# MOLECULAR DYNAMICS OF PROTEINS I

SAPIENZA
UNIVERSITÀ DI ROMA

CB_23_24 LECTURE N. 13 Rome, tue oct 24 2023.
Partly based on materials from the lab of jaanhan Chen (https://people.chem.umass.edu/jchenlab/ ) and Berend Smit (http://www.cchem.berkeley.edu/molsim/personal_pages/berend/index.html )

# Molecular Dynamics

- Objective: $\{r_1(t), \ldots, r_N(t)\} \rightarrow \{r_1(t+\Delta t), \ldots, r_N(t+\Delta t)\}$ $f = ma$
- Basic idea: solve Newton's equation of motion numerically
  - Given current coordinates (x), velocities (v)
    - Forces can be calculated based on coordinates (from $f = -\partial V/\partial x$)
    - $x(t+\Delta t) = x(t) + v(t)\,\Delta t$
    - $v(t+\Delta t) = v(t) + f(t)/m\,\Delta t$
    - Repeat above operations
- More accurate integrators (better energy conservation)
  - Verlet Algorithm (Verlet J. Chem. Phys. 1967)

consider Taylor's expansions:

$$x(t\pm\Delta t) = x(t) \pm v(t)\,\Delta t + 1/2m\, f(t)\,\Delta t^2 \pm 1/6\, d^3x/dt^3\,\Delta t^3 + O(\Delta t^4)$$

Adding expansion $x(t+\Delta t)$ and $x(t-\Delta t)$ and rearrange:

$$x(t+\Delta t) = 2x(t) - x(t-\Delta t) + f(t)/m\,\Delta t^2 + O(\Delta t^4)$$

Subtracting expansion $x(t+\Delta t)$ and $x(t-\Delta t)$ and rearrange:

$$v(t) = [x(t+\Delta t) - x(t-\Delta t)/(2\Delta t) + O(\Delta t^2)$$

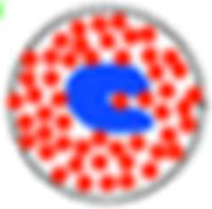
velocities lag

# Solvent and Periodic Boundary Conditions

*Vacuum*

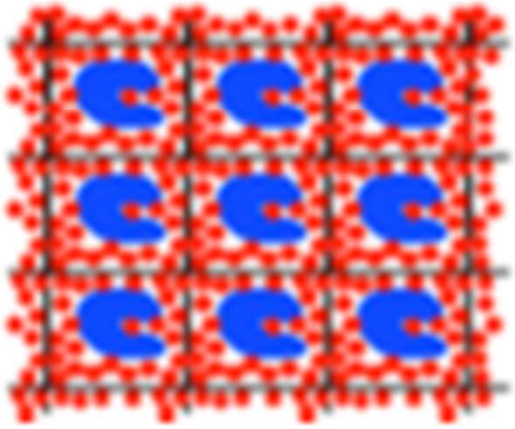
- Surface effects (surface tension)
- No dielectric screening

*Droplets*



- Still surface effects (at water – vacuum interface)
- Only partial dielectric screening
- Evaporation of the solvent

***Periodic:*** *system is surrounded by copies of itself*



Advantage:

- No surface effects

Disadvantage:

