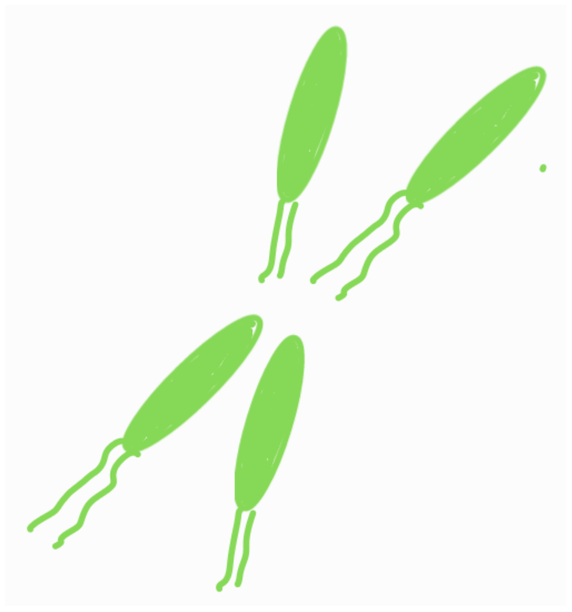
Slides will be distributed by the organizers

ACTIVE MATTER
Millennium Nucleus Physics of Active Matter

Computational Modeling of Active Systems

Rodrigo Soto
Universidad de Chile



School on Active Matter, ICTP-SAIFR, Sao Paulo, 2024

Computational Modeling of Active Systems

Contents

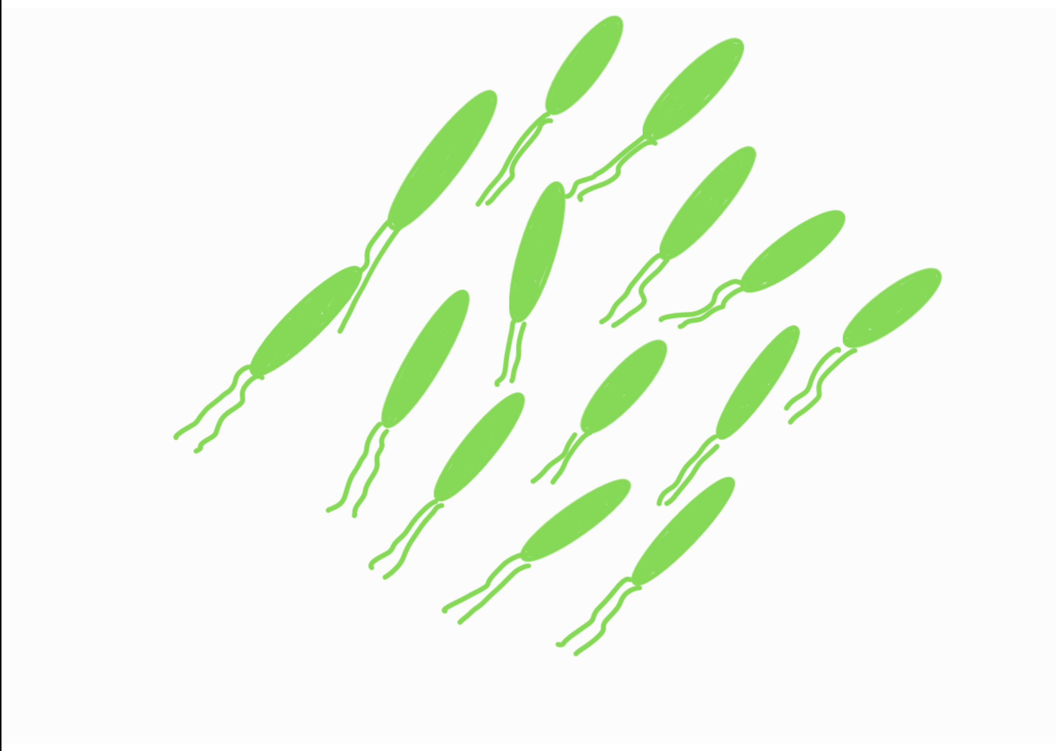
What will we see?

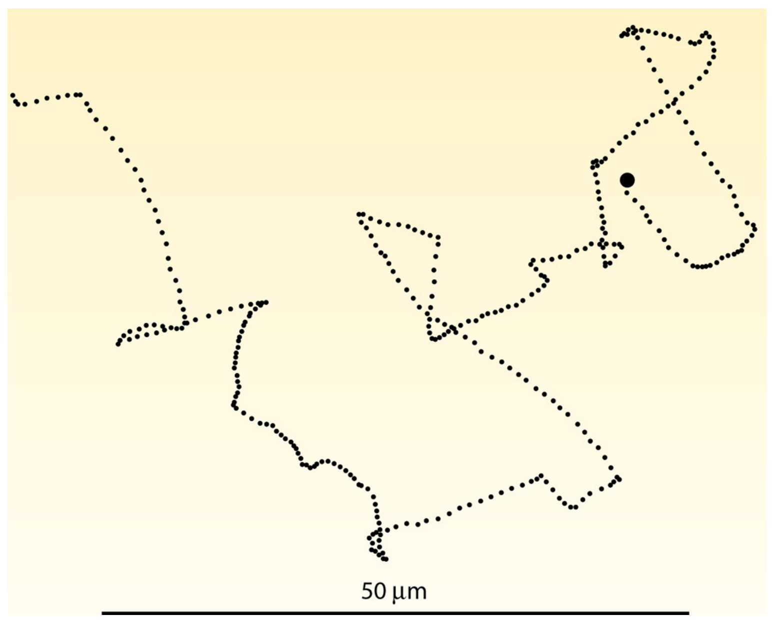
- Models and their implementation
- Observables. Why and what we get from them

What will we not see?

