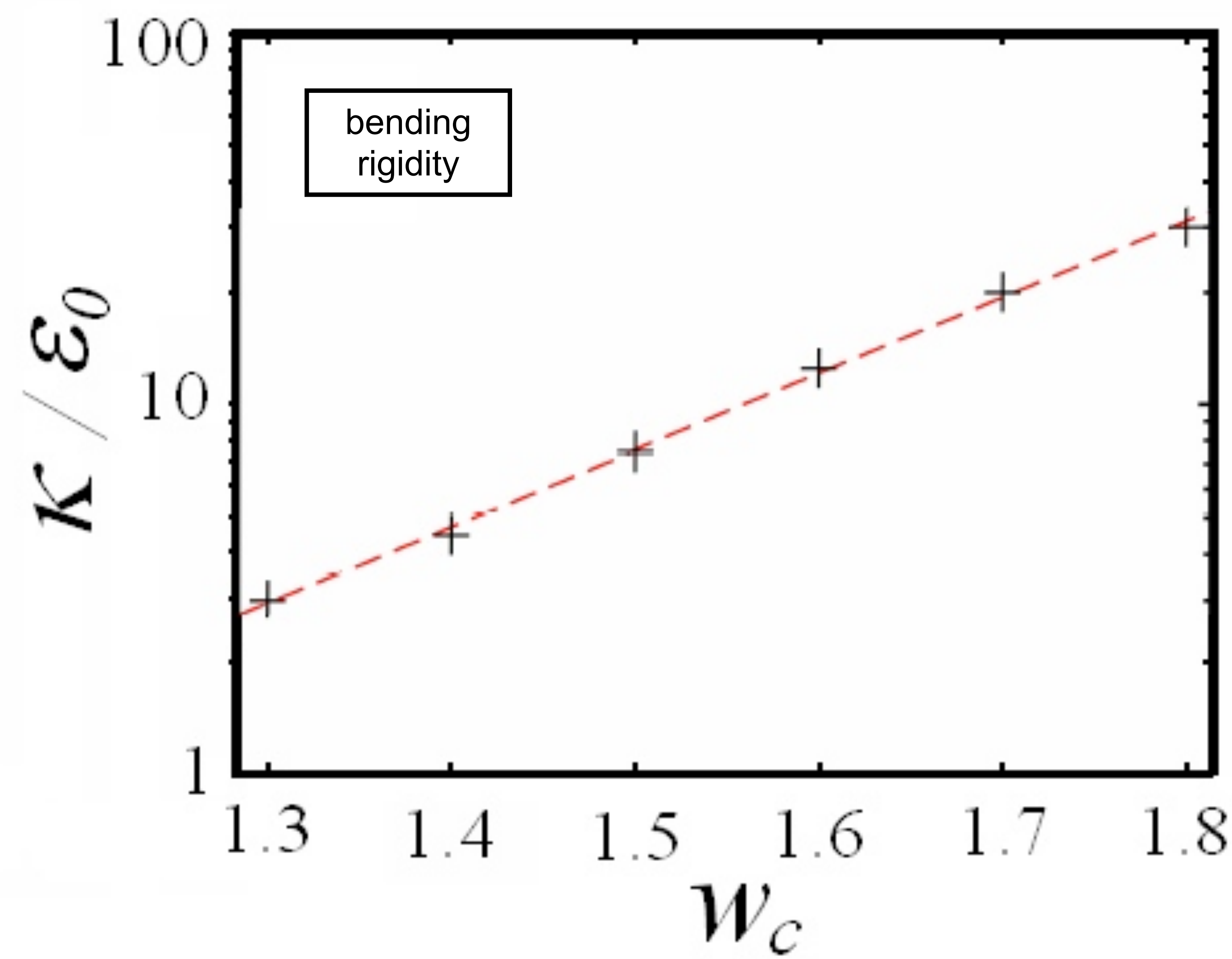
Equipartition theorem:

$$\frac{L^2}{2}\left(\gamma|\mathbf{q}|^2 + \kappa|\mathbf{q}|^4\right)\left\langle|h_{\mathbf{q}}|^2\right\rangle = \frac{kT}{2} \quad \Rightarrow \quad \boxed{L^2\left\langle|h_{\mathbf{q}}|^2\right\rangle = \frac{kT}{\gamma|\mathbf{q}|^2 + \kappa|\mathbf{q}|^4}}$$



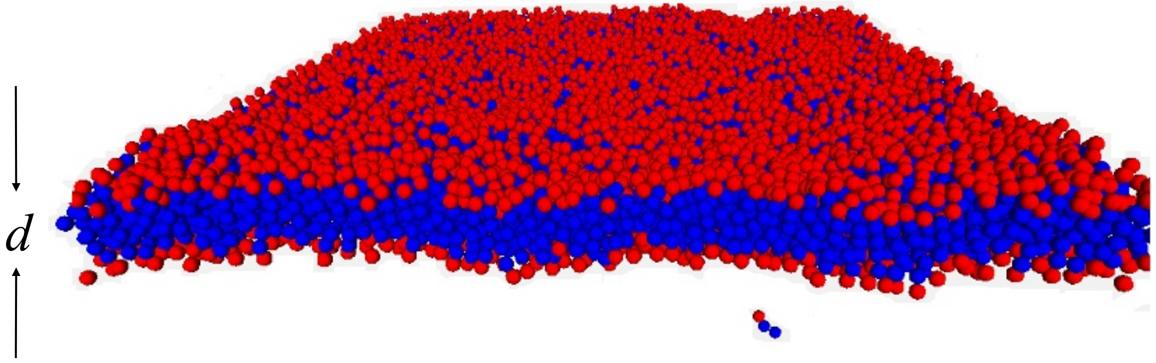
$$L^2\left\langle\left|h_{\mathbf{q}}\right|^2\right\rangle = \frac{kT}{\kappa}q^{-4}$$

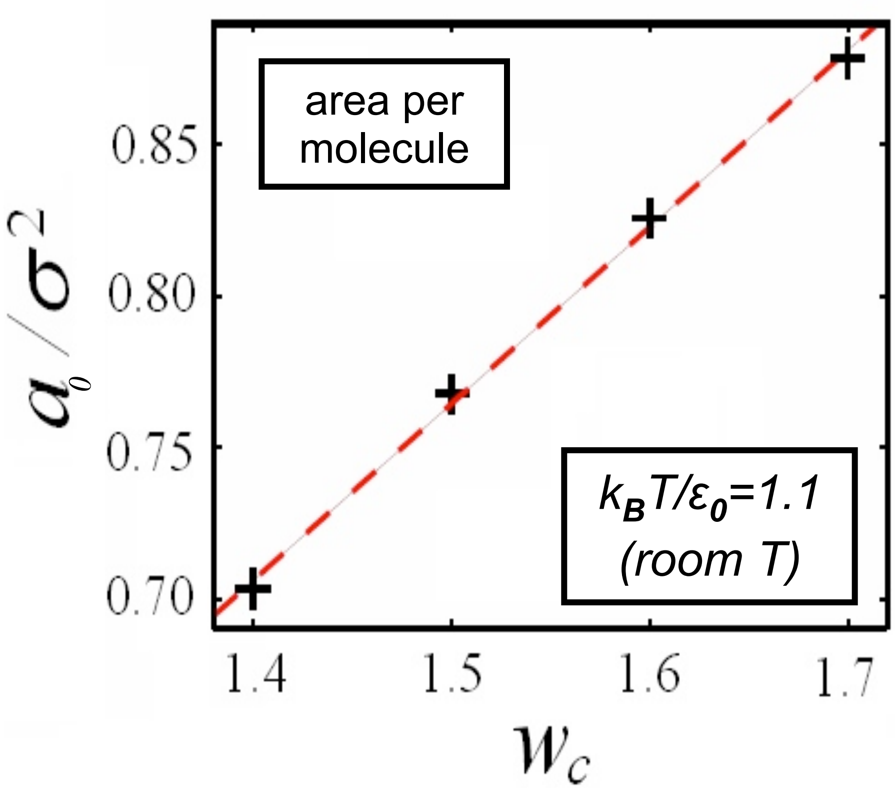
$\gamma = 0$

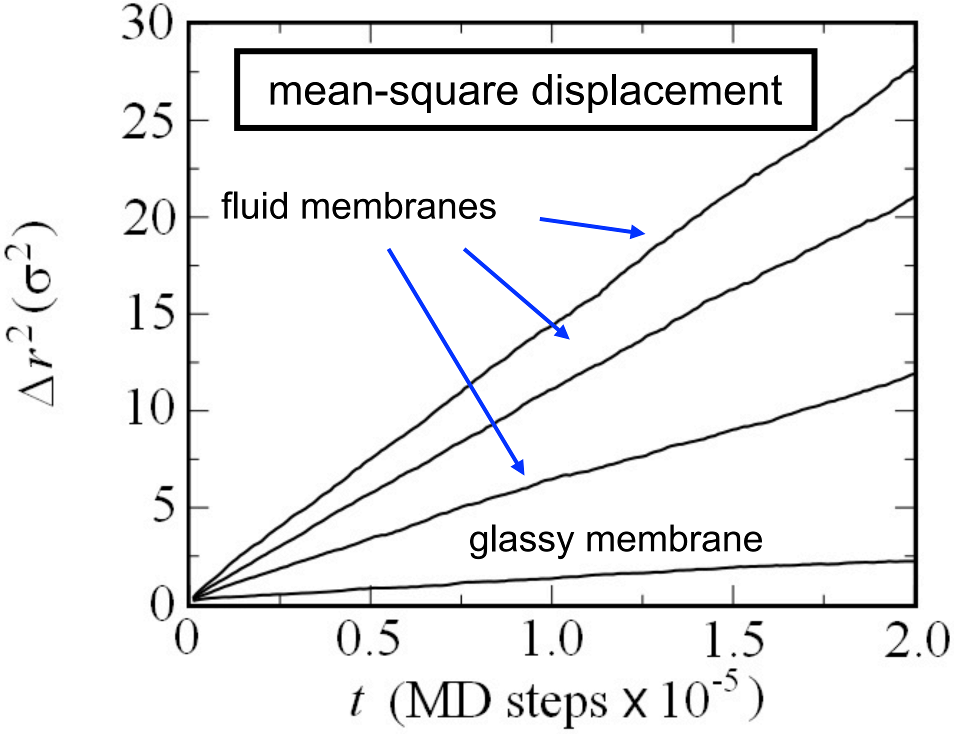

bending
rigidity
$\kappa\ /\ \varepsilon_0$
100
10
1
1.3
1.4
1.5
1.6
1.7
1.8
w_c

Properties of a physical membrane

- membrane thickness
- two-dimensional density
- molecular diffusion
- lateral compressibility