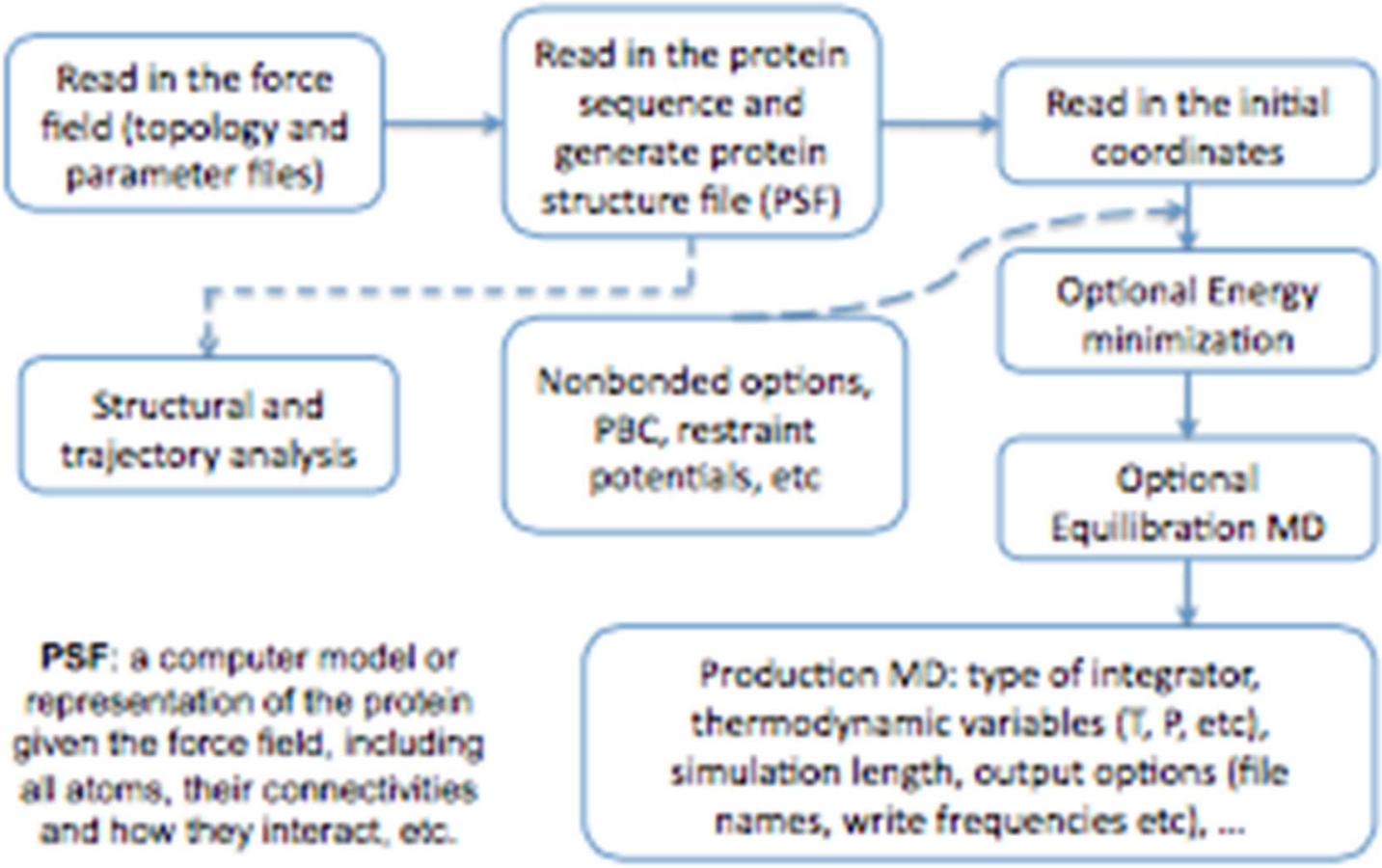
- Artificial periodicity
- High effective concentration

van Gunsteren Angew Chem Int Ed (2006)

