PHYS-E0412, Computational Physics, Lecture 5, 5 February 2019

Ilja Makkonen

Aalto University
School of Science

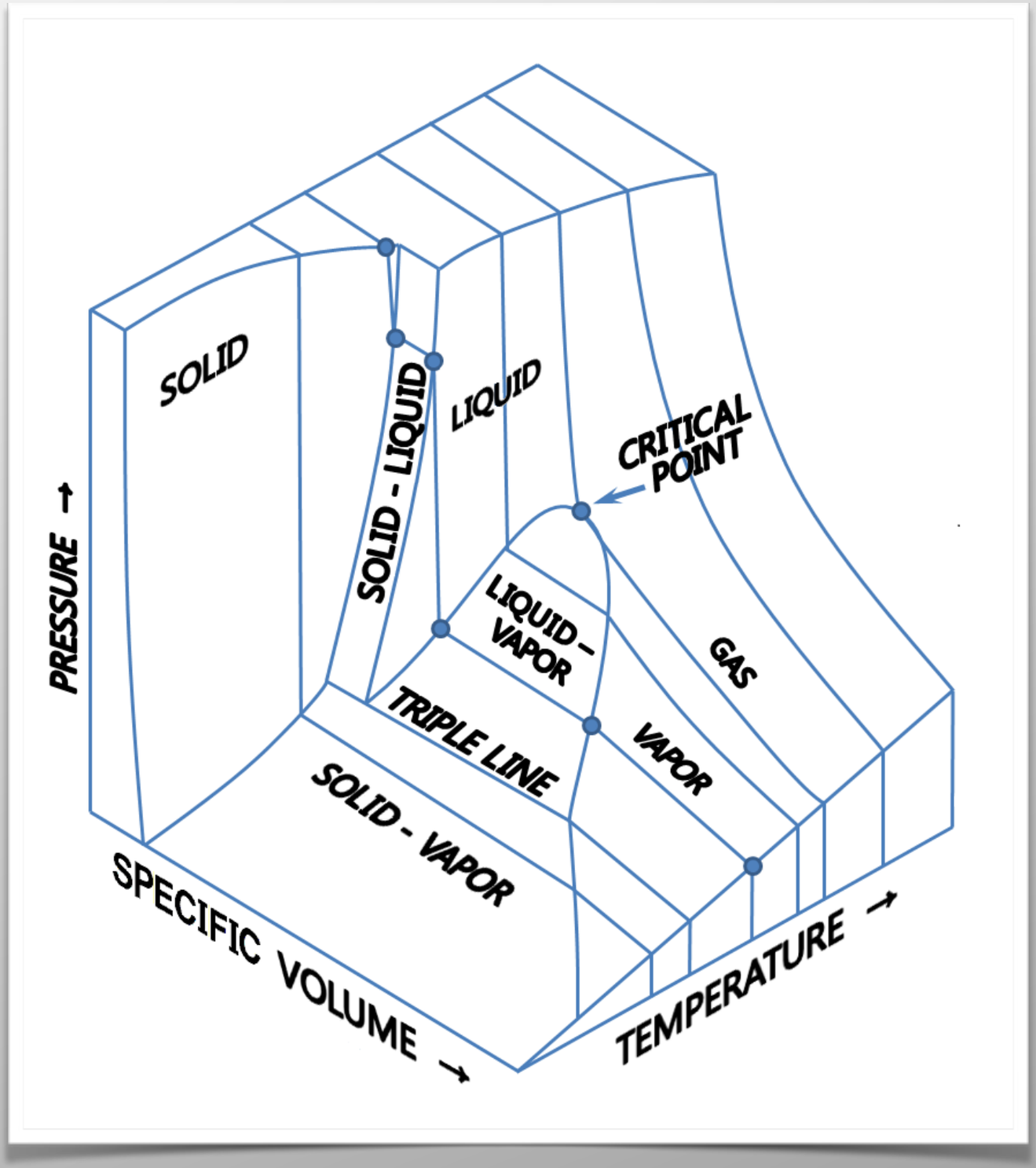
Learning objectives for week 5

- The basic idea of molecular dynamics simulations in microcanonical (NVE) and canonical (NVT) ensembles
- Lennard Jones potential
- Time propagation: Verlet and velocity Verlet algorithms
- Sampling and analysing some important observables such as radial distribution function (RDF) and mean-square displacement (MSD)

Homework 5:

Simulating phases of the Lennard-Jones potential using Velocity Verlet algorithm for Molecular dynamics. Focus on the RDS.