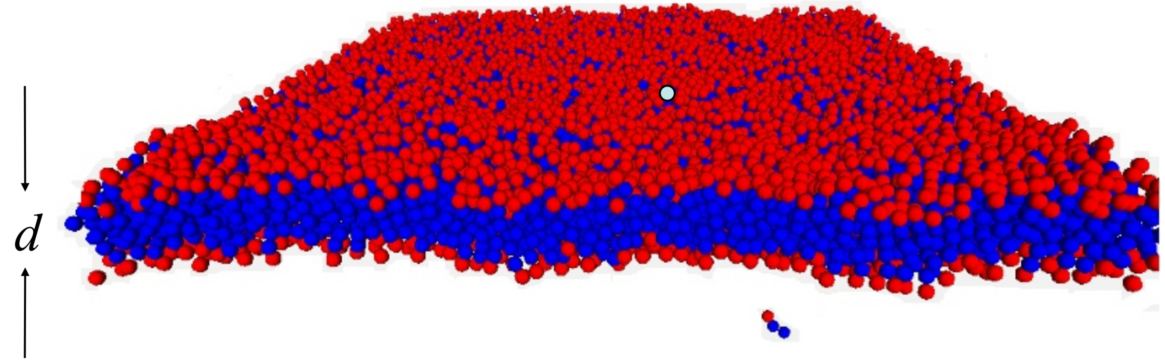


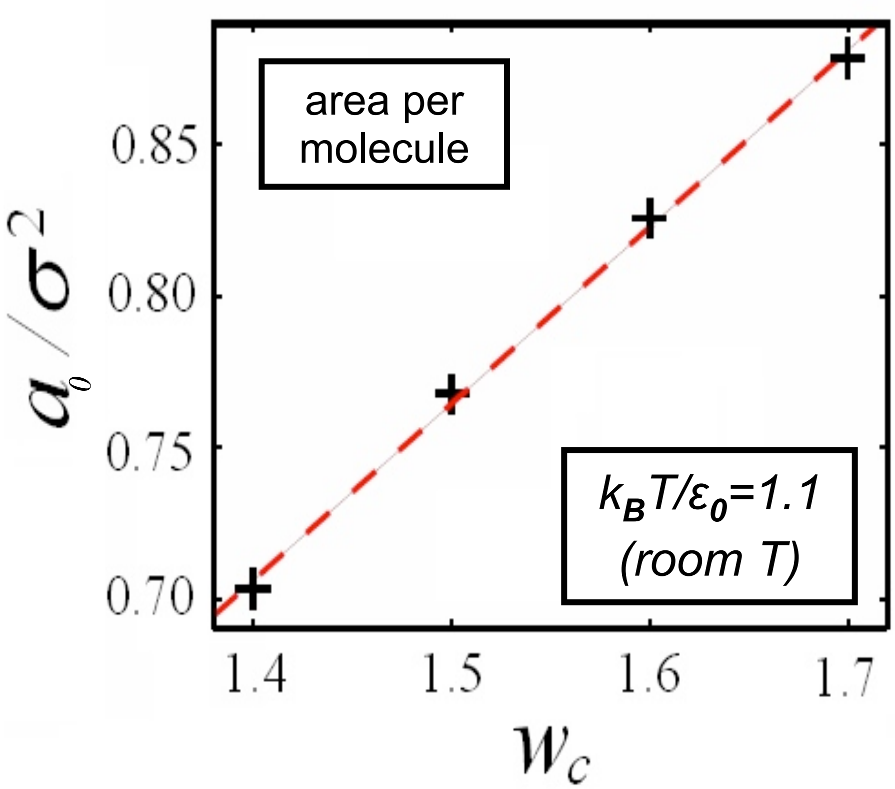


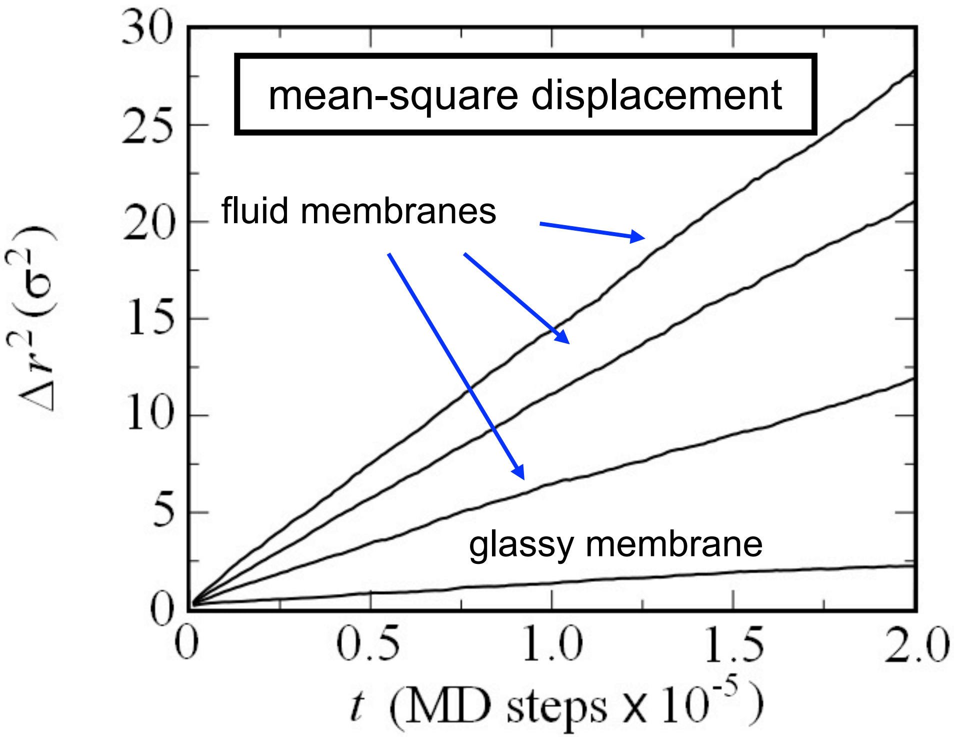


Properties of a physical membrane

- membrane thickness
- two-dimensional density
- molecular diffusion
- lateral compressibility

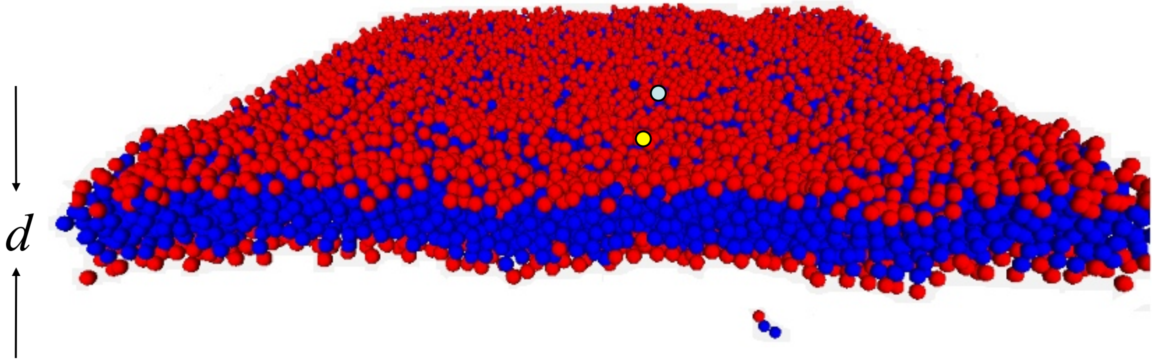


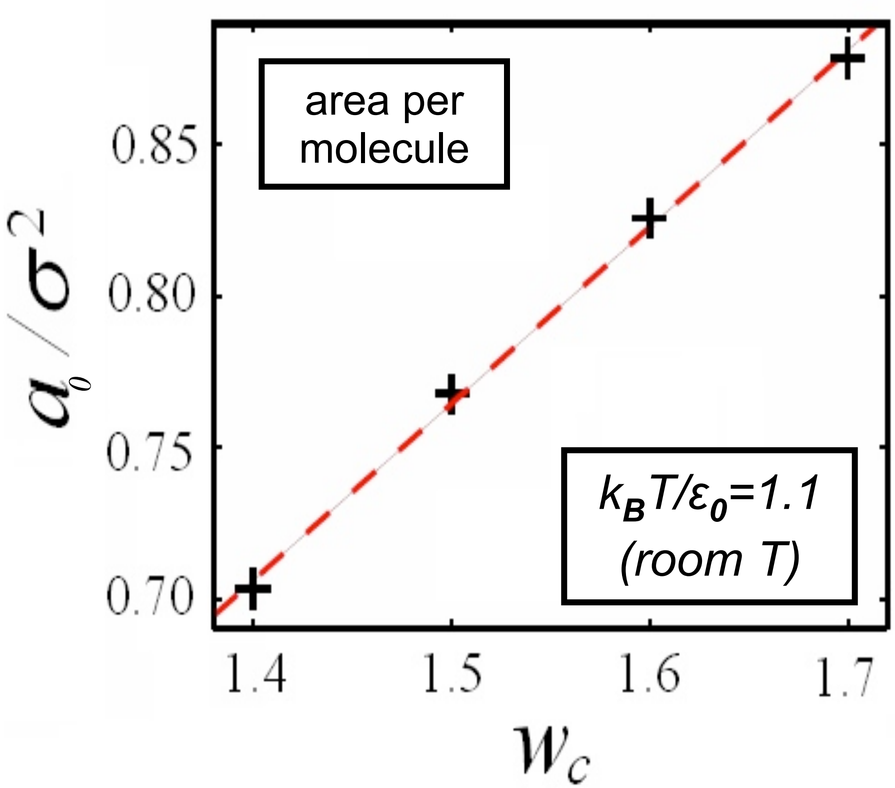


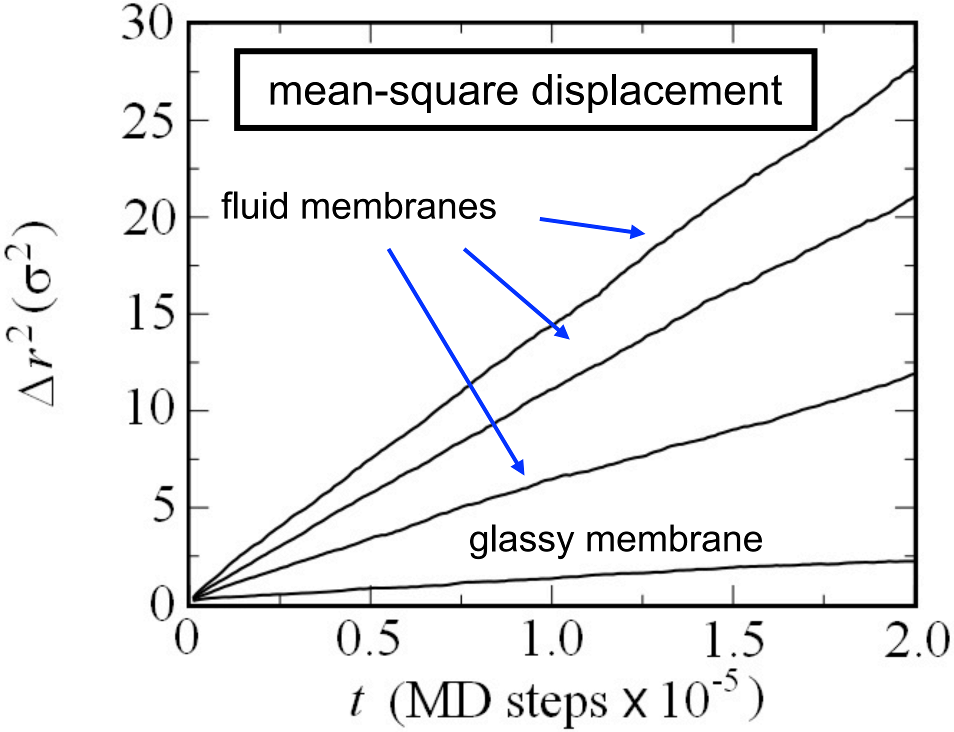


Properties of a physical membrane

- membrane thickness
- two-dimensional density
- molecular diffusion
- lateral compressibility