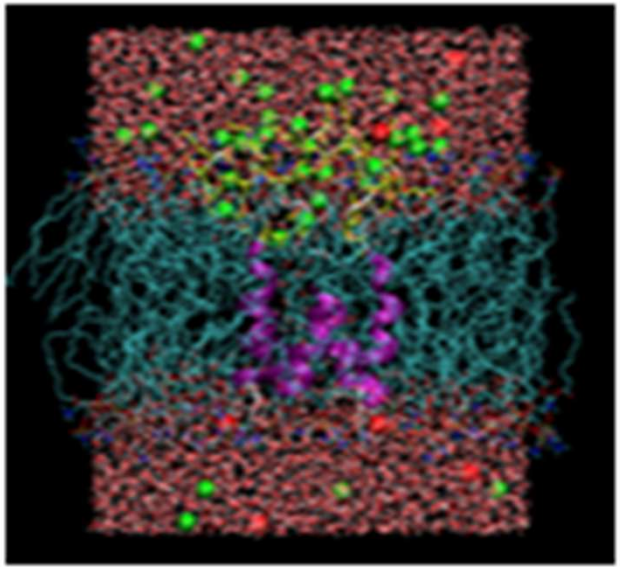
# Controlling Thermodynamic Variables

- MD generate statistical ensembles that connect microscopic details to macroscopic/thermodynamic properties
- NVE (microcanonical - Entropy rules!)
- NVT (Canonical - Helmholtz free energy is relevant, A)
  - temperature $T = \sum m\langle v2\rangle/(3k_B)$
- NPT (Isothermal-isobaric - Gibbs free energy is relevant, G)
  - P=kinetic + virial contributions
- Thermostats, barostats, etc., allow one to choose appropriate ensembles
  - Following Nose', Hoover, Evans and others...
  - See Brooks, Curr. Opin. Struct. Biol., 5, 211(1995)]

# Basic Flow of a MD Simulation



# Why is MD so slow?



Channel-forming peptides
in a fully solvated membrane bilayer;
Channel: 1795 atoms; All: 26254 atoms

**Simulated Time**

1 ns ($10^{-9}$ s)
(500,000 MD steps)

**CPU Time**

~200 hours ($10^{6}$ s)

**Wall Time**

~1 days ($10^{5}$ s) / 8 CPUs

- very small time step required
  - $\delta t$ ~ fs ($10^{-15}$ s)
- interactions between thousands of atoms need to be computed

NOTE, the basic scaling of the Verlet's integrator:
is detemined by the computation of forces $O(N^2)$

# Biological Time Scale

- Bond vibrations — 1 fs ($10^{-15}$ s)
- Sugar repuckering — 1 ps ($10^{-12}$ s)
- DNA bending — 1 ns ($10^{-9}$ s)
- Domain movement — 1 ms ($10^{-6}$ s)
- Base pair opening — 1 ms ($10^{-3}$ s)

- Transcription — 2.5 ms / nucleotide
- Protein synthesis — 6.5 ms / amino acid
- Protein folding — ~ 10 s (speed limit: μs)
- RNA lifetime — ~ 300 s

Simulation time should exceed the time scale of interest by ~10-fold !

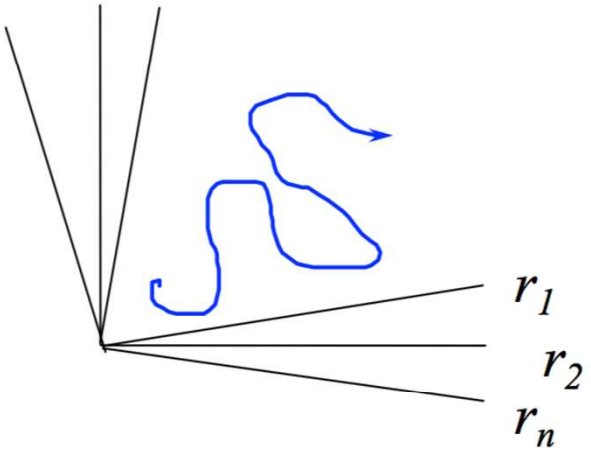
WHY? How many points you need to resolve a sin function?

# Molecular Simulations

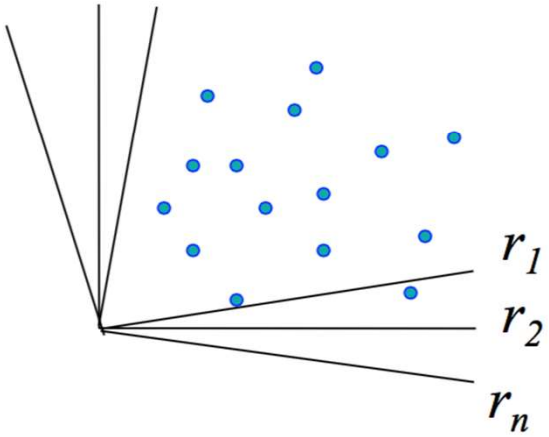
(Notes by Berend Smit)

MOLSIM/CECAM School http://www.acmm.nl/molsim/molsim2020/pictures/MolSim2020-GroupPicture.png