Phases of matter



How can we study real systems?

We can do molecular dynamics

Molecular dynamics

Study of large number of classical particles (atoms, molecules).

Numerically integrate equations of motion.

Example: N particles in 3D, $3N$ equations to be solved:

$$m_i \ddot{\mathbf{r}}_i = \mathbf{F}_i = -\nabla_{\mathbf{r}_i} V \ ,$$

where $\mathbf{F}_i$ is force on particle i and V is potential.

No quantum effects.

Forces are not exactly known.

Only finite system sizes possible. (PBC)

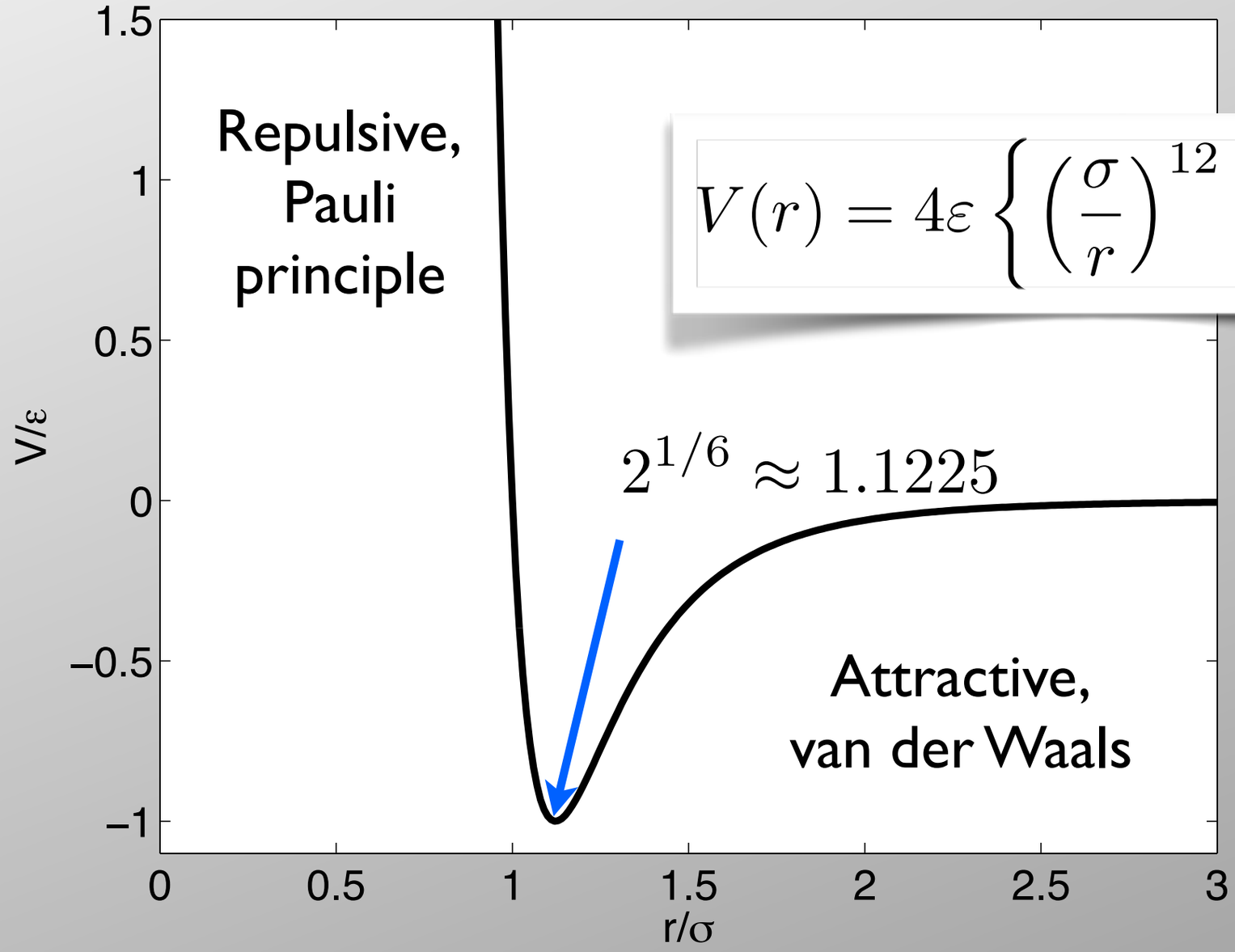
Potential approximated as

$$V = \sum_i v_1(\mathbf{r}_i) + \sum_{i<j} v_2(\mathbf{r}_i, \mathbf{r}_j) + \sum_{i<j<k} v_3(\mathbf{r}_i, \mathbf{r}_j, \mathbf{r}_k) + \dots \ ,$$

where v_1 is external potential, v_2 is dominating pair potential etc.

Lennard-Jones pair potential

Interaction potential between two atoms ($V(r_{ij}) = v_2(\mathbf{r}_i, \mathbf{r}_j)$):



$$V(r) = 4\varepsilon \left\{ \left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right\}$$

Force on i'th particle:

$$\mathbf{F}_i = -\nabla_{\mathbf{r}_i} V = \frac{48\varepsilon}{\sigma^2} \sum_{j \neq i}^{N} (\mathbf{r}_i - \mathbf{r}_j) \left\{ \left(\frac{\sigma}{r_{ij}}\right)^{14} - \frac{1}{2} \left(\frac{\sigma}{r_{ij}}\right)^{8} \right\}$$

Verlet algorithm

Setting $m = 1$ and considering just one coordinate x.

Equation to solve is

$$\ddot{x}(t) = F[x(t)] \, .$$

Task to find $x(t)$ with a given initial position $x(0)$ and velocity $\dot{x}(0)$.