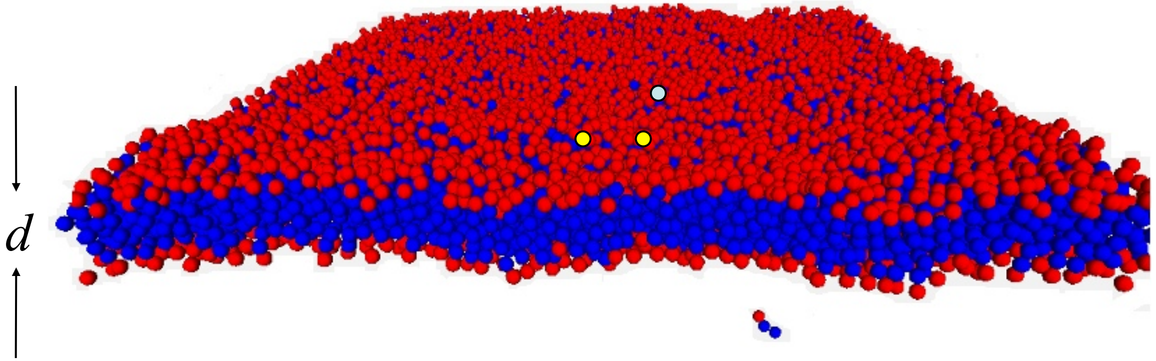


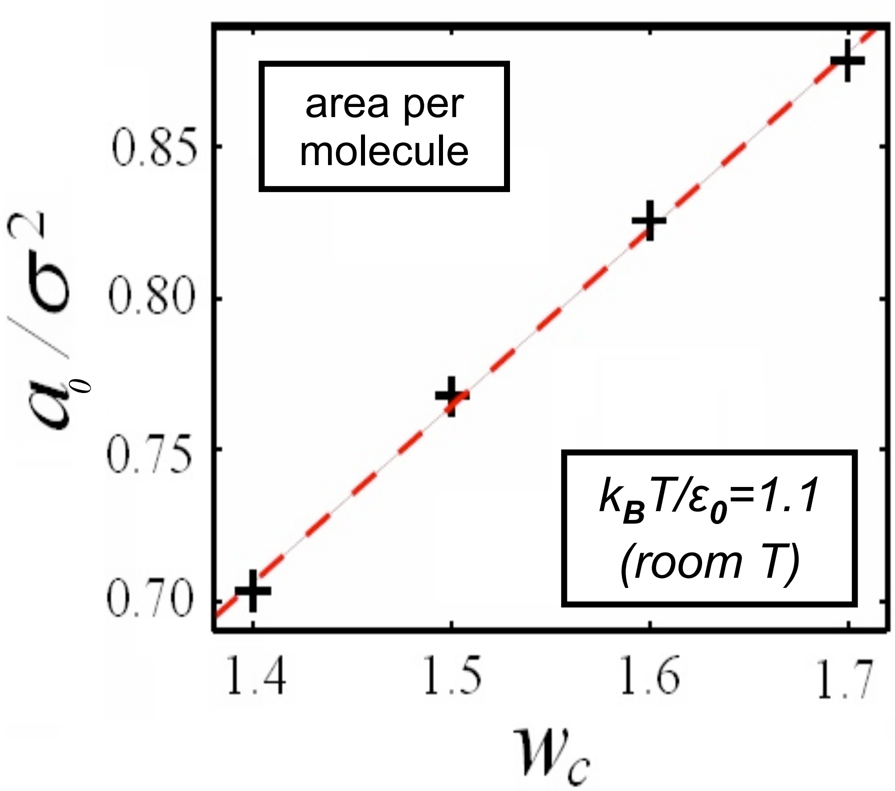




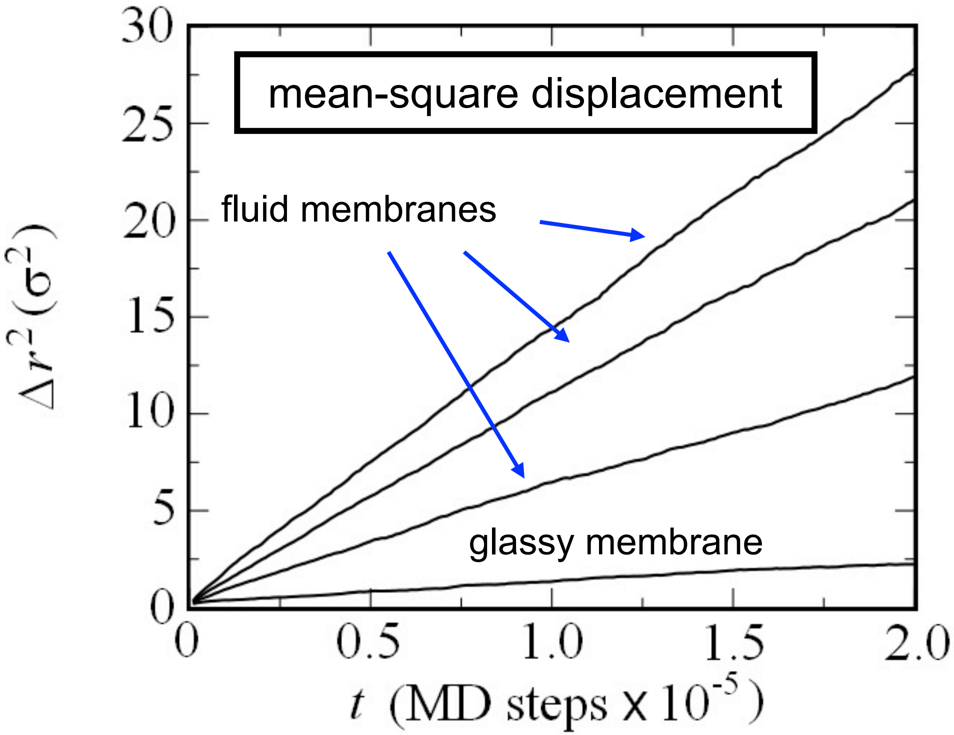
Properties of a physical membrane

- membrane thickness
- two-dimensional density
- molecular diffusion
- lateral compressibility



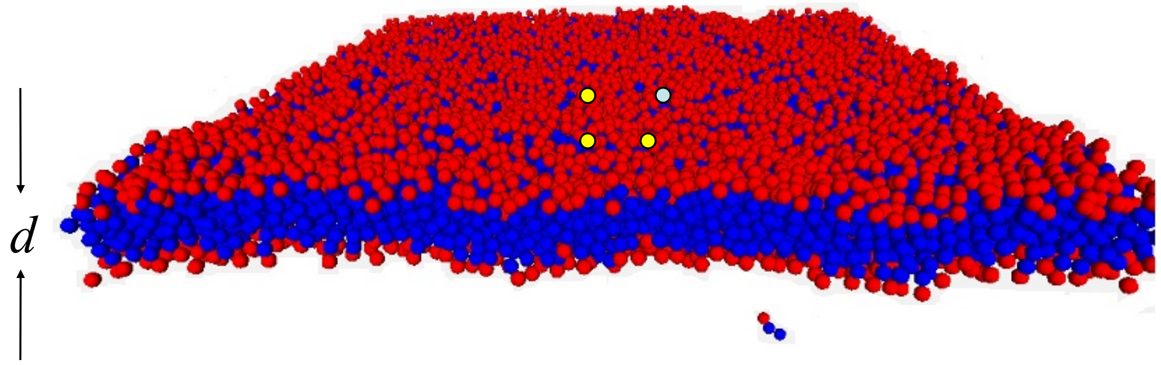

area per molecule

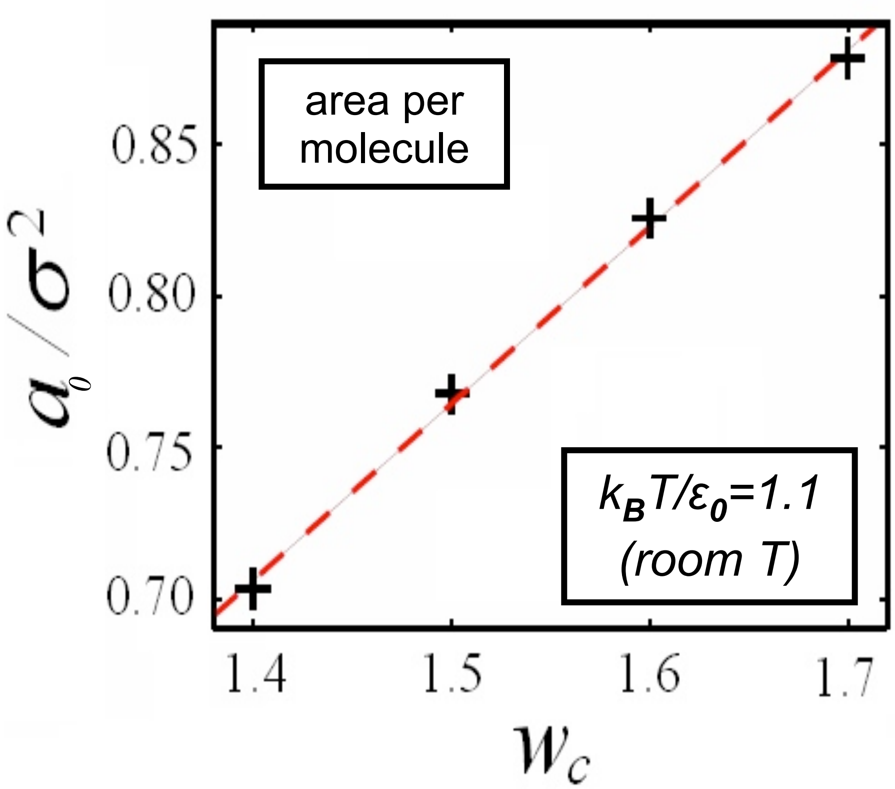
k_BT/ε_0=1.1
(room T)

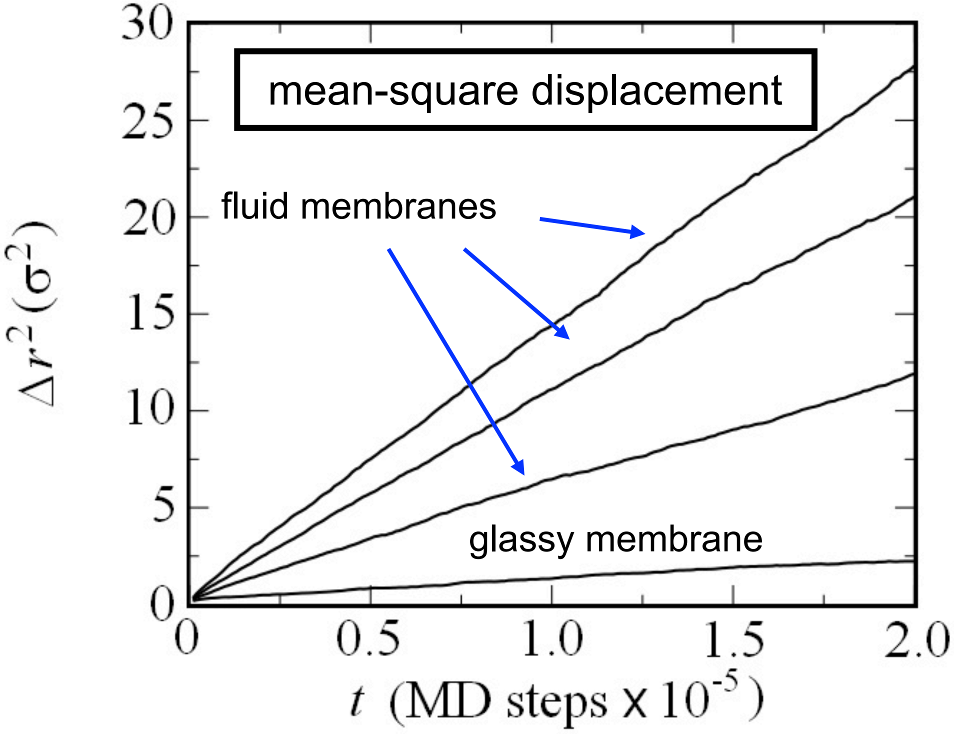

mean-square displacement

Properties of a physical membrane

- membrane thickness
- two-dimensional density
- molecular diffusion
- lateral compressibility