➡ **Molecular dynamics**: solve equations of motion




➡ **Monte Carlo**: importance sampling



# 4. Molecular Dynamics

## 4.1. Introduction

## 4.2. Basics

## 4.3. Liouville formulation

## 4.4. Multiple time steps

Theory:

$$F = m\frac{d^2r}{dt^2}$$

- Compute the forces on the particles
- Solve the equations of motion
- Sample after some # of timesteps

**Initialization**

- Total momentum should be zero (no external forces)
- Temperature rescaling to desired temperature
- Particles start on a lattice

**Force calculations**

- Periodic boundary conditions
- Order NxN algorithm,
- Order N: neighbor lists, linked cell
- Truncation and shift of the potential

**Integrating the equations of motion**

- Velocity Verlet
- Kinetic energy

**Algorithm 3 (A Simple Molecular Dynamics Program)**

```
program md                          simple MD program

call init                           initialization
t=0
do while (t.lt.tmax)                MD loop
   call force(f,en)                 determine the forces
   call integrate(f,en)             integrate equations of motion
   t=t+delt
   call sample                      sample averages
enddo
stop
end
```

**Algorithm 4 (Initialization of a Molecular Dynamics Program)**

```
subroutine init                 initialization of MD program
sumv=0
sumv2=0
do i=1,npart
  x(i)=lattice_pos(i)           place the particles on a lattice
  v(i)=(ranf()-0.5)             give random velocities
  sumv=sumv+v(i)                velocity center of mass
  sumv2=sumv2+v(i)**2           kinetic energy
enddo
sumv=sumv/npart                 velocity center of mass
sumv2=sumv2/npart               mean-squared velocity
fs=sqrt(3*temp/sumv2)           scale factor of the velocities
do i=1,npart                    set desired kinetic energy and set
  v(i)=(v(i)-sumv)*fs           velocity center of mass to zero
  xm(i)=x(i)-v(i)*dt            position previous time step
enddo
return
end
```

## Algorithm 5 (Calculation of the Forces)

```
subroutine force(f,en)                    determine the force
en=0                                      and energy
do i=1,npart
   f(i)=0                                 set forces to zero
enddo
do i=1,npart-1
  do j=i+1,npart                          loop over all pairs
     xr=x(i)-x(j)
     xr=xr-box*nint(xr/box)               periodic boundary conditions
     r2=xr**2
     if (r2.lt.rc2) then                  test cutoff
       r2i=1/r2
       r6i=r2i**3
       ff=48*r2i*r6i*(r6i-0.5)            Lennard-Jones potential
       f(i)=f(i)+ff*xr                    update force
       f(j)=f(j)-ff*xr
       en=en+4*r6i*(r6i-1)-ecut           update energy
     endif
  enddo
enddo
return
end
```