- Efficient programming

Self-propelled particles (SPP)





Berg et al. 1972

Self-propelled particles (SPP)

Model for bacteria, migrating cells, Janus colloids and other non-inertial agents

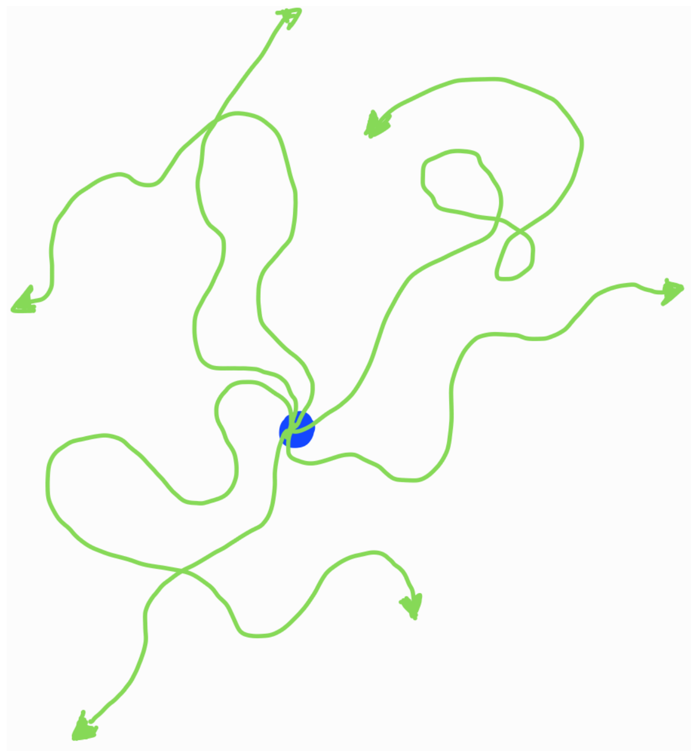
Key elements:

- **Self propulsion**: velocity $\overrightarrow{V} = V_0\hat{n}$
- **Persistence**: $\hat{n}$ changes rarely
  - To model some bacteria changes by tumbles: Run-and-tumble particles (RTP)
  - For Janus colloids, changes by rotational diffusion: Active Brownian Particles (ABP)

Active Brownian Particles

The director $\hat{n}$ diffuses on the unit sphere, described by the Fokker-Planck equation for the probability

$$\frac{\partial P(\hat{n},t)}{\partial t} = D_r \nabla^2_{\hat{n}} P$$



in 2D

$$\frac{\partial P(\phi,t)}{\partial t} = D_r \frac{\partial^2 P}{\partial \phi^2}$$

in 3D

$$\frac{\partial P(\theta,\phi,t)}{\partial t} = D_r \left[\frac{1}{\sin\theta} \frac{\partial}{\partial\theta} \left(\sin\theta \frac{\partial P}{\partial \theta} \right) + \frac{1}{\sin^2\theta} \frac{\partial^2 P}{\partial \phi^2} \right]$$

Active Brownian Particles

In simulations, instead of describing the probability distribution, we simulate a **realization** of $\hat{n}(t)$ and average over all (really many) possible realizations.

There is a theorem: Fokker-Planck is equivalent to Langevin

$$\frac{d\hat{n}}{dt} = \sqrt{2D_r}\vec{\xi} \times \hat{n}$$

The cross product guarantees that $\hat{n}$ remains unitary

Here $\vec{\xi}$ is a white noise (Gaussian stochastic process with)

$\langle \xi_i(t) \rangle = 0, \quad \langle \xi_i(t)\xi_k(t') \rangle = \delta(t-t')\delta_{ik}, \quad \langle \vec{\xi}(t)\hat{n}(t') \rangle = 0$ if $t > t'$

Active Brownian Particles

In 2D, it is direct to show (homework) that

$$\frac{d\hat{n}}{dt} = \sqrt{2D_r}\vec{\xi} \times \hat{n}$$

reduces to

$$\frac{d\phi}{dt} = \sqrt{2D_r}\xi$$

where ξ is a Gaussian stochastic process with

$$\langle \xi(t) \rangle = 0, \quad \langle \xi(t)\xi(t') \rangle = \delta(t - t'), \quad \langle \xi(t)\phi(t') \rangle = 0 \text{ if } t > t'$$

It is a stochastic differential equation (SDE)

Integration of SDE

Consider a simple stochastic differential equation

$$\frac{dx}{dt} = f(x) + \sqrt{2D}\,\xi(t)$$

Time discretization: $t_n = n\Delta t; \quad x_n = x(t_n)$

We integrate the equation from t_n to t_{n+1}

$$x_{n+1} - x_n = \underbrace{\int_{t_n}^{t_{n+1}} f(x(t))dt}_{F_n} + \sqrt{2D}\underbrace{\int_{t_n}^{t_{n+1}} \xi(t)dt}_{I_n}$$

Euler scheme $F_n = f(x_n)\Delta t$

I_n ????

Integration of SDE

$$I_n \equiv \int_{t_n}^{t_{n+1}} \xi(t)dt$$

- Are stochastic variables
- Sum of Gaussian, then Gaussian (we only need the mean and covariance)
- $\langle I_n \rangle = 0$
- If $n \neq m$, then $\langle I_n I_m \rangle = 0$ because $\langle \xi(t)\xi(t') \rangle = \delta(t - t')$

$$\begin{aligned}\langle I_n I_n \rangle &= \int_{t_n}^{t_{n+1}} ds_1 \int_{t_n}^{t_{n+1}} ds_2 \langle \xi(s_1)\xi(s_2) \rangle \\ &= \int_{t_n}^{t_{n+1}} ds_1 \int_{t_n}^{t_{n+1}} ds_2 \delta(s_1 - s_2) = \int_{t_n}^{t_{n+1}} ds_1 \\ &= \Delta t\end{aligned}$$

In summary I_n are independent Gaussian variables of zero mean an variance Δt

Integration of SDE

$$\frac{dx}{dt} = f(x) + \sqrt{2D}\,\xi(t)$$

$$x_{n+1} - x_n = f(x_n)\Delta t + \sqrt{2D_r}I_n = f(x_n)\Delta t + \sqrt{2D_r\Delta t}J_n$$

with $\langle J_n^2 \rangle = 1$

Algorithm:

```
For (many realizations)
   x = Initial condition
   For t in time
      J = random.normal(0,1)
      x = x + f(x)*Dt + sqrt(2*Dr*Dt)*J
```

Tumbles

With rate ν a new director $\hat{n}'$ is chosen at random with probability $w(\hat{n}, \hat{n}') = \hat{w}(\hat{n} \cdot \hat{n}') = \hat{w}(\alpha)$

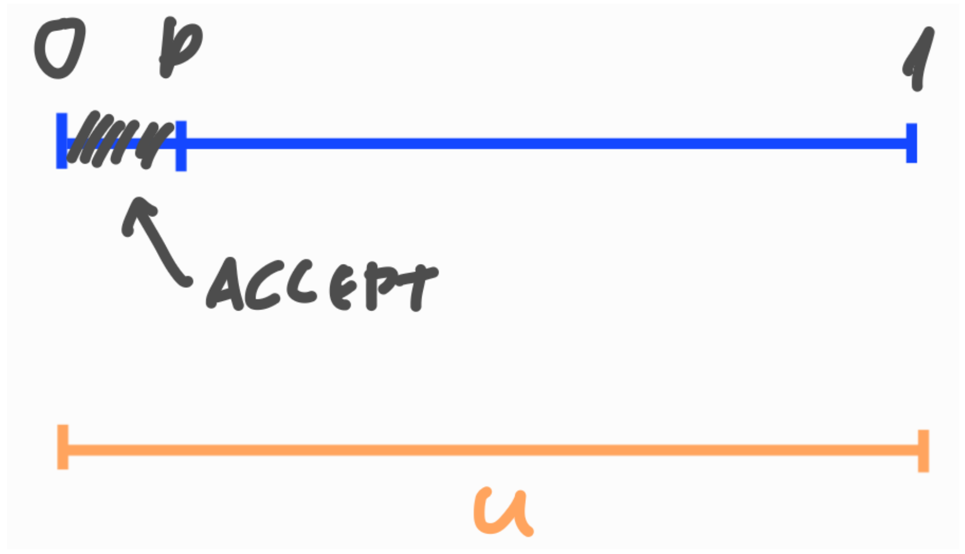
How to sample the rate and $\hat{w}$?