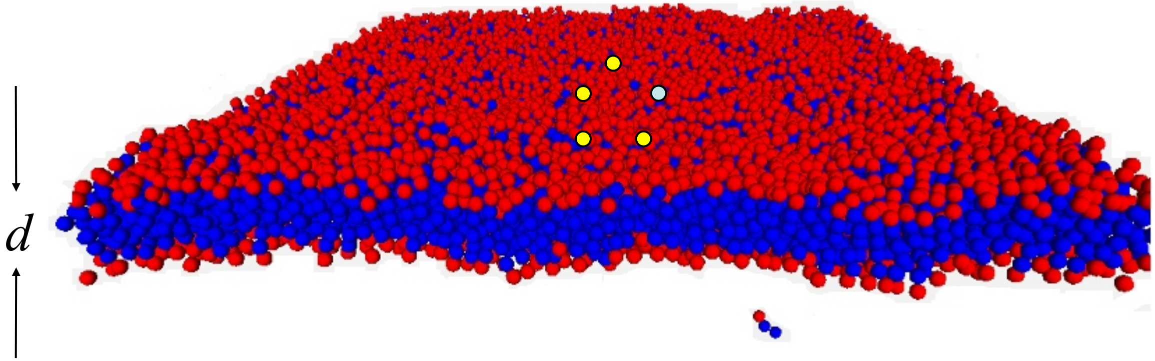


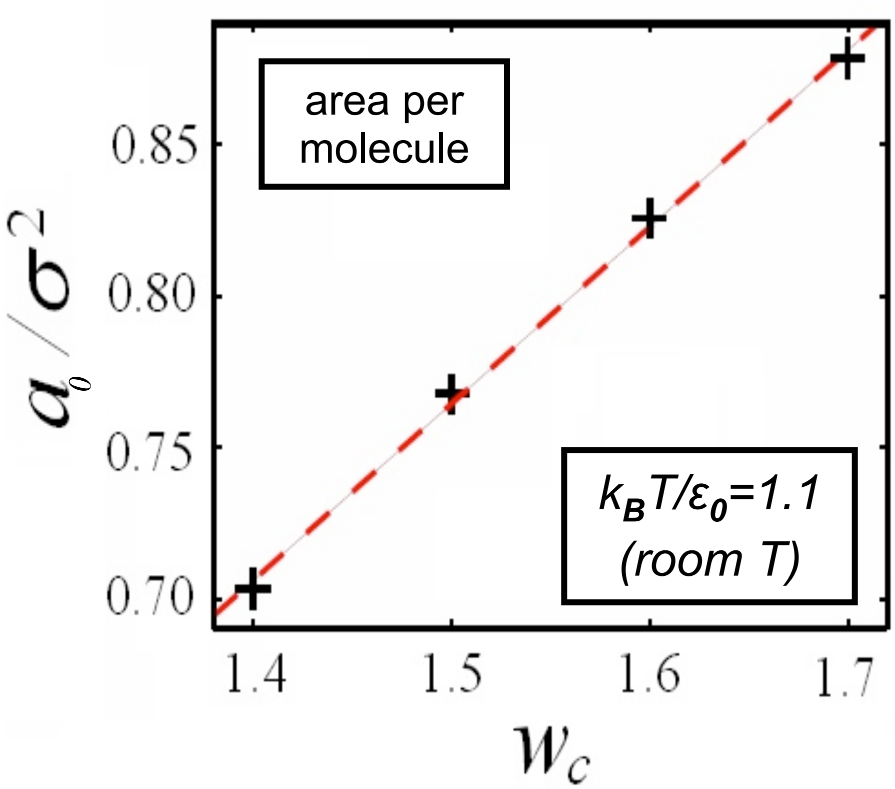


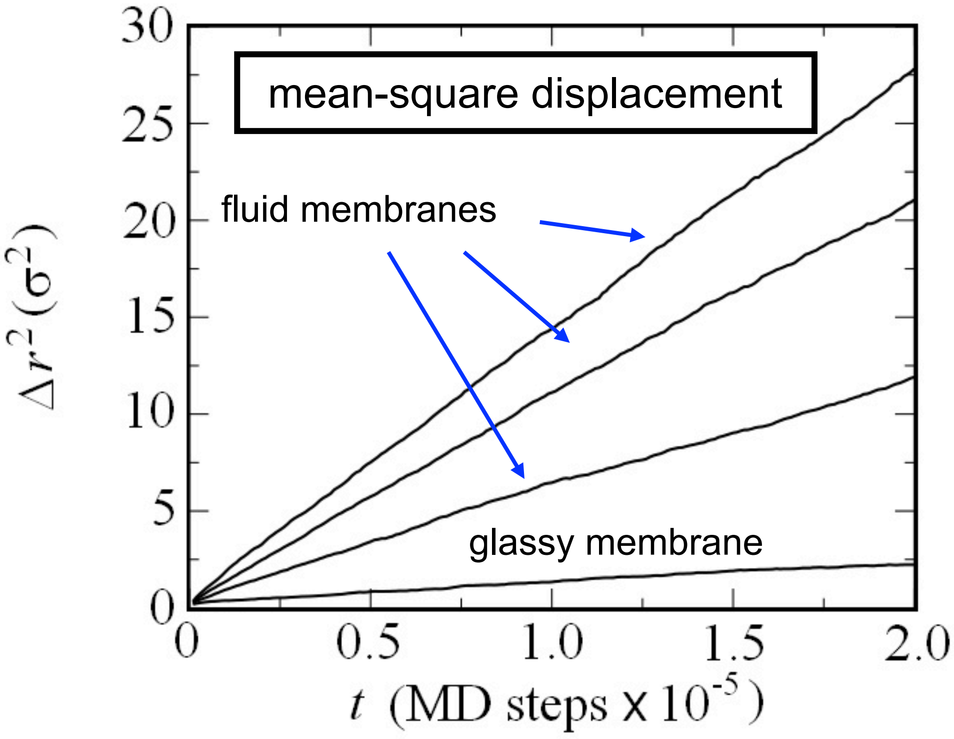


Properties of a physical membrane

- membrane thickness
- two-dimensional density
- molecular diffusion
- lateral compressibility

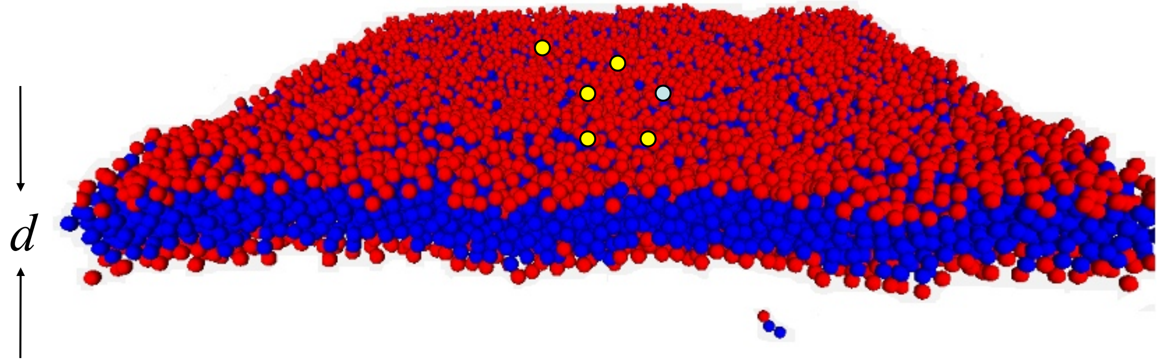


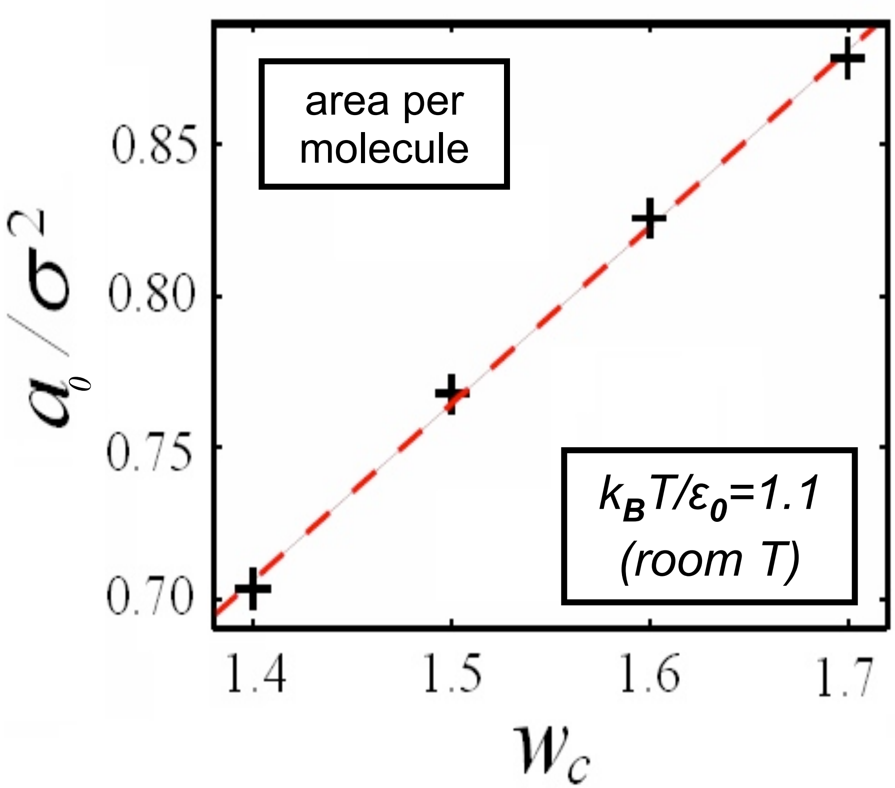


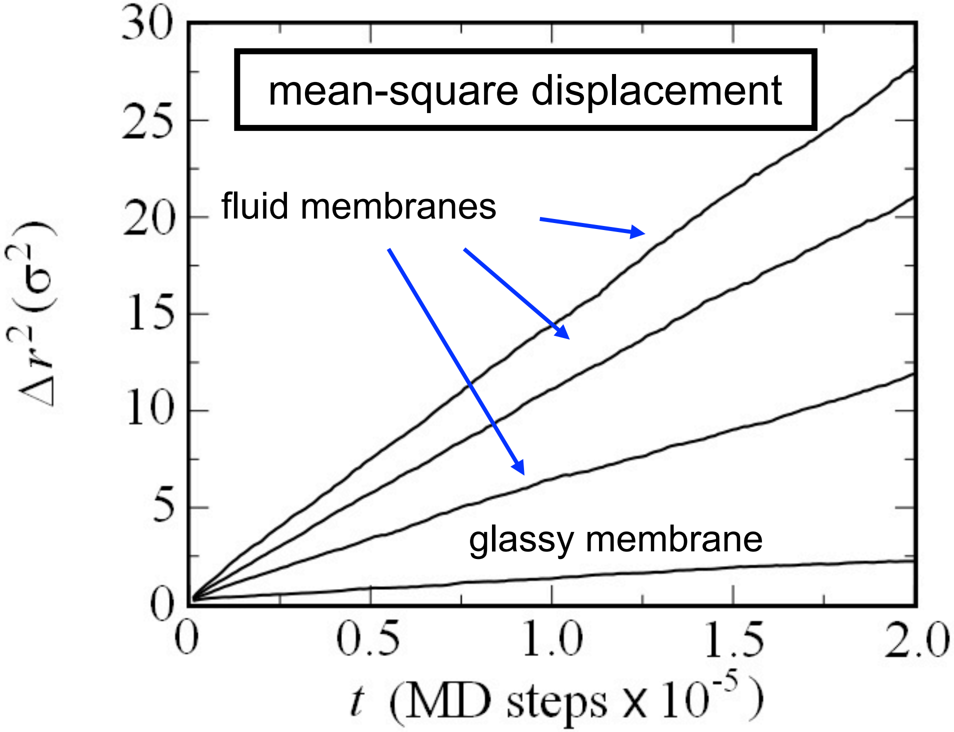


Properties of a physical membrane

- membrane thickness
- two-dimensional density
- molecular diffusion
- lateral compressibility

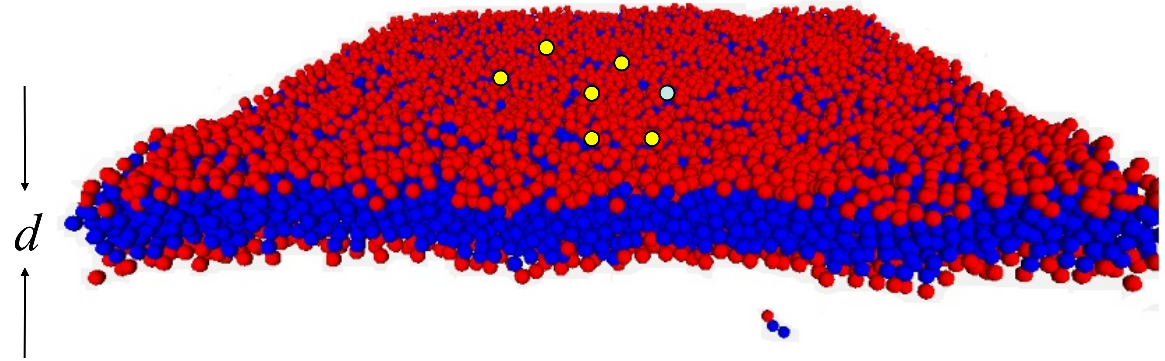


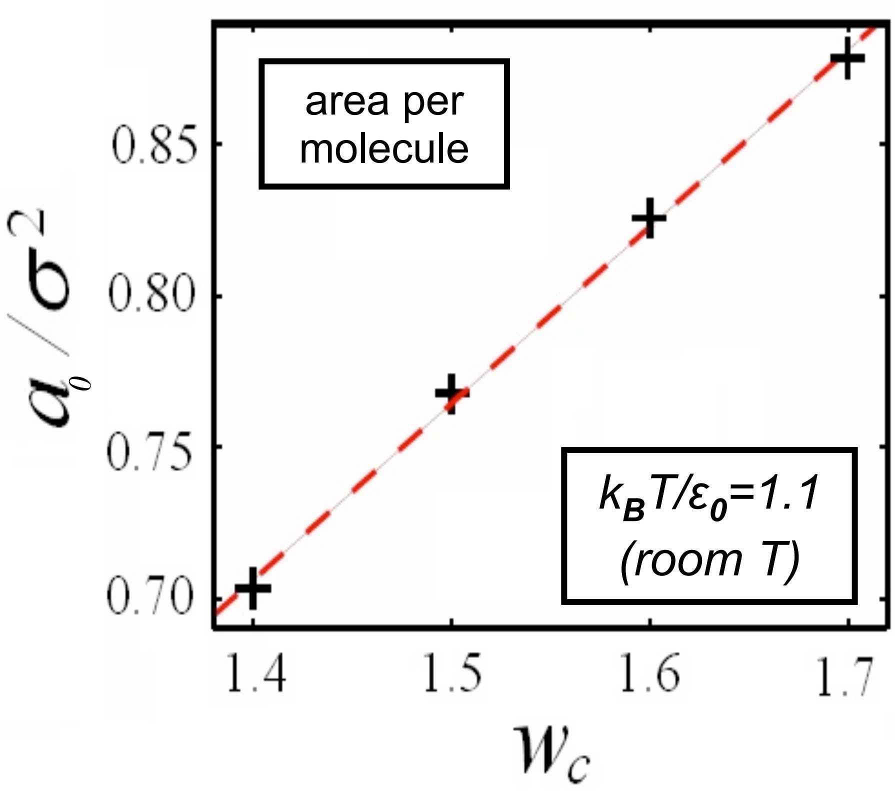


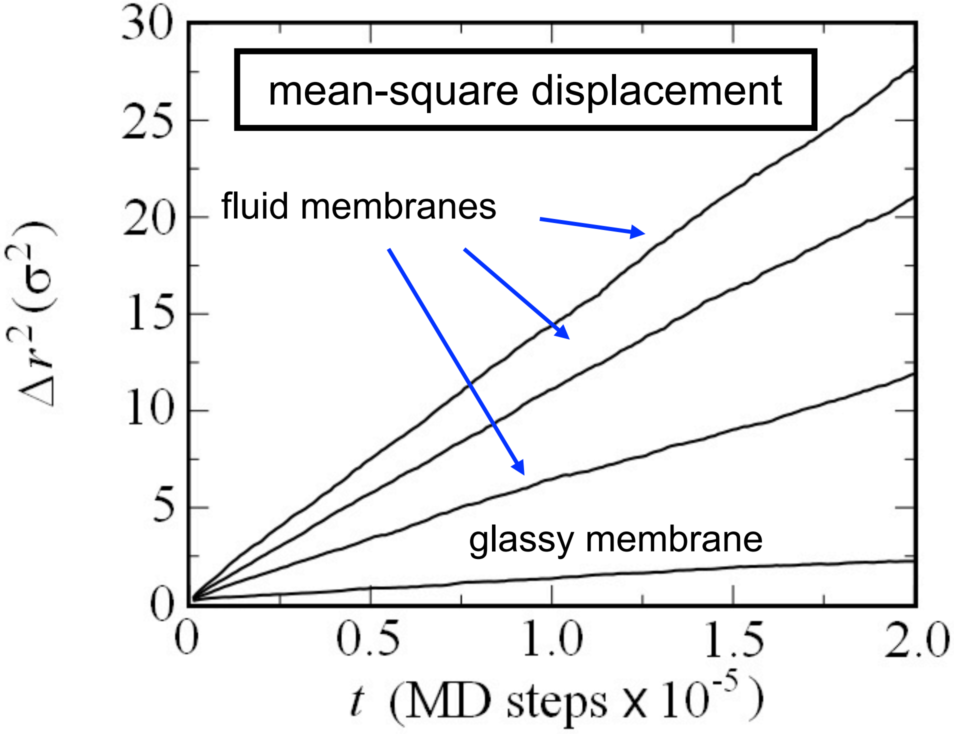


Properties of a physical membrane

- membrane thickness
- two-dimensional density
- molecular diffusion
- lateral compressibility