# The Lennard-Jones potentials

- The Lennard-Jones potential

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

- The truncated Lennard-Jones potential

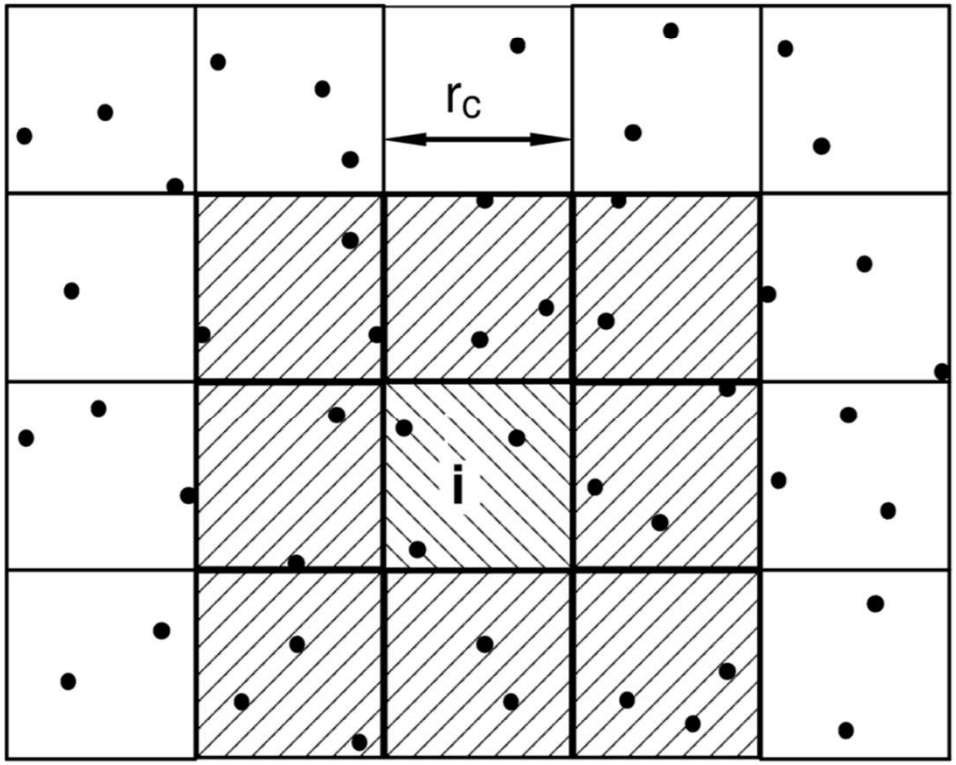
$$U^{LJ}_{TR}(r) = \begin{cases} U^{LJ}(r) & r \le r_c \\ 0 & r > r_c \end{cases}$$

- The truncated and shifted Lennard-Jones potential

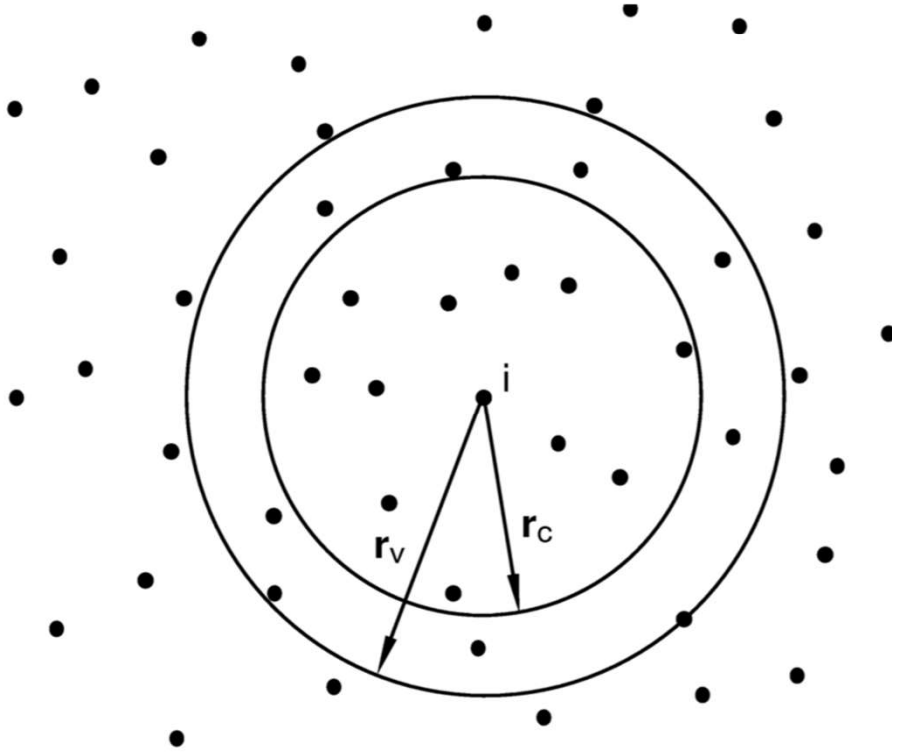
$$U^{LJ}_{TR-SH}(r) = \begin{cases} U^{LJ}(r) - U^{LJ}(r_c) & r \le r_c \\ 0 & r > r_c \end{cases}$$