Taylor expansion of x at $t \pm h$ about t:

$$x(t+h) = x(t) + h\dot{x}(t) + \frac{h^2}{2} F[x(t)]$$

$$x(t-h) = x(t) - h\dot{x}(t) + \frac{h^2}{2} F[x(t)]$$

and these added up gives

$$x(t+h) = 2x(t) - x(t-h) + h^2 F[x(t)] \, .$$

Verlet has error of $\mathcal{O}(h^4)$

Velocity form $\dot{x}(t) = \dfrac{x(t+h) - x(t-h)}{2h}$ is only $\mathcal{O}(h^2)$!

Velocity Verlet

$$
\begin{aligned}
x(t+h) &= x(t) + h\dot{x}(t) + h^2 F[x(t)]/2 \\
\dot{x}(t+h) &= \dot{x}(t) + h(F[x(t+h)] + F[x(t)])/2
\end{aligned}
$$

is numerically very stable.

Velocities more accurate.

Velocity needed, e.g., for kinetic energy (K).

Equipartition theorem: energy kT/2 for each degree of freedom:

$$K = \frac{D}{2} NkT \ ,$$

where D is dimension of space.

Molecular dynamics

Initialize simulation: Give particles positions and velocities.

Do velocity-Verlet to reach equilibrium.

[Note for Verlet: At beginning we don't have $x(-h)$. Use Taylor expansion to get $x(h)$.]

Time needed to reach equilibrium depends on initial configuration and relaxation time.

Best to monitor several observables.

Continue simulation and measure physical quantities.

Note: Constant energy is not what typically is wanted. Rather, e.g, constant T.

One can scale velocities to get wanted T.

(Remember, average kinetic energy is $\frac{D}{2}kT$.)

Frictional forces: $\ddot{x} = F - \gamma\dot{x}$, and equation for motion very similarly as above.

A nice e-book
(possible project topics inside?)