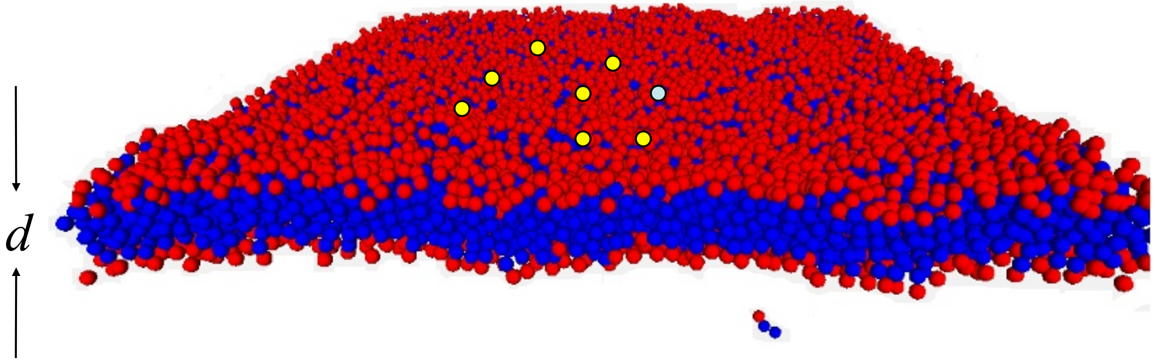


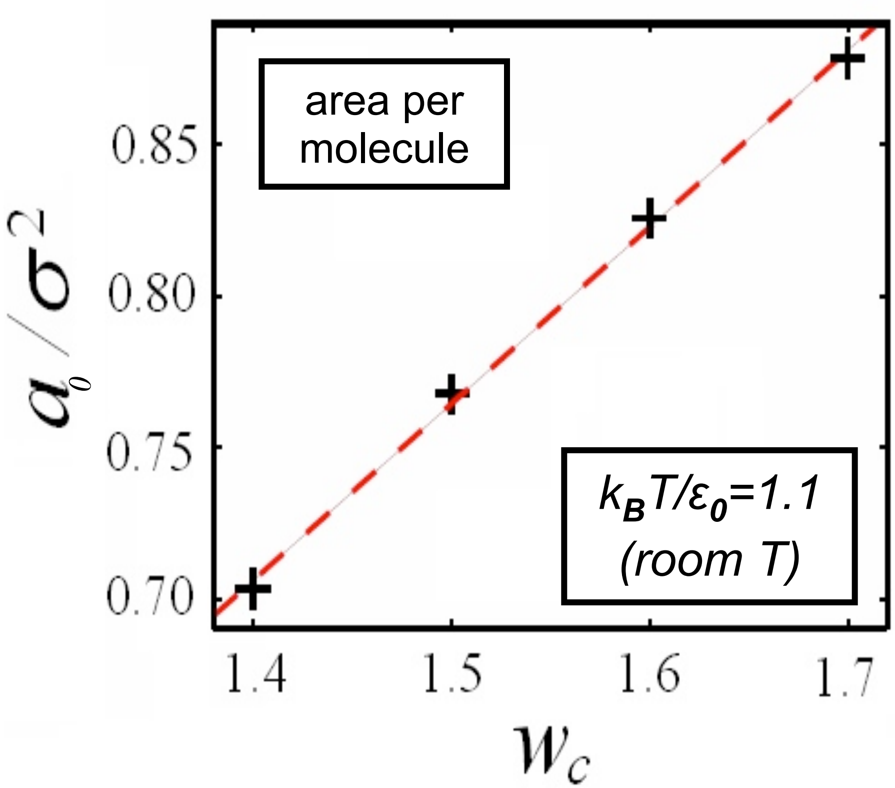


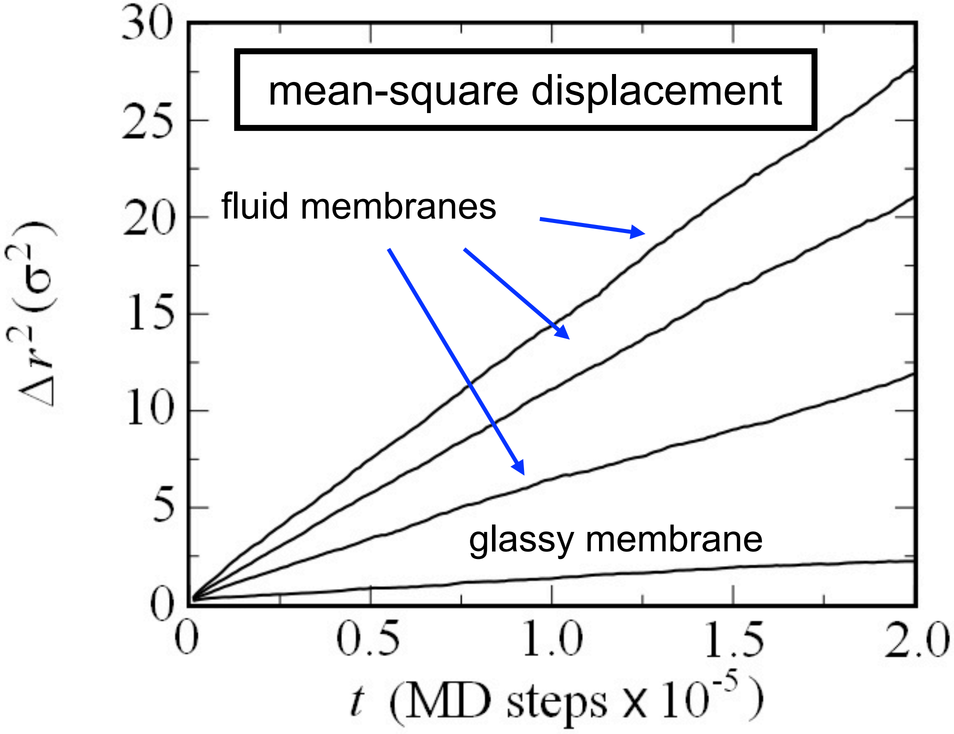


Properties of a physical membrane

- membrane thickness
- two-dimensional density
- molecular diffusion
- lateral compressibility