# Saving CPU-time

## Cell list




## Verlet-list



## Algorithm 6 (Integrating the Equations of Motion)

```
subroutine integrate(f,en)                integrate equations of motion
sumv=0
sumv2=0
do i=1,npart                              MD loop
   xx=2*x(i)-xm(i)+delt**2*f(i)           Verlet algorithm (4.2.3)
   vi=(xx-xm(i))/(2*delt)                 velocity (4.2.4)
   sumv=sumv+vi                           velocity center of mass
   sumv2=sumv2+vi**2                      total kinetic energy
   xm(i)=x(i)                             update positions previous time
   x(i)=xx                                update positions current time
enddo
temp=sumv2/(3*npart)                      instantaneous temperature
etot=(en+0.5*sumv2)/npart                 total energy per particle
return
end
```

# Equations of motion

We can make a Taylor expansion for the positions:

$$r\left(t+\Delta t\right)=r\left(t\right)+\frac{dr\left(t\right)}{dt}\Delta t+\frac{d^2r\left(t\right)}{dt^2}\frac{\Delta t^2}{2!}+O\left(\Delta t^3\right)$$

The simplest form (Euler):

$$r\left(t+\Delta t\right)=r\left(t\right)+v\left(t\right)\Delta t+O\left(\Delta t^2\right)$$

$$v\left(t+\Delta t\right)=v\left(t\right)+m\frac{df\left(t\right)}{dt}\Delta t$$

We can do better!

We can make a Taylor expansion for the positions:

$$r(t+\Delta t) = r(t) + \frac{dr(t)}{dt}\Delta t + \frac{d^2r(t)}{dt^2}\frac{\Delta t^2}{2!} + \frac{d^2r(t)}{dt^2}\frac{\Delta t^3}{3!} + O(\Delta t^4)$$

$$r(t-\Delta t) = r(t) - \frac{dr(t)}{dt}\Delta t + \frac{d^2r(t)}{dt^2}\frac{\Delta t^2}{2!} - \frac{d^2r(t)}{dt^2}\frac{\Delta t^3}{3!} + O(\Delta t^4)$$

When we add the two:

$$r(t+\Delta t) + r(t-\Delta t) = 2r(t) + \frac{d^2r(t)}{dt^2}\Delta t^2 + O(\Delta t^4)$$

Verlet algorithm $$r(t+\Delta t) = 2r(t) - r(t-\Delta t) + f(t)\frac{\Delta t^2}{m} + O(\Delta t^4)$$

Verlet algorithm: $$r(t+\Delta t)=2r(t)-r(t-\Delta t)+f(t)\frac{\Delta t^2}{m}+O(\Delta t^4)$$

Velocity Verlet algorithm

$$r(t+\Delta t)=r(t)+v(t)\Delta t+f(t)\frac{\Delta t^2}{2m}+O(\Delta t^4)$$

$$v(t+\Delta t)=v(t)+\frac{\Delta t}{2m}\left[f(t+\Delta t)+f(t)\right]$$

to see the equivalence:

$$r(t+2\Delta t)=r(t+\Delta t)+v(t+\Delta t)\Delta t+f(t+\Delta t)\frac{\Delta t^2}{2m}$$

$$r(t)=r(t+\Delta t)-v(t)\Delta t-f(t)\frac{\Delta t^2}{2m}$$

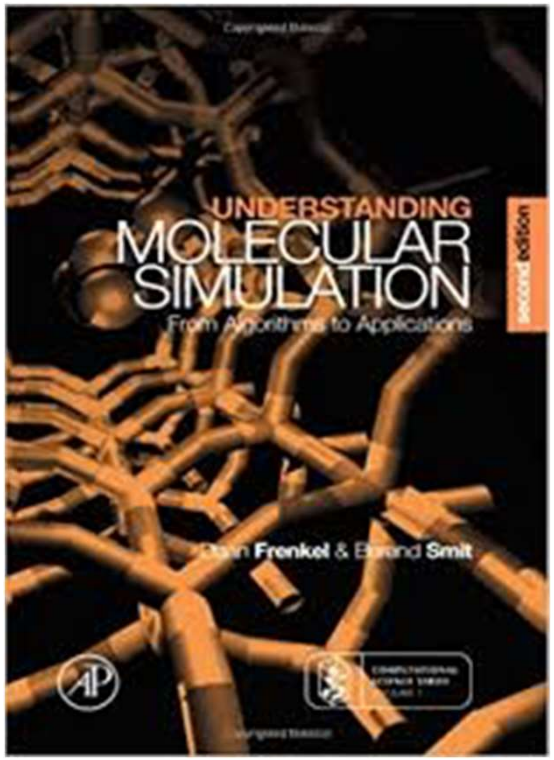
adding the two

$$r(t+2\Delta t)=2r(t+\Delta t)-r(t)+\left[v(t+\Delta t)-v(t)\right]\Delta t+\left[f(t+\Delta t)-f(t)\right]\frac{\Delta t^2}{2m}$$