Understanding Molecular Simulation : From Algorithms to Applications

Daan Frenkel and Berend Smit

(Elsevier Science and Technology, 2001)

Access from Aalto network (or with Aalto VPN)

Homework 5

PHYS-E0412 Computational Physics :: Homework 5

Due date 12.2.2019 at 10 am

Molecular dynamics

This week's problem is to simulate molecular dynamics (MD) using the Velocity Verlet iteration of N particles in a three-dimensional box with volume $V = L^3$ in periodic boundary conditions using the Lennard-Jones (LJ) pair potential where $r_m = 2^{1/6}\sigma$ is the potential minimum, and we set $\epsilon = 1$, $\sigma = 1$. An example molecular dynamics code written in Python can be found in MyCourses in the file hw5_md.py It should be useful for solving this homework.

1. Implement a function that evaluates the force based on the Lennard-Jones potential, see also lecture slides and hw5_md.py. Use the analytical formula for the force. Test that your forces are implemented correctly by testing that the total energy (kinetic + potential) is conserved in a NVE MD run for a two particle system and check that your total energy value is correct (microcanonical ensemble, i.e. constant N, V and E.). (1p)

2. Implement the sampling of the pair-correlation function (radial distribution function, RDF) $g(r)$ and implement velocity initialization at given temperature T. Test these by initializing a simulation $N = 108, \rho = 0.8442, T = 0.728$ with simple cubic lattice and plotting RDF. Explain the location of the first peak in RDF. (2p)

3. Identify the phase in previous assignment and find the two missing phases by plotting analyzing resulting radial distribution functions $g(r)$. In addition to temperature and density you should pay attention to the initial configuration. (2p)

Radial Distribution function (RDF) (or Pair Correlation function)

The RDF describes how density varies as a function of distance from a reference particle

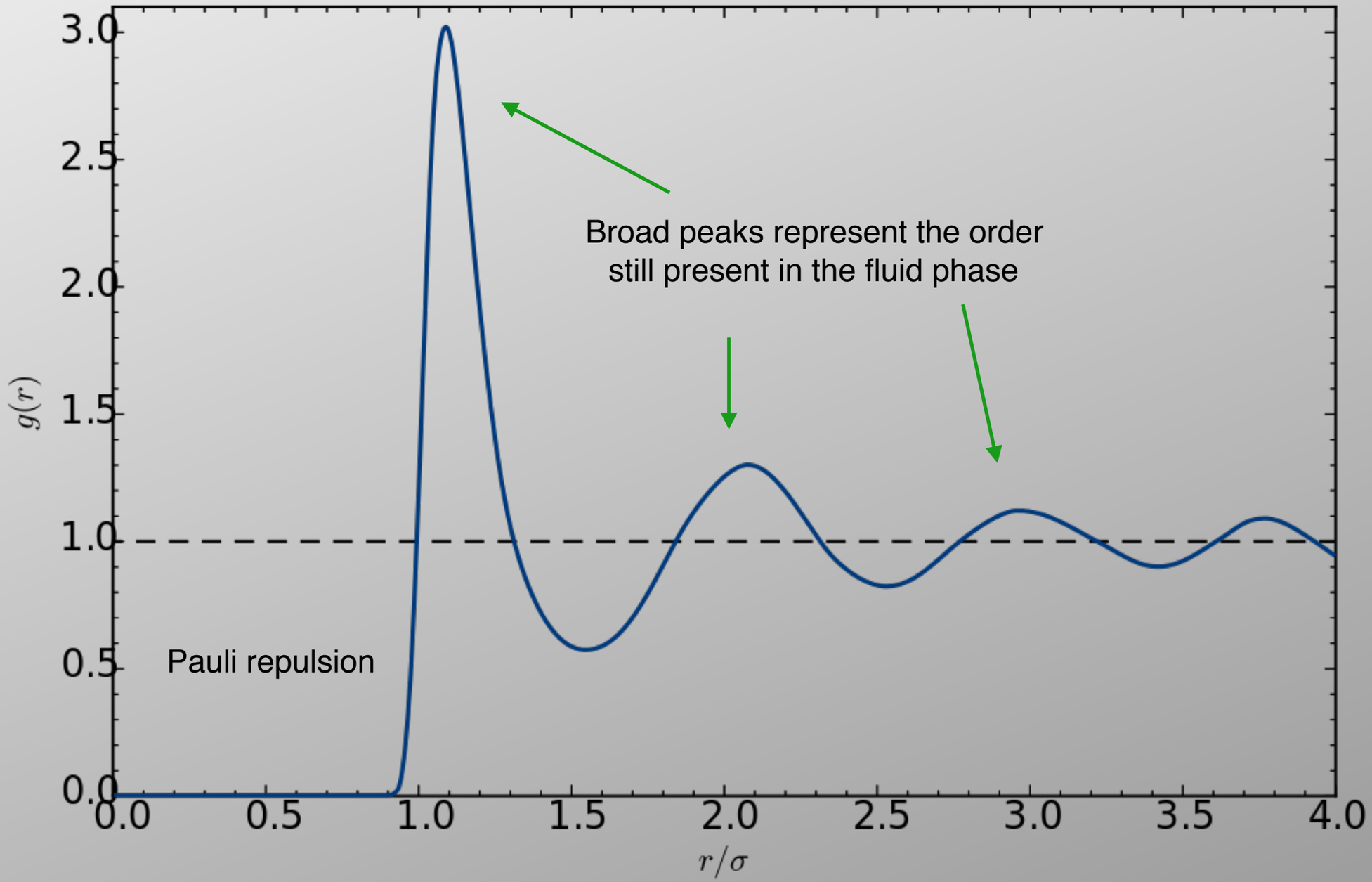
$$g(r) = \frac{1}{\rho}\langle \sum_{i \neq 0} \delta(|\mathbf{r} - \mathbf{r}_i|) \rangle$$