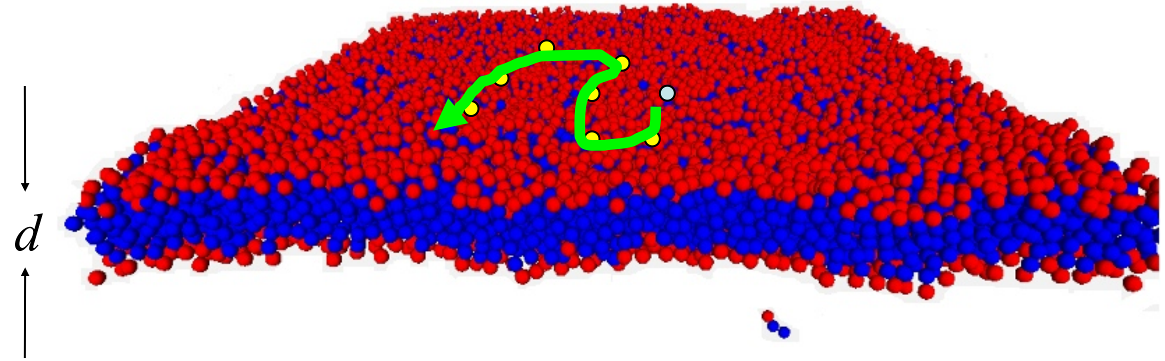


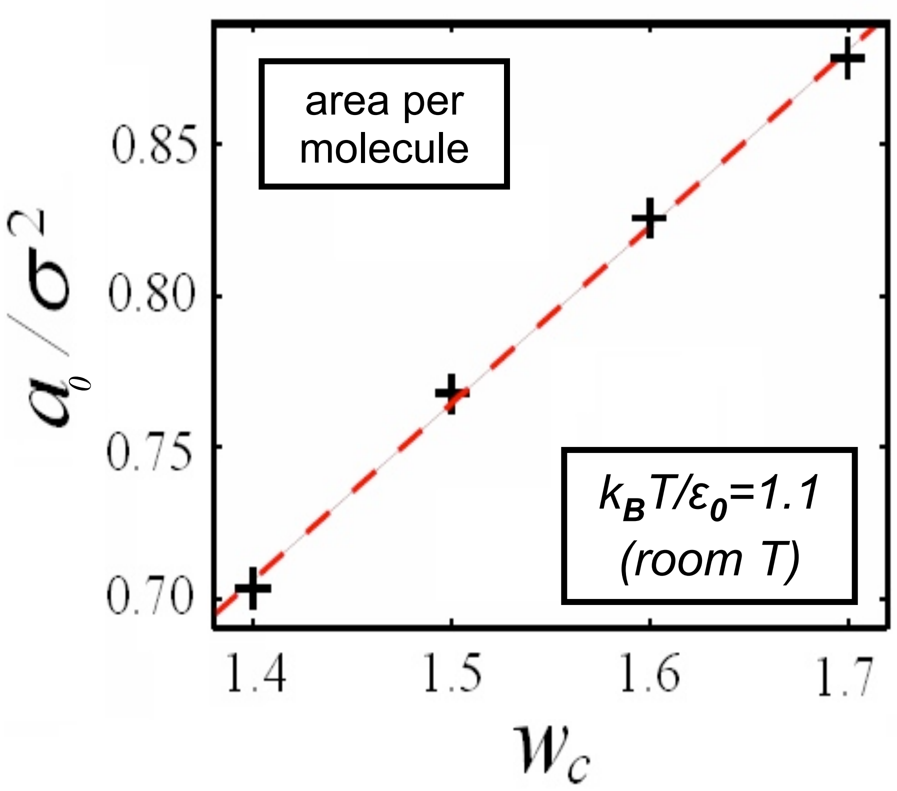


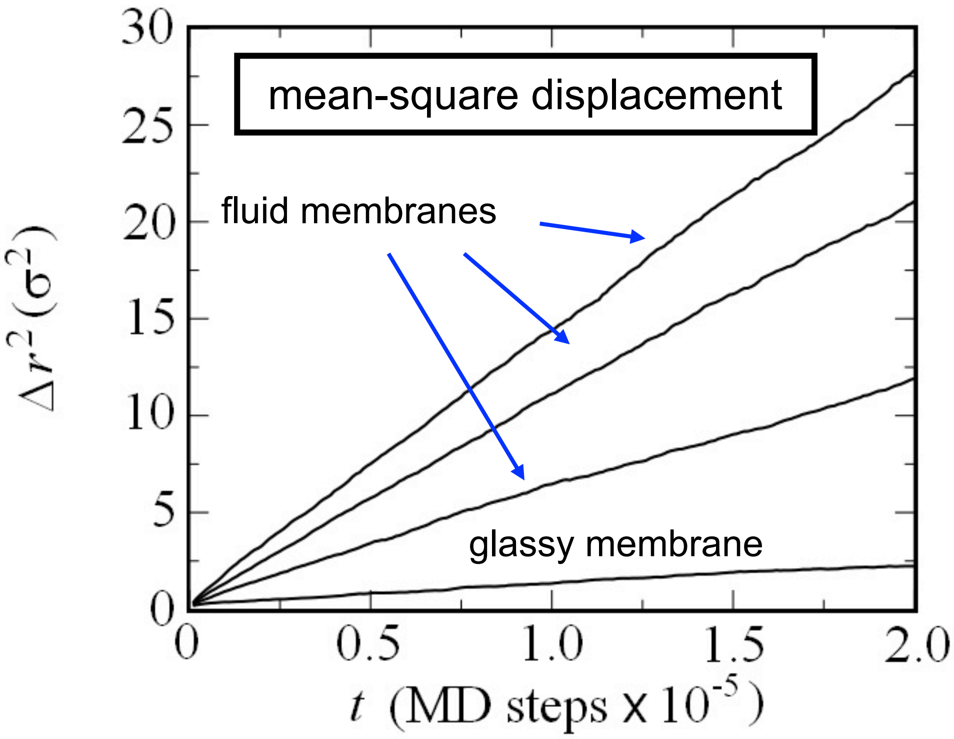


Properties of a physical membrane

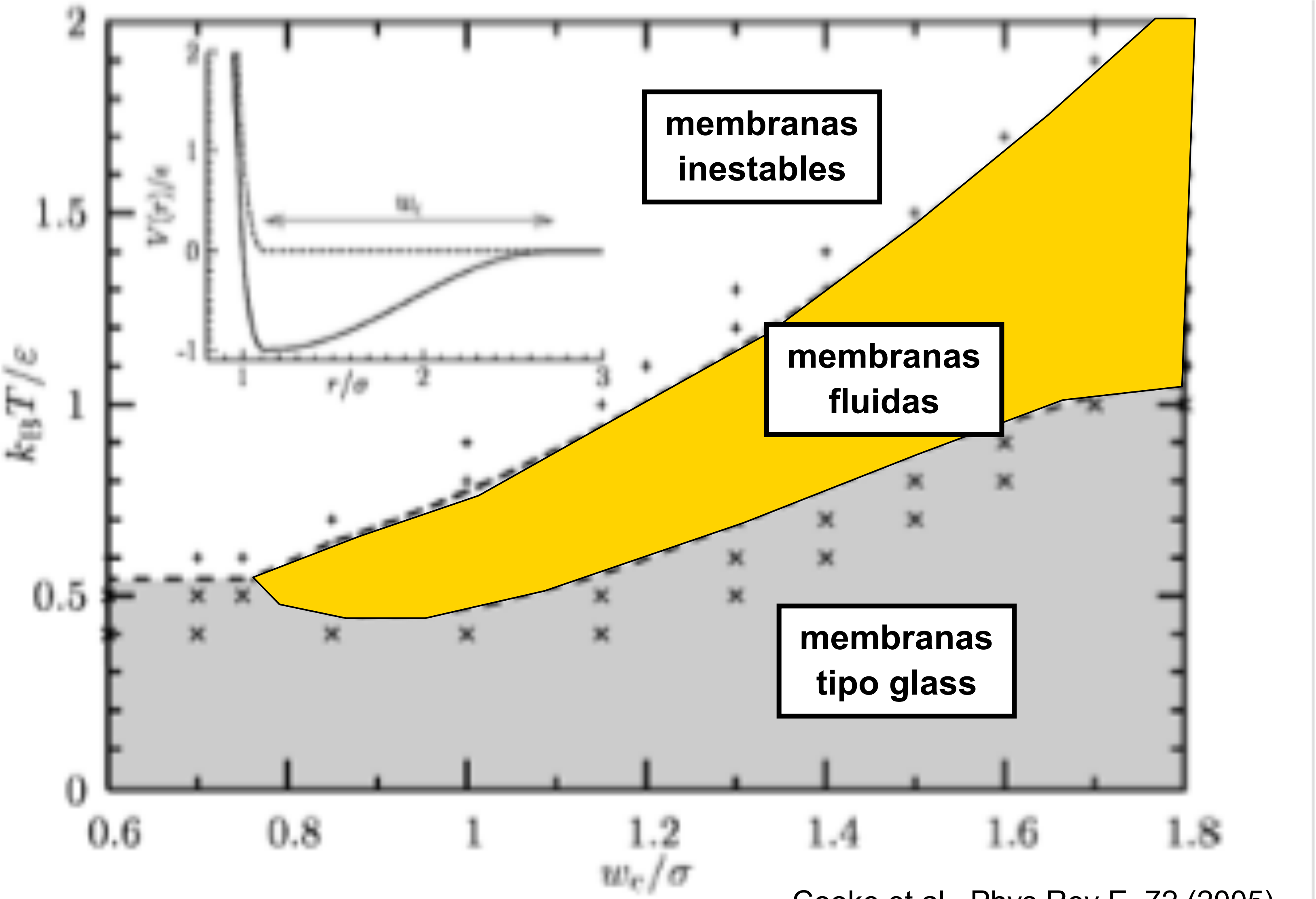
- membrane thickness
- two-dimensional density
- molecular diffusion
- lateral compressibility





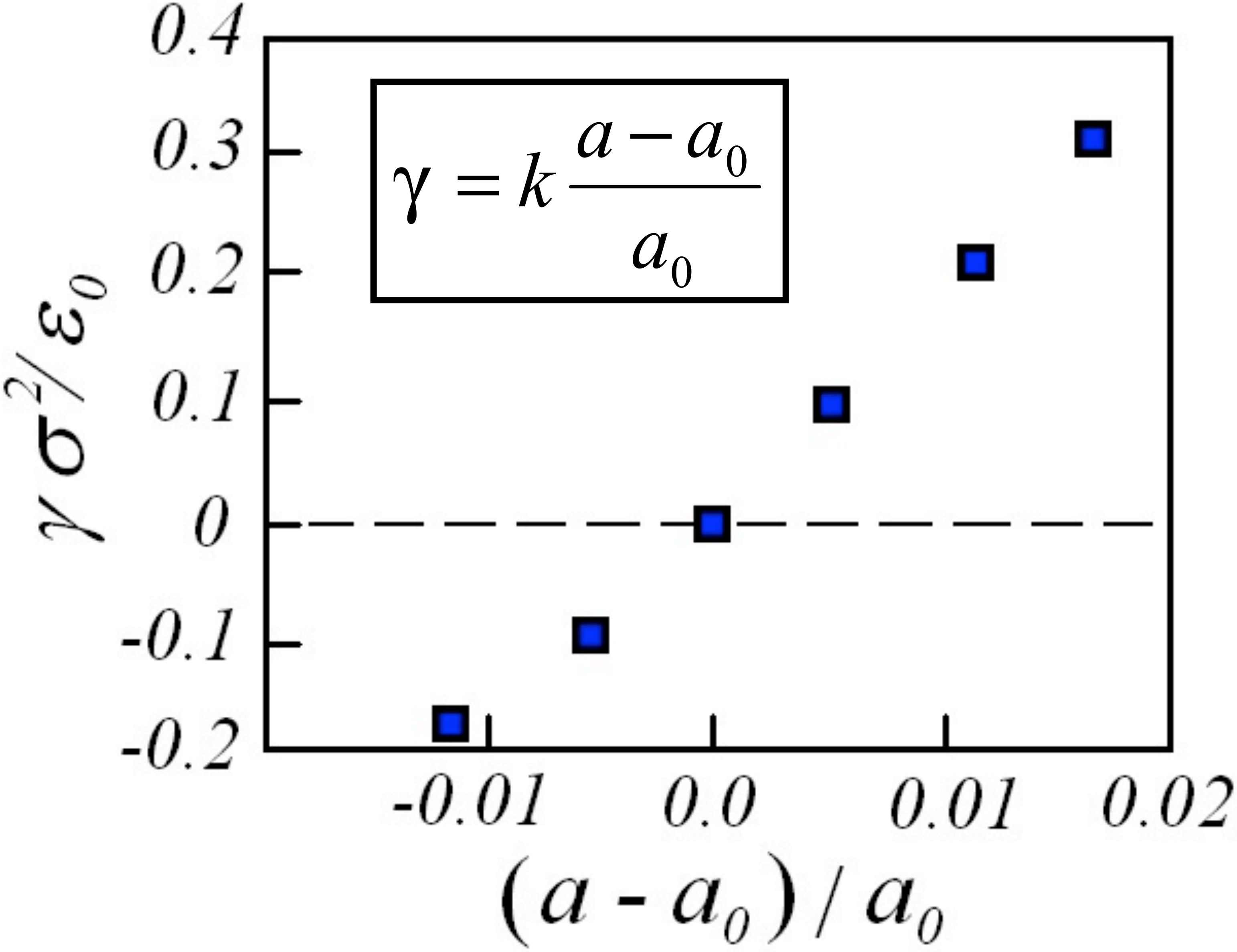


Influencia del parámetro de alcance w_c



Cooke et al., Phys Rev E, 72 (2005)

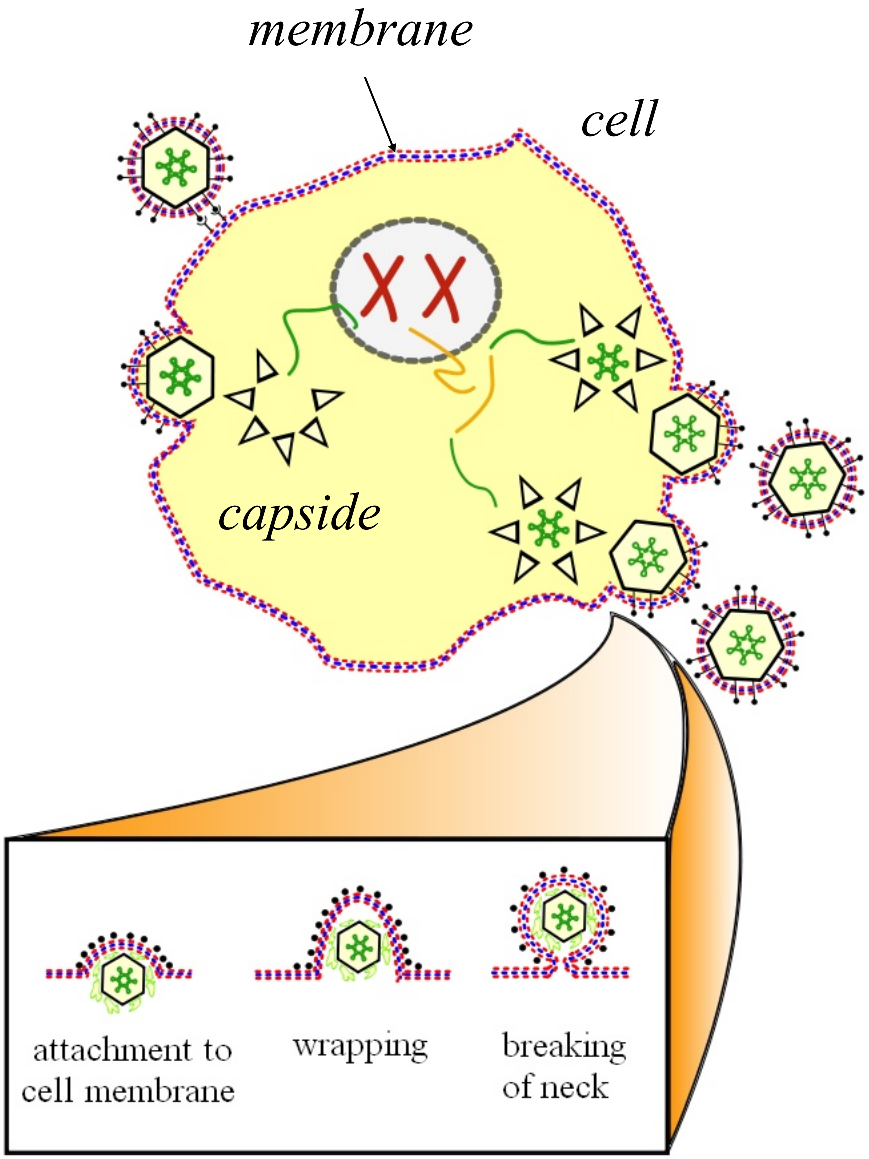
lateral compressibility, *k*


$$\gamma = k\,\frac{a - a_0}{a_0}$$

Application: wrapping and budding of viruses

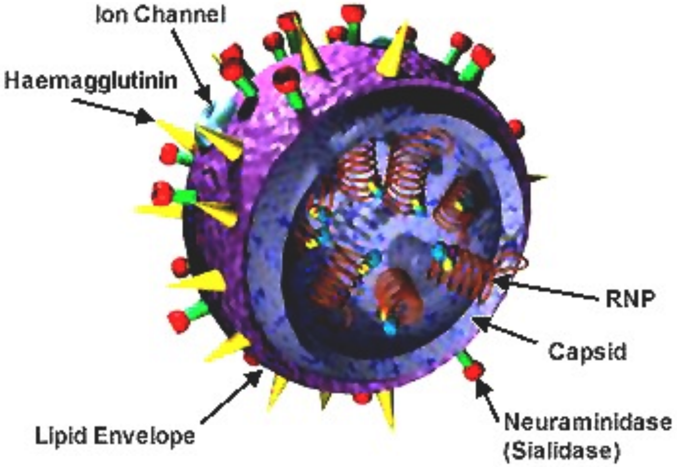
After using the cell machinary to replicate, some viruses leave the cell by acquiring a membrane coating