A **rate** ν means that in a small Δt the probability of the event is $p = \nu \Delta t \ll 1$

For that, take u random number in [0,1)

Algorithm:

```
For t in time
   x = x + (something)
   u = random.uniform(0,1)
   if (u < nu*dt)
      make_tumble
```



Tumbles

How to choose the new $\hat{n}$?

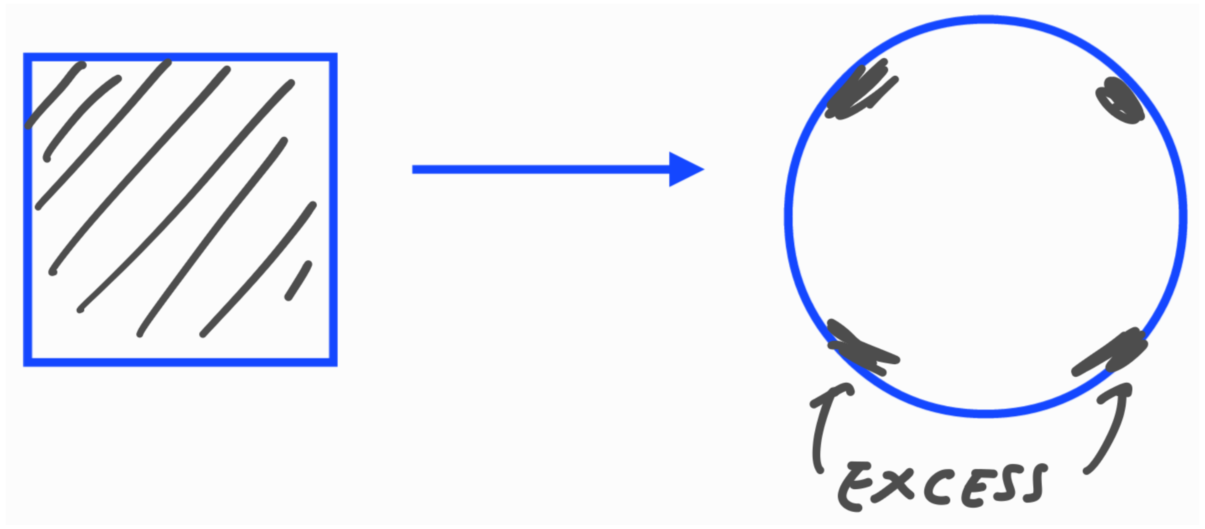
1) If the distribution is uniform

In 2D, simple:

```
phi = random.uniform(0,2*pi)
```

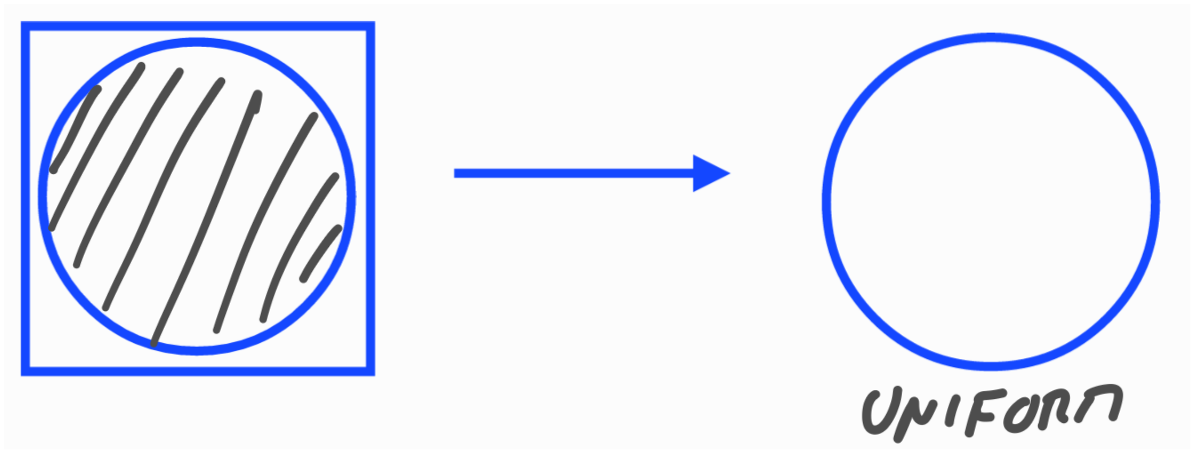
In 3D, wrong algorithm

```
nx = random.uniform(-1,1)
ny = random.uniform(-1,1)
nz = random.uniform(-1,1)
n = sqrt(nx**2 + ny**2 + nz**2)
nx = nx/n
ny = nx/n
nz = nx/n
```



In 3D, correct algorithm

```
do
    nx = random.uniform(-1,1)
    ny = random.uniform(-1,1)
    nz = random.uniform(-1,1)
    n = sqrt(nx**2 + ny**2 + nz**2)
while(n>1)
nx = nx/n
ny = nx/n
nz = nx/n
```



Also, other methods using change of variable or Gaussian variables

Tumbles

How to choose the new $\hat{n}$?