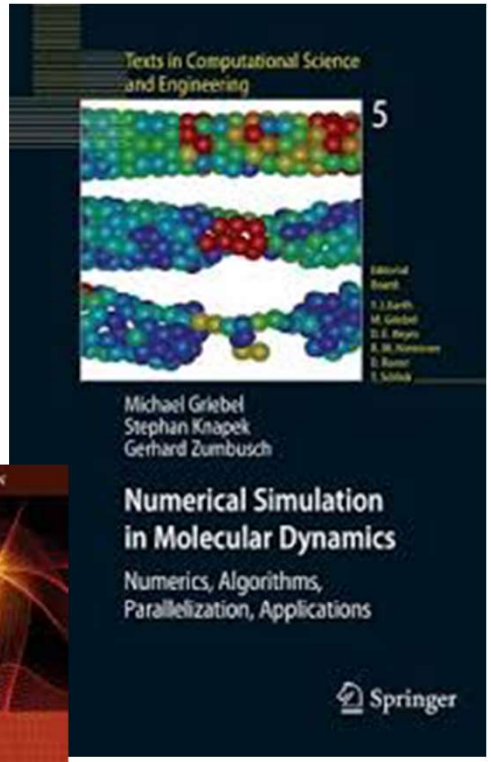
with $$v(t+\Delta t)=v(t)+\frac{\Delta t}{2m}\left[f(t+\Delta t)+f(t)\right]$$

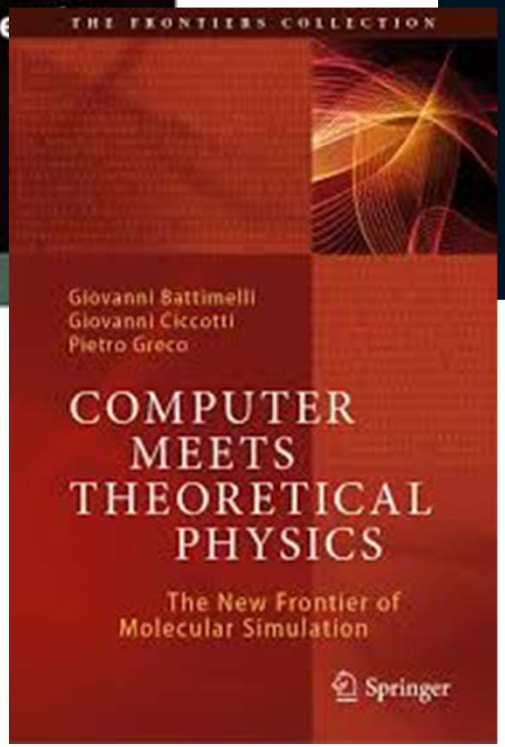
$$r(t+2\Delta t)=2r(t+\Delta t)-r(t)+f(t+\Delta t)\frac{\Delta t^2}{m}$$

A NICE MOVIE BY Giff Ransom Strongly recommended as a recapitulation
https://slideplayer.com/slide/4795252/







# Where to start ?

- GROMACS http://www.gromacs.org
- NAMD VMD https://www.ks.uiuc.edu/Research/namd/
- K. Hinsen MMTK https://github.com/khinsen/MMTK
- . Marchi ORAC http://www.chim.unifi.it/orac/
  - CECAM ICTP SCHOOL https://www.cecam.org/workshop-details/60

MolSim School (http://www.acmm.nl/molsim/molsim2020/ )

HOMEWORK

Familiarize with the GROMACS Package: tutorial n.1 at

http://www.mdtutorials.com/gmx/index.html