We aim at understanding the physics of the problem using a very simplified model



enveloped
virus

d= 10 nm



$2R$

40 – 600 nm

$$\frac{2R}{d} = 4 - 60.$$

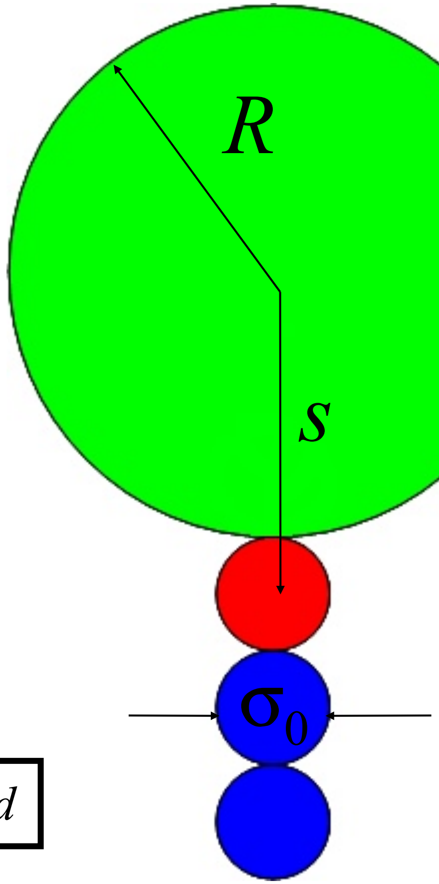
Since $d = 6\sigma_0$, we have:

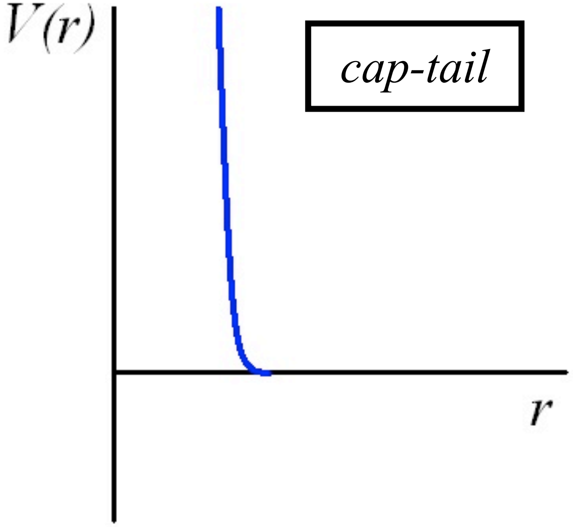
$$R = 12 - 180\sigma_0$$

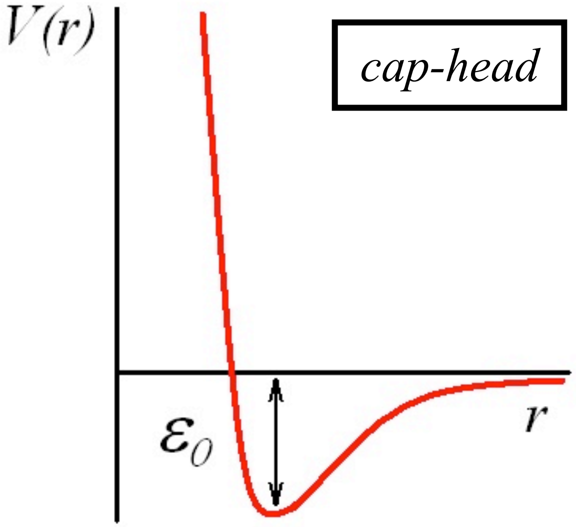
Membrane-particle interaction:

$$V_{cap-tail}(r) = 4\varepsilon_0 \begin{cases} \left[\left(\dfrac{\sigma_0}{r-s}\right)^{12} - \left(\dfrac{\sigma_0}{r-s}\right)^{6} + \dfrac{1}{4}\right], & r < r_c, \\ 0, & r > r_c. \end{cases}$$

$$V_{cap-head}(r) = 4\varepsilon_s \left[\left(\dfrac{\sigma_0}{r-s}\right)^{12} - \left(\dfrac{\sigma_0}{r-s}\right)^{6}\right]$$