2) If the distribution is not uniform

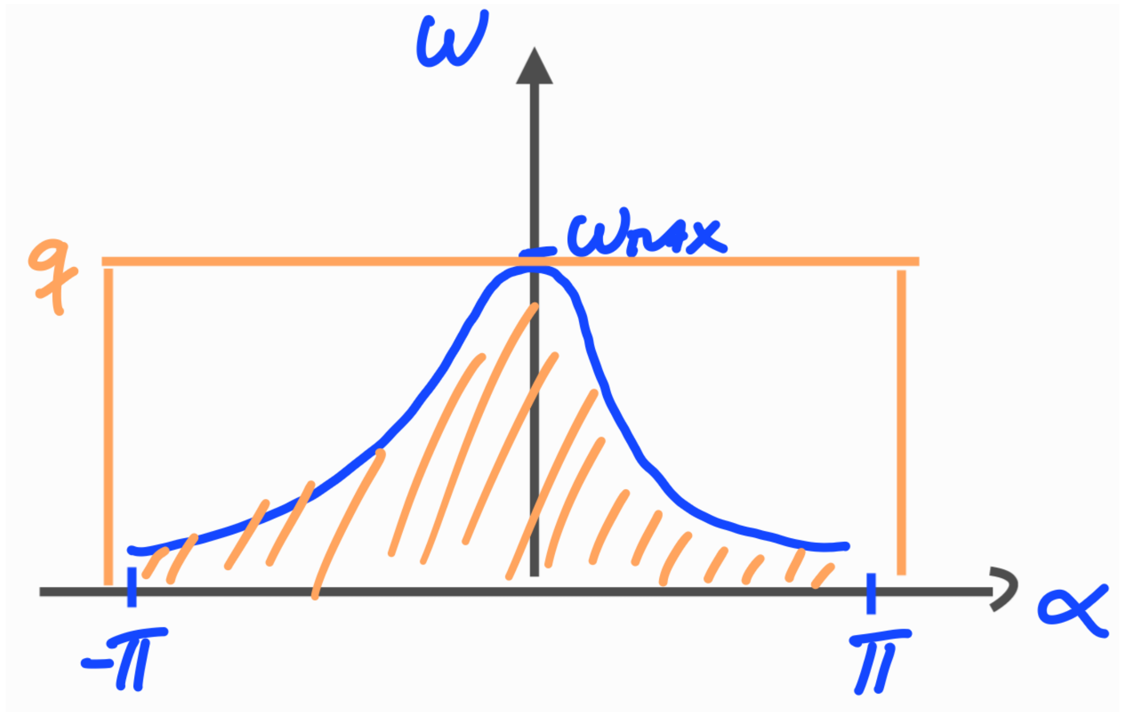
If possible apply the method of change of variables

If not, use the rejection method (Monte Carlo)

For example, in 2D, with $w(\alpha)$

Algorithm:

```
do
    q = random.uniform(0,wmax)
    alpha = random.uniform(-pi,pi)
while(q > w(alpha))
phi=phi+alpha
```



In 3D, with $w(\hat{n} \cdot \hat{n}')$

```
do
    q = random.uniform(0,wmax)
    hatnprime = random.uniformunitvector()
while(q > w(hatn. hatnprime))
hatn= hatnprime
```

Interactions

Up to here, independent ABPs or RTP

In active colloids, particle are spherical and hard, effectively impenetrable

Equations of motion with inertia

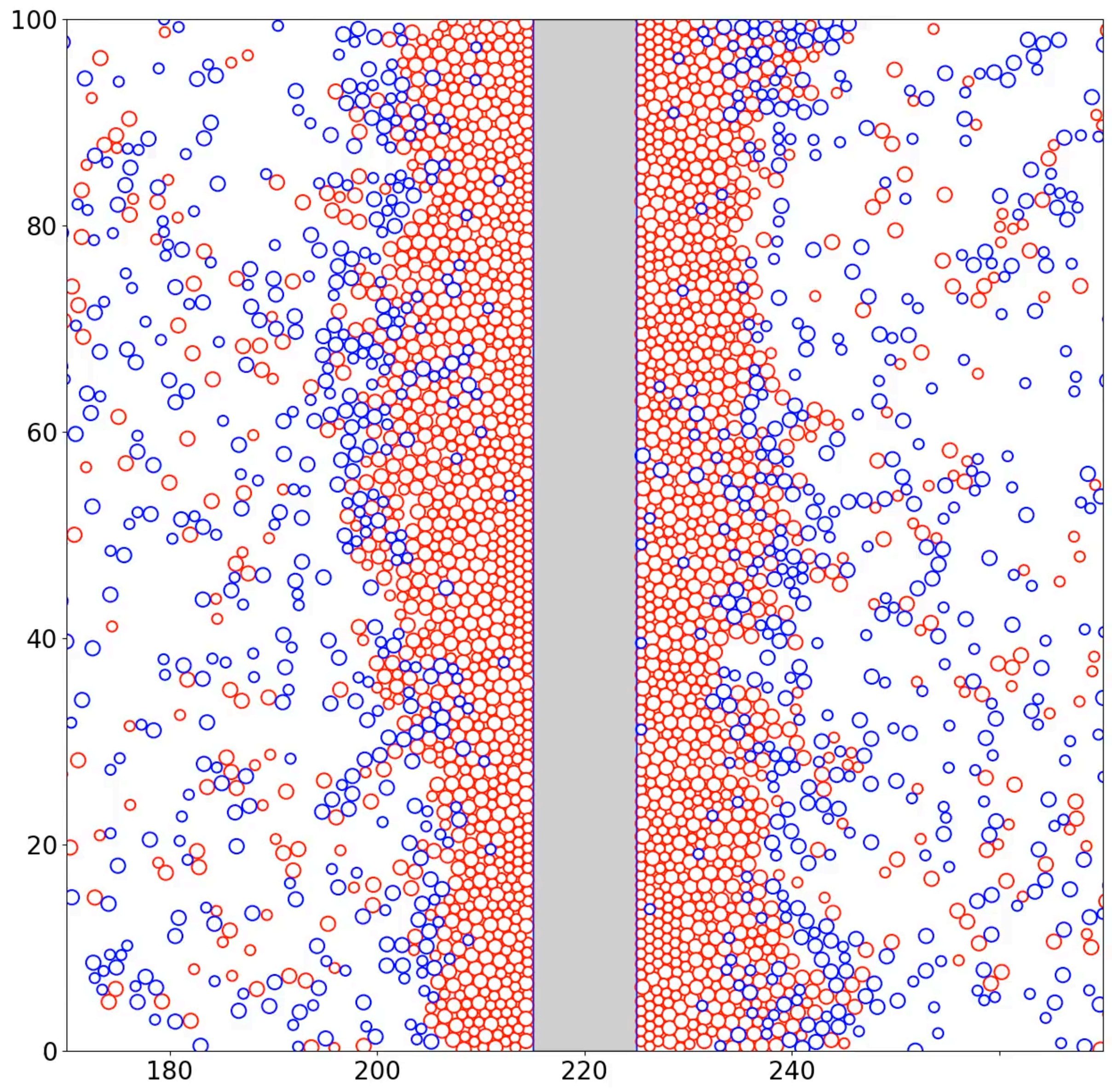
$$m\dot{V}_i = -\gamma V_i + F_0 \hat{n}_i - \nabla_i U_T$$

with $U_T = \sum_{i,j} U(\vec{r}_i - \vec{r}_j)$

If inertia in neglected ($m \to 0$)

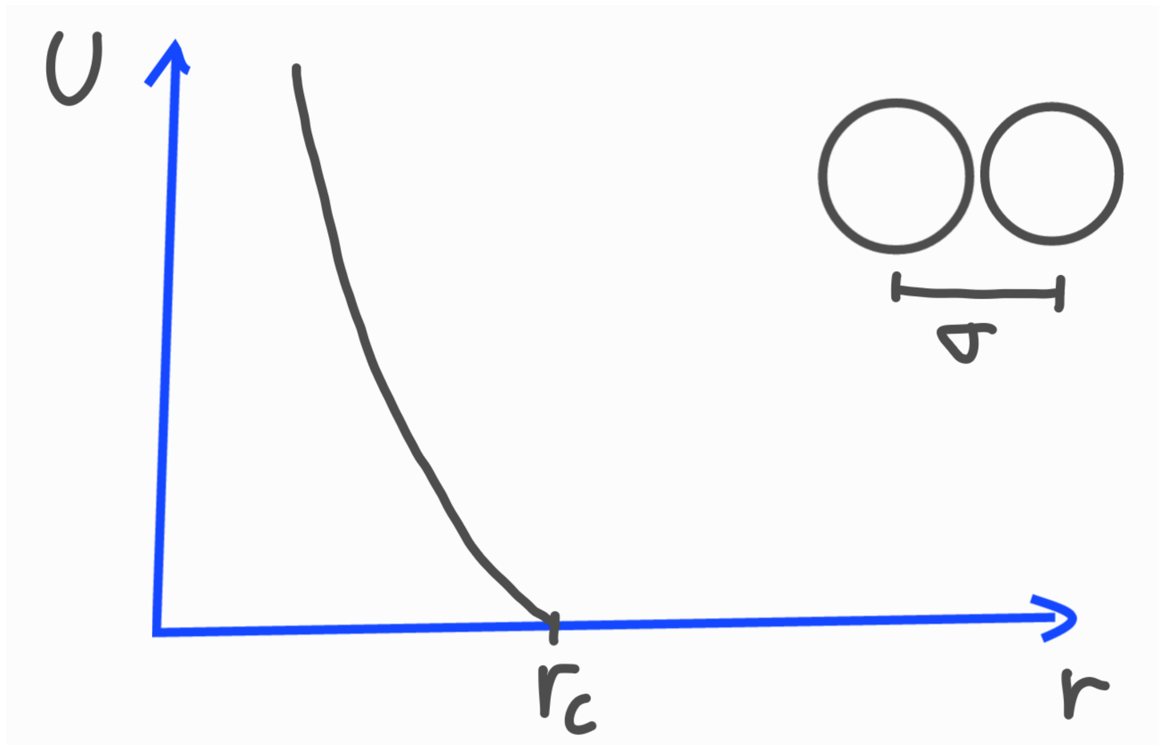
$$V_i = (F_0/\gamma)\hat{n}_i - \nabla_i (U_T/\gamma)$$

is the interacting SPP model with $V_0 = F_0/\gamma$ and $U_T/\gamma \to U_T$ with units of L^2/T (diffusion coefficient)



Interactions

Simple models for the interaction potential



Elastic $U(r) = \begin{cases} \frac{k}{2}(\sigma - r)^2, & r < \sigma \\ 0, & \sim \end{cases}$

WCA (aka LJ) $U(r) = \begin{cases} 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right], & r < 2^{1/6}\sigma \\ 0, & \sim \end{cases}$

Interactions

Brute force algorithm

```
For t in time
   for i in NumberOfParticles
      V[i] = V_0*hatn[i]                        } $O(N)$

   for i in NumberOfParticles
      for j<i
         f = F(r[i]-r[j])
         V[i] = V[i] + f                        } $O(N^2)$
         V[j] = V[j] - f

   for i in NumberOfParticles
      r[i] = r[i] + V[i]*dt                     } $O(N)$
      hatn[i] = (something / ABP or RTP)
```

It is too slow for large systems

Interactions

If the force has a finite range (as WCA)

Naïve solution

```
For t in time
   for i in NumberOfParticles
      V[i] = V_0*hatn[i]

   for i in NumberOfParticles
      for j<i
         if(rij < rc)
            f = F(r[i]-r[j])
            V[i] = V[i] + f
            V[j] = V[j] - f

   for i in NumberOfParticles
      r[i] = r[i] + V[i]*dt
      hatn[i] = (something / ABP or RTP)
```

Still $O(N^2)$, slightly faster

Efficient solution

Linked cells

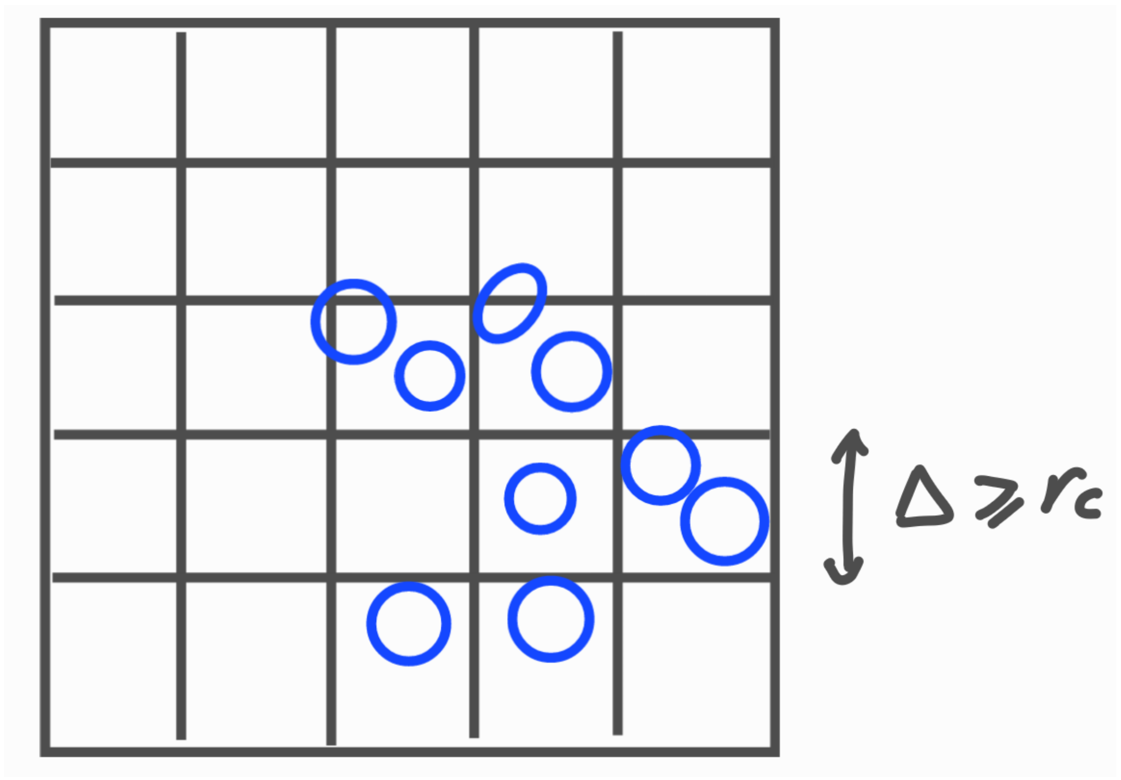
Every particle interacts at most with neighbor cells

```
For t in time
    AllocateParticlesInCells()

    for i in NumberOfParticles
        V[i] = V_0*hatn[i]

    for i in NumberOfParticles
        for j in Neighborhood(i)
            if(rij < rc)
                f = F(r[i]-r[j])
                V[i] = V[i] + f
                V[j] = V[j] - f

    for i in NumberOfParticles
        r[i] = r[i] + V[i]*dt
        hatn[i] = (something / ABP or RTP)
```