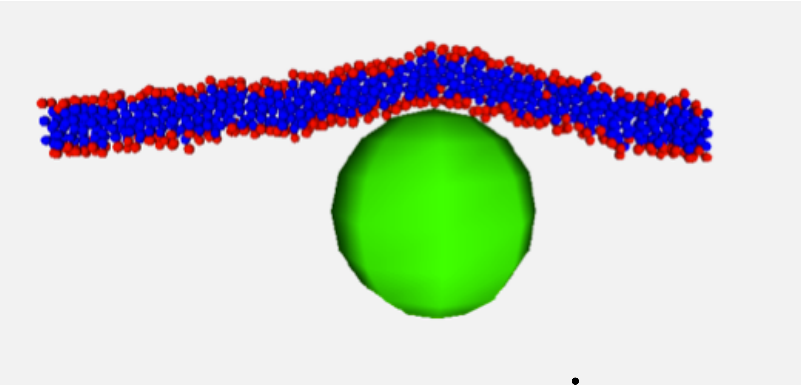




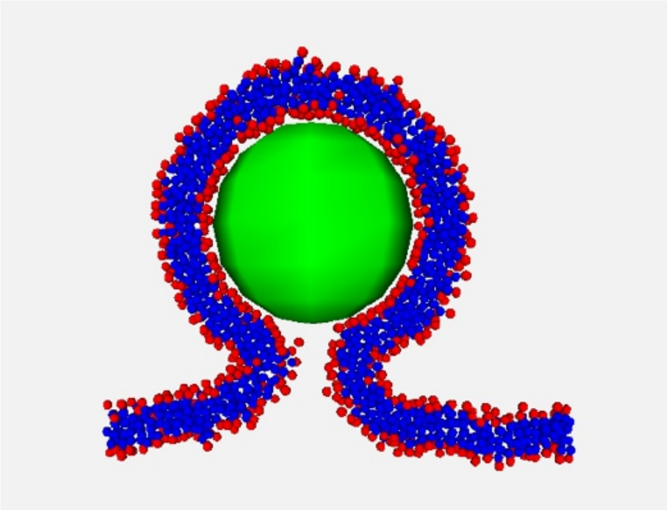


$$s = R + \frac{\sigma_0}{2}$$

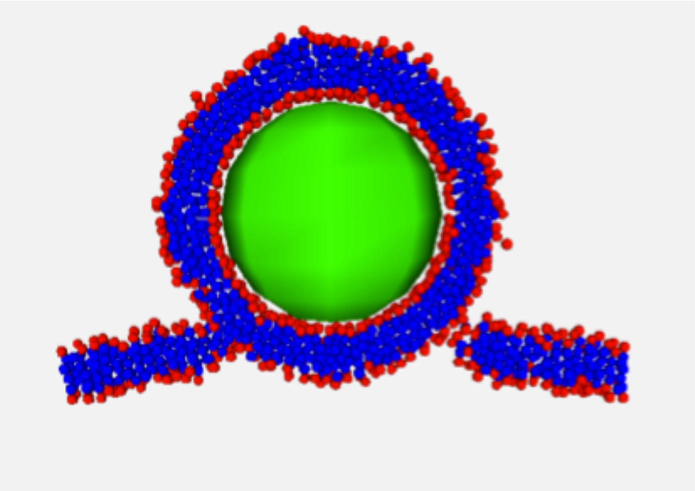
Typical behaviours



non-wrapping



wrapping &
budding

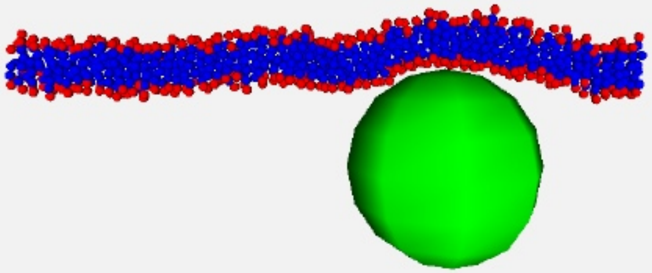


membrane
breaking

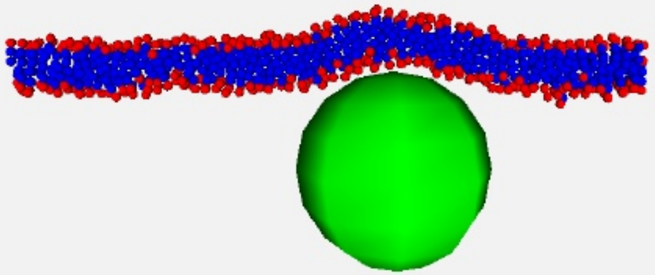
NON - WRAPPING

$\tau_0 = 9.85$ ps

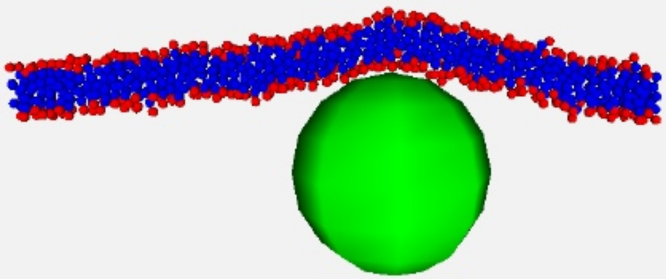
$R = 10\,\sigma_0$



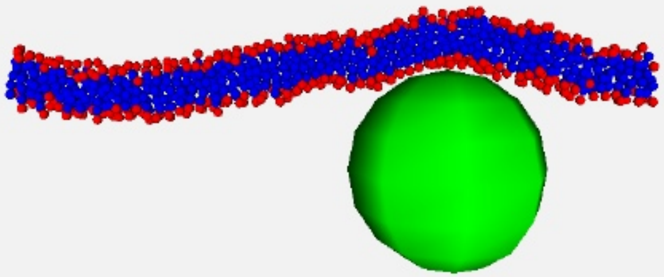
$\tau/\tau_0 = 10^3$



$\tau/\tau_0 = 5\text{x}10^3$

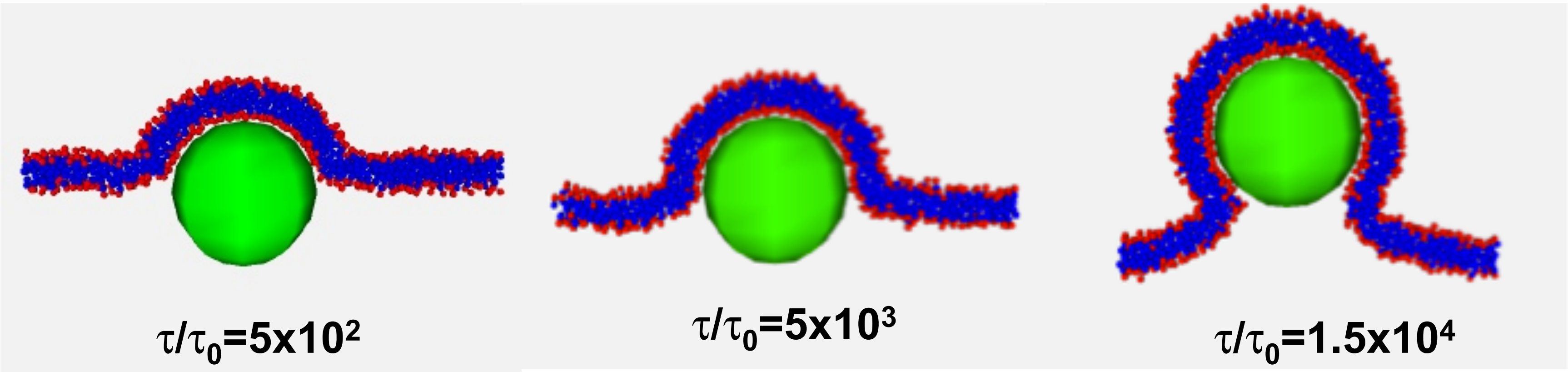


$\tau/\tau_0 = 10^4$



$\tau/\tau_0 = 3\text{x}10^4$

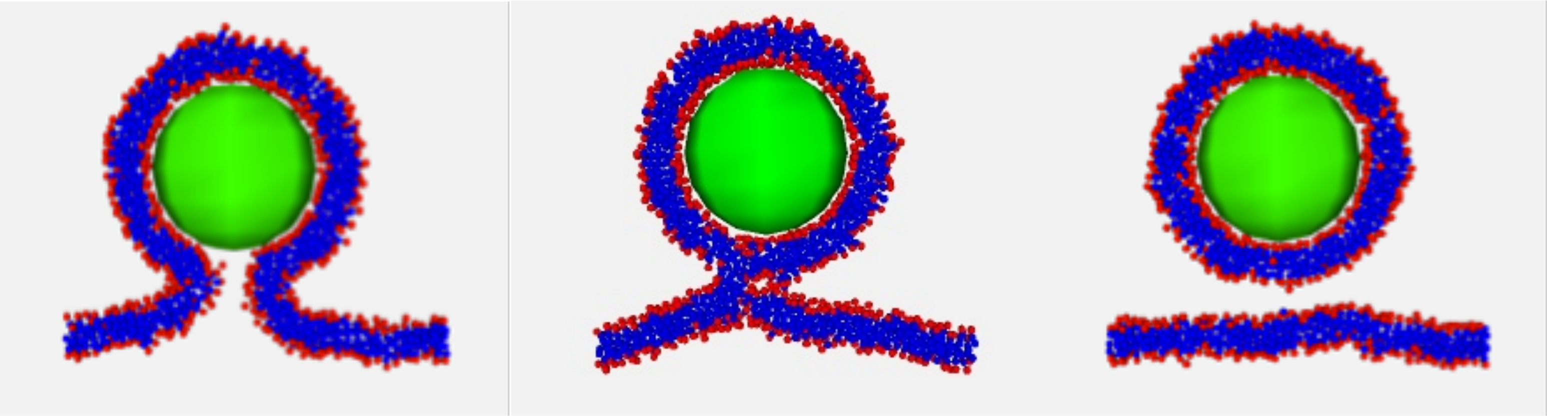
WRAPPING



$\tau/\tau_0=5x10^2$

$\tau/\tau_0=5x10^3$

$\tau/\tau_0=1.5x10^4$

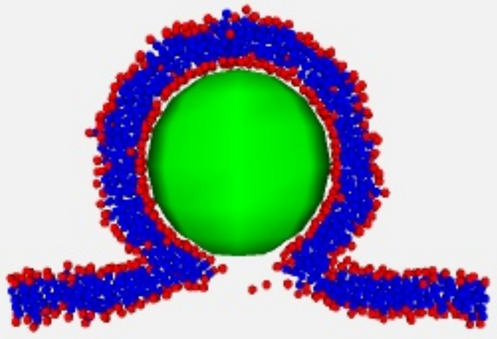


$\tau/\tau_0=1.55x10^4$

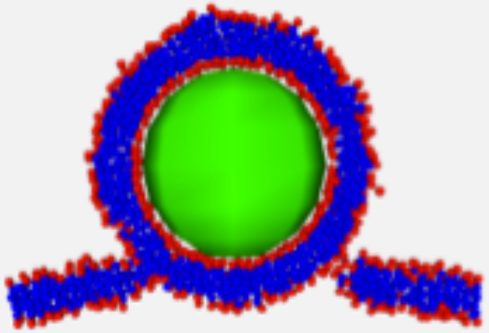
$\tau/\tau_0=1.6x10^4$

$\tau/\tau_0=1.65x10^4$

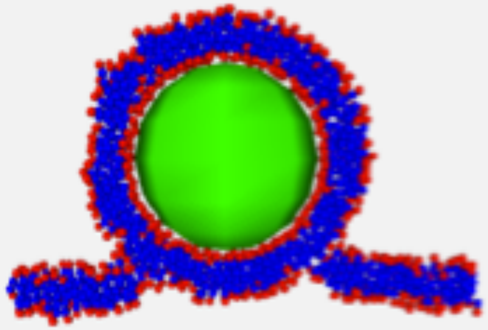
MEMBRANE BREAKING



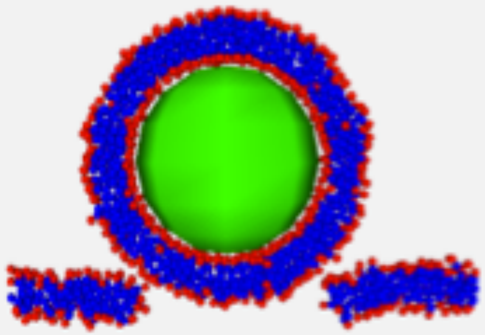
$\tau/\tau_0=5.5x10^3$



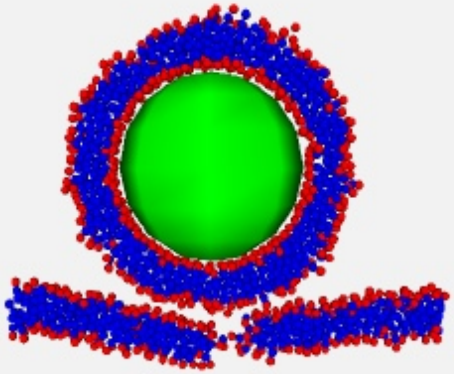
$\tau/\tau_0=6x10^3$



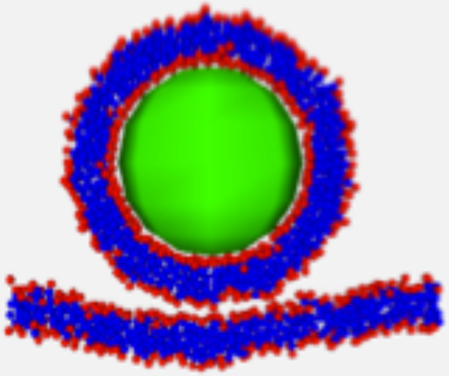
$\tau/\tau_0=7x10^3$



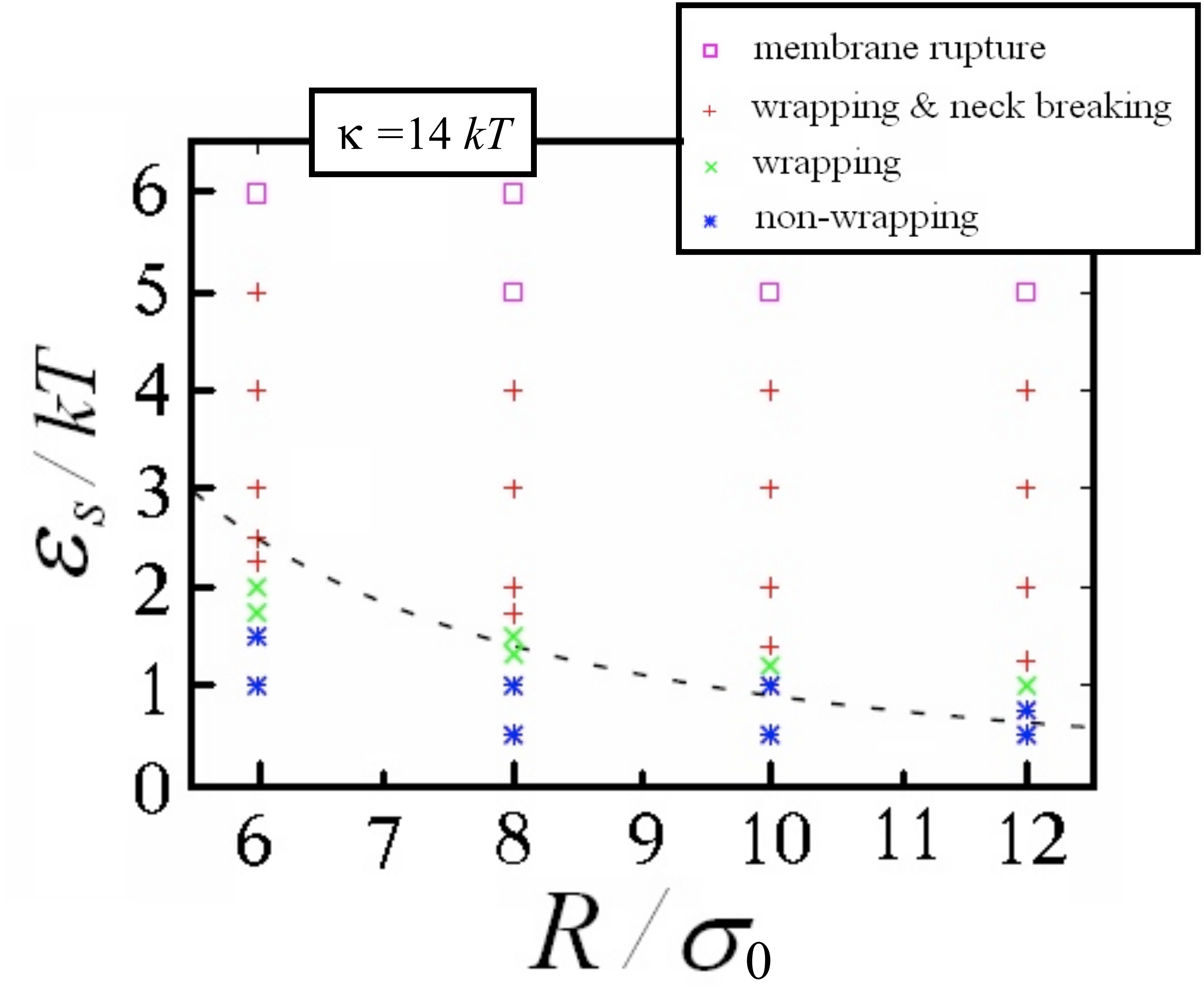
$\tau/\tau_0=7.5x10^3$

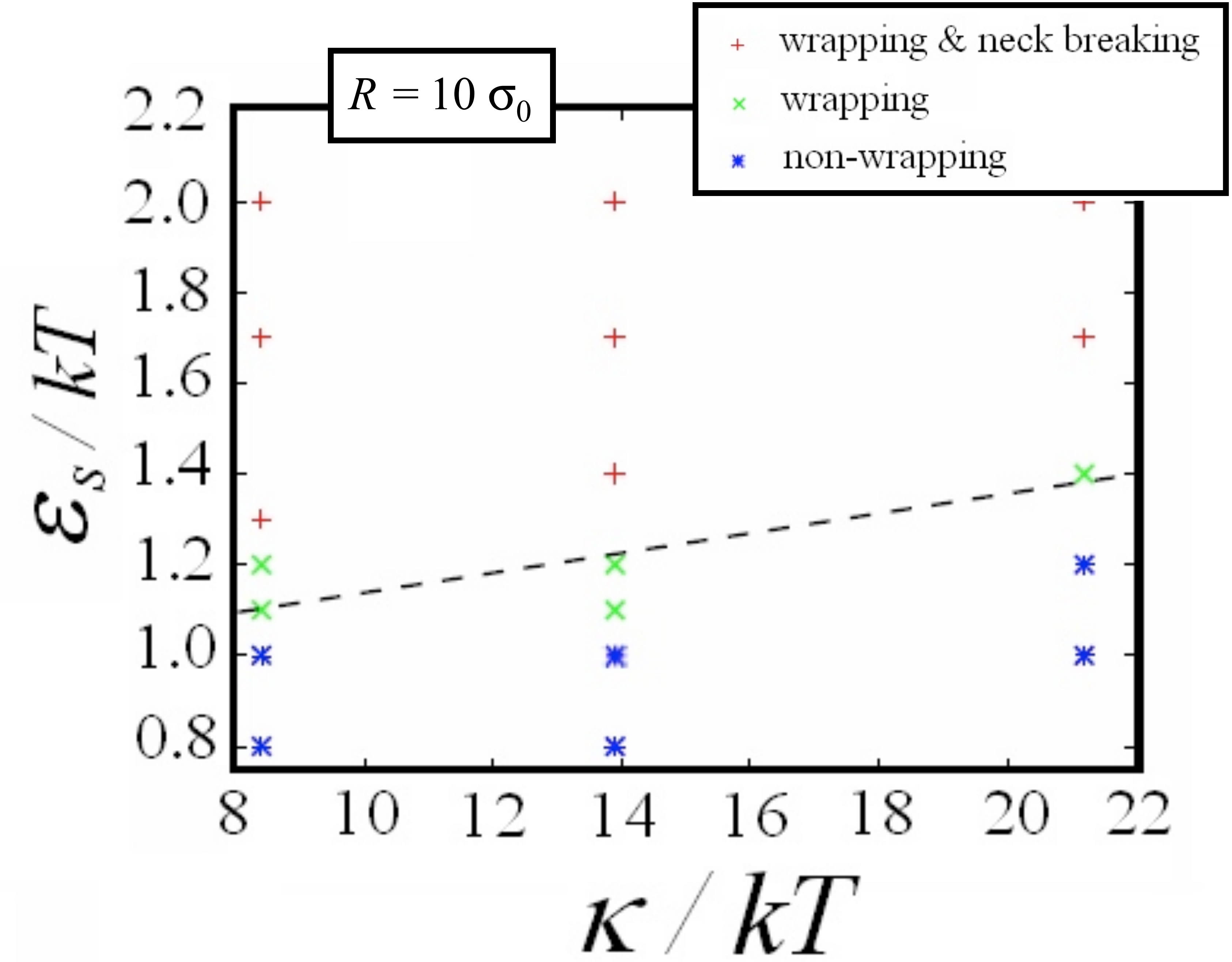


$\tau/\tau_0=9.5x10^3$



$\tau/\tau_0=10^4$


κ =14 kT
□ membrane rupture
+ wrapping & neck breaking
× wrapping
✱ non-wrapping
ε_S / kT
0
1
2
3
4
5
6
R / σ_0
6
7
8
9
10
11
12


$R = 10\ \sigma_0$
+ wrapping & neck breaking
× wrapping
* non-wrapping
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.8
ε_s / kT
8
10
12
14
16
18
20
22
κ / kT