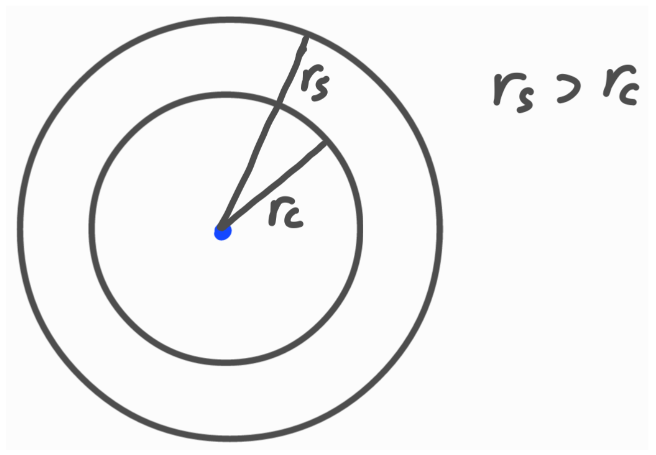
Also Verlet lists.

For each particle the list with neighbors up to r_s is built every several time steps

```
For t in time
    if (t several)
        BuildVerletLists()

    for i in NumberOfParticles
        V[i] = V_0*hatn[i]

    for i in NumberOfParticles
        for j in VerletList(i)
            if(rij < rc)
```



Measurements

We will see

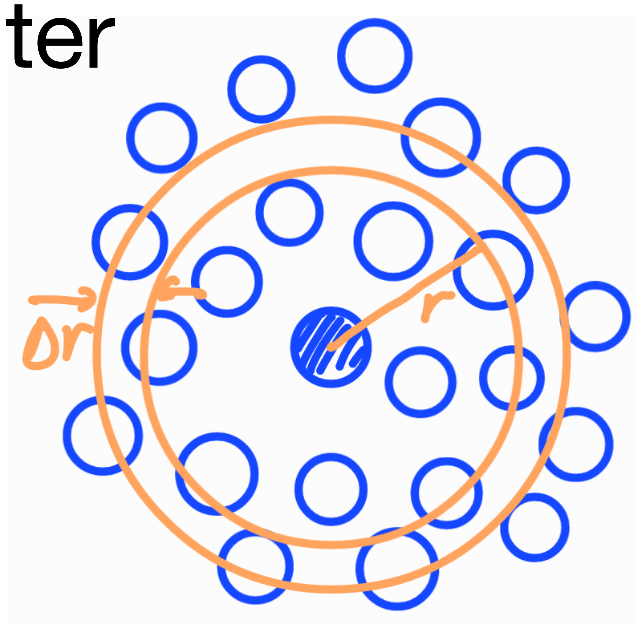
- Pair correlation functions
- Spatial fields
- Mean square displacement
- Temporal correlation functions

In all cases, we will implement **on-the-fly** measurements

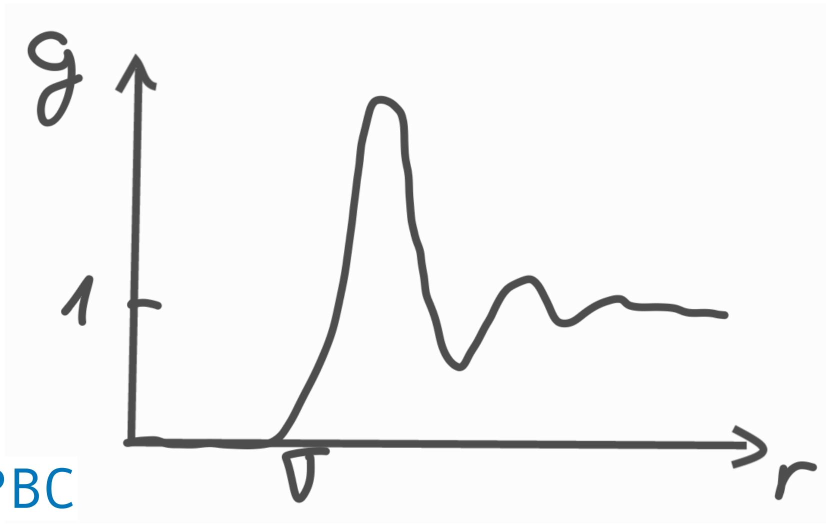
Avoid recording the full trajectory for postprocessing: unnecessary and too heavy.

Pair correlation functions

Measure the probability to find another particle at a certain distance from any, taken as center



We use a binning distance Δr



```
For t in time
   … Simulate …

   if (time to measure)
      NMedG += 1
      for i in NumberOfParticles
         for j in Neighborhood(i)
            distance = |r[i] - r[j]| # considering PBC
            bin = int(distance/Deltar)
            Gacum[bin] += 1
# after the end of the simulation
for bin
   g[bin] = Gacum[bin]/(NMedG*N*2*pi*(bin*Deltar)*Deltar*rho)
```

Pair correlation functions

But also, we can measure the angular correlation. There are three angles ϕ_1, ϕ_2, and ψ

What to measure? $\langle \cos(\phi_1 - \phi_2) \rangle$, $\langle \cos\phi_1 \cos\phi_2 \rangle$, ...

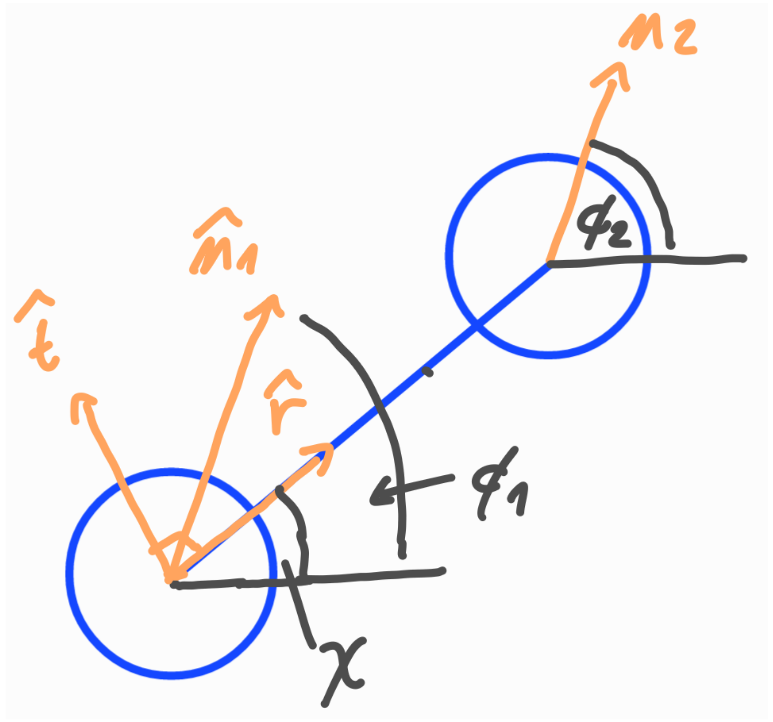
The angle ψ fixes the reference axis, angles are measured w/r to it

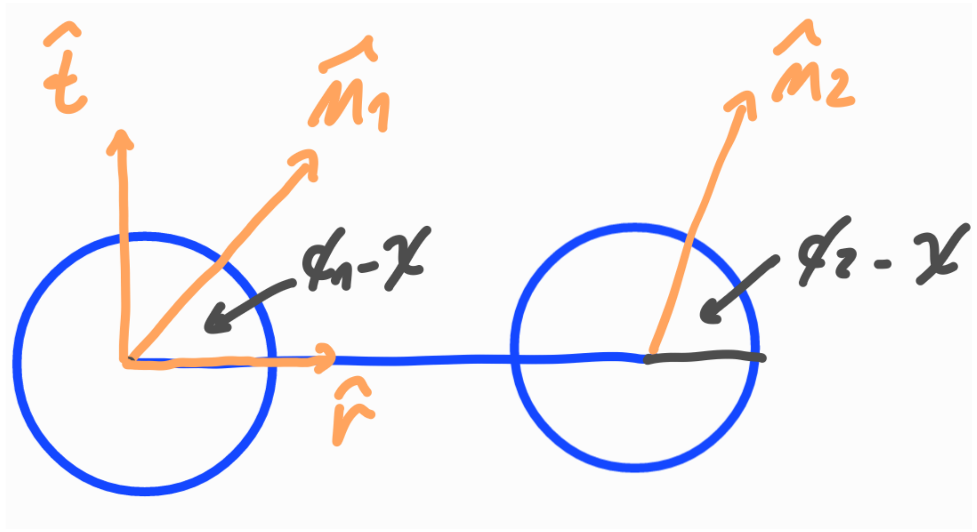
$$C_{\|} = \langle \cos(\phi_1 - \psi)\cos(\phi_2 - \psi) \rangle = \langle (\hat{n}_1 \cdot \hat{r})(\hat{n}_2 \cdot \hat{r}) \rangle$$

$$C_{\perp} = \langle \sin(\phi_1 - \psi)\sin(\phi_2 - \psi) \rangle = \langle (\hat{n}_1 \cdot \hat{t})(\hat{n}_2 \cdot \hat{t}) \rangle$$

Note that $\langle \cos(\phi_1 - \psi)\sin(\phi_2 - \psi) \rangle = 0$ by symmetry

and $C_\alpha = \langle \hat{n}_1 \cdot \hat{n}_2 \rangle = C_{\|} - C_{\perp}$





Pair correlation functions

The angle ψ fixes the reference axis, angles are measured w/r to it

$$C_{\parallel} = \langle \cos(\phi_1 - \psi)\cos(\phi_2 - \psi) \rangle = \langle (\hat{n}_1 \cdot \hat{r})(\hat{n}_2 \cdot \hat{r}) \rangle$$

$$C_{\perp} = \langle \sin(\phi_1 - \psi)\sin(\phi_2 - \psi) \rangle = \langle (\hat{n}_1 \cdot \hat{t})(\hat{n}_2 \cdot \hat{t}) \rangle$$

and $C_\alpha = \langle \hat{n}_1 \cdot \hat{n}_2 \rangle = C_{\parallel} - C_{\perp}$

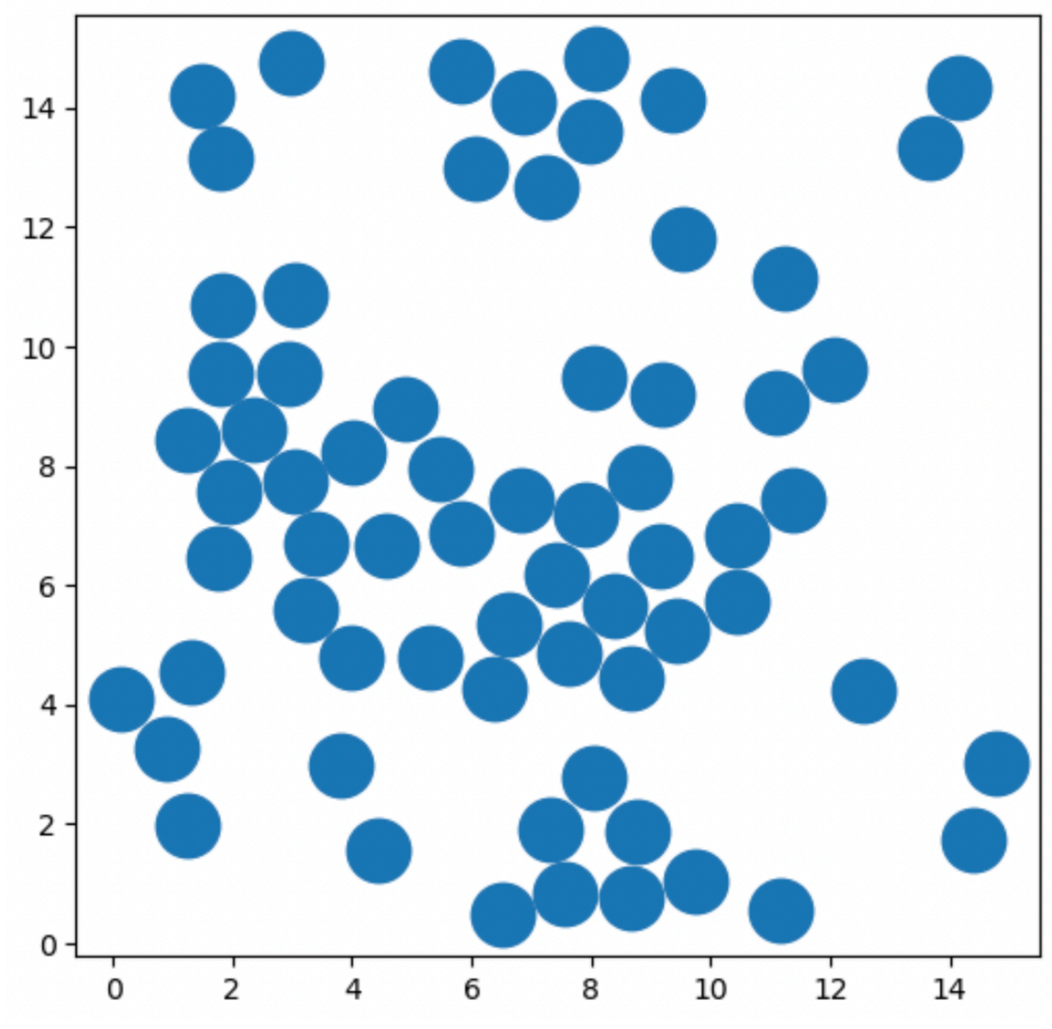
With the same binning

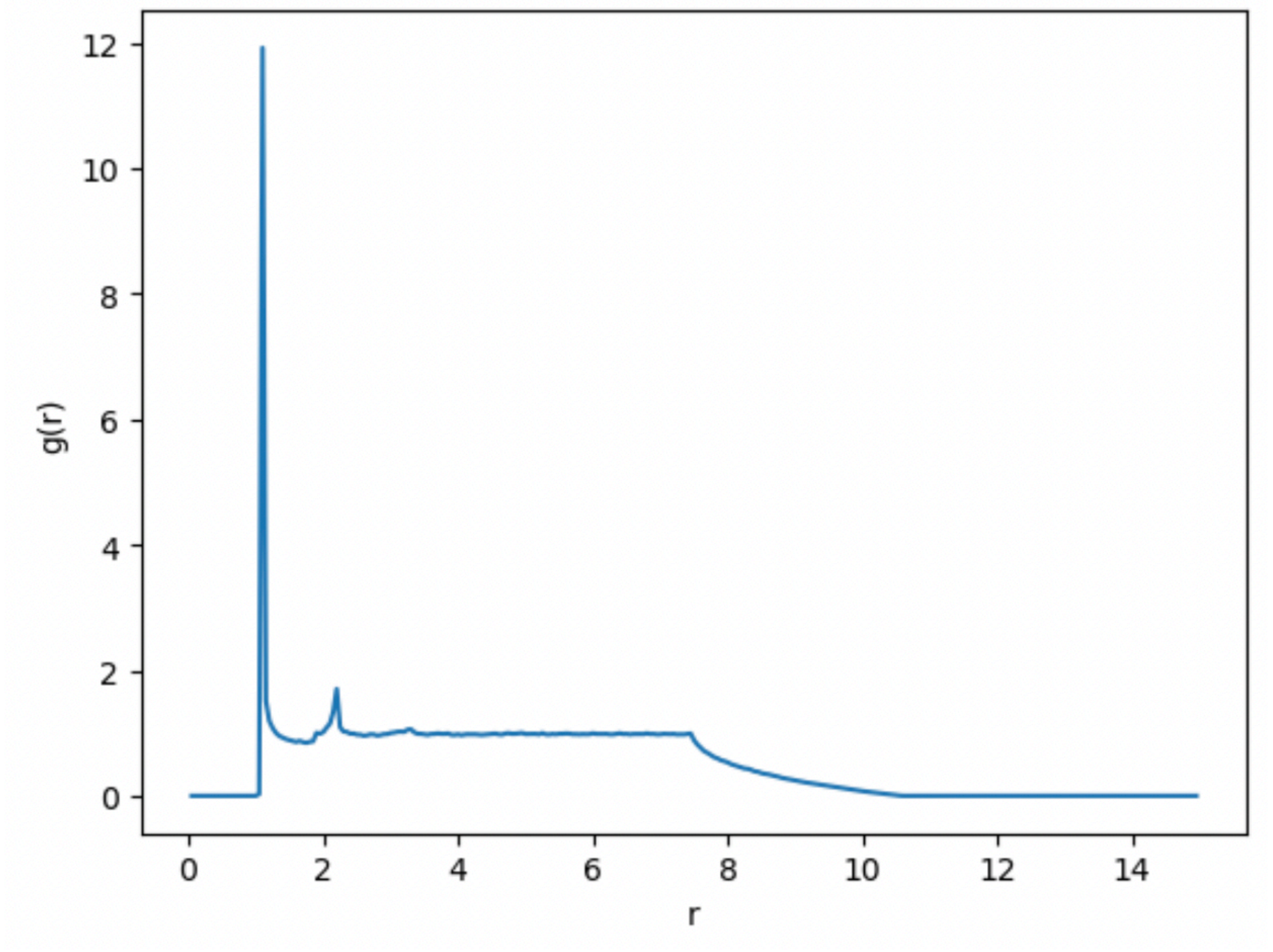
```
For t in time
    … Simulate …
    if (time to measure)
        NMedG += 1
        for i in NumberOfParticles
            for j in Neighborhood(i)
                distance = |r[i] - r[j]| # considering PBC
                bin = int(distance/Deltar)
                Gacum[bin] += 1
                Gparlallelacum[bin] += cos*cos
                Gperpendicularacum[bin] += sin*sin
# after the end of the simulation
for bin
    g[bin] = Gacum[bin]/(NMedG*N*2*pi*(bin*Deltar)*Deltar*rho)
    Gparallel[bin] = Gparlalleacum[bin]/Gacum[bin]
    Gperpendicular[bin] = Gperpendicularacum[bin]/Gacum[bin]
```

Pair correlation functions

Simulation with

Lx=Ly=15   n0=0.3   N=67   Dr=0.1


0
2
4
6
8
10
12
14


g(r)
r
0
2
4
6
8
10
12
14