Here $\langle\rangle$ denotes ensemble average. Normalization

$$\rho \int g(r) d\mathbf{r} = N - 1 \approx N.$$

corresponds to the limit $g(r) \to 1$
(when/if long-range correlations die out).

Radial Distribution function (RDF) for a model Lennard-Jones fluid



Mean square displacement (MSD)

Defined as

$$\langle \Delta r(t)^2 \rangle = \frac{1}{N} \sum_{i=1}^{N} \langle (\mathbf{r}_i(t) - \mathbf{r}_i(0))^2 \rangle$$

tells how atoms diffuse in system.

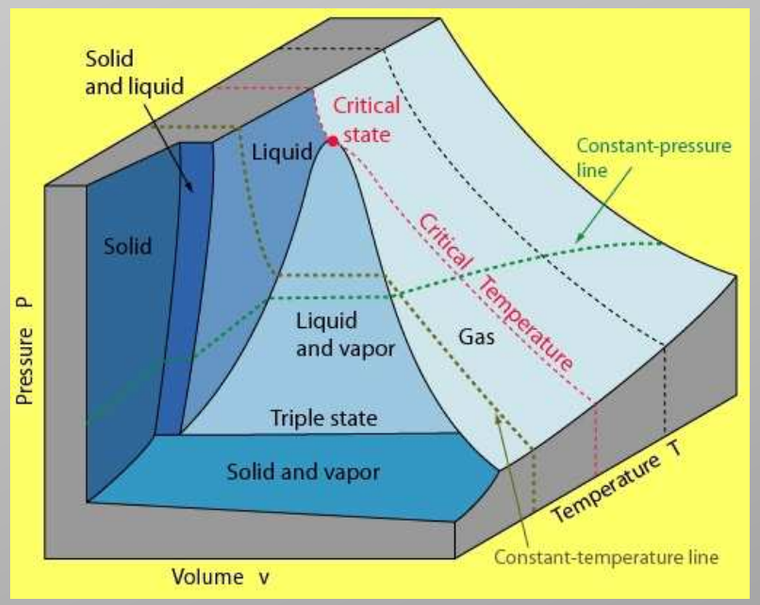
For small time, behaves as $\propto t^2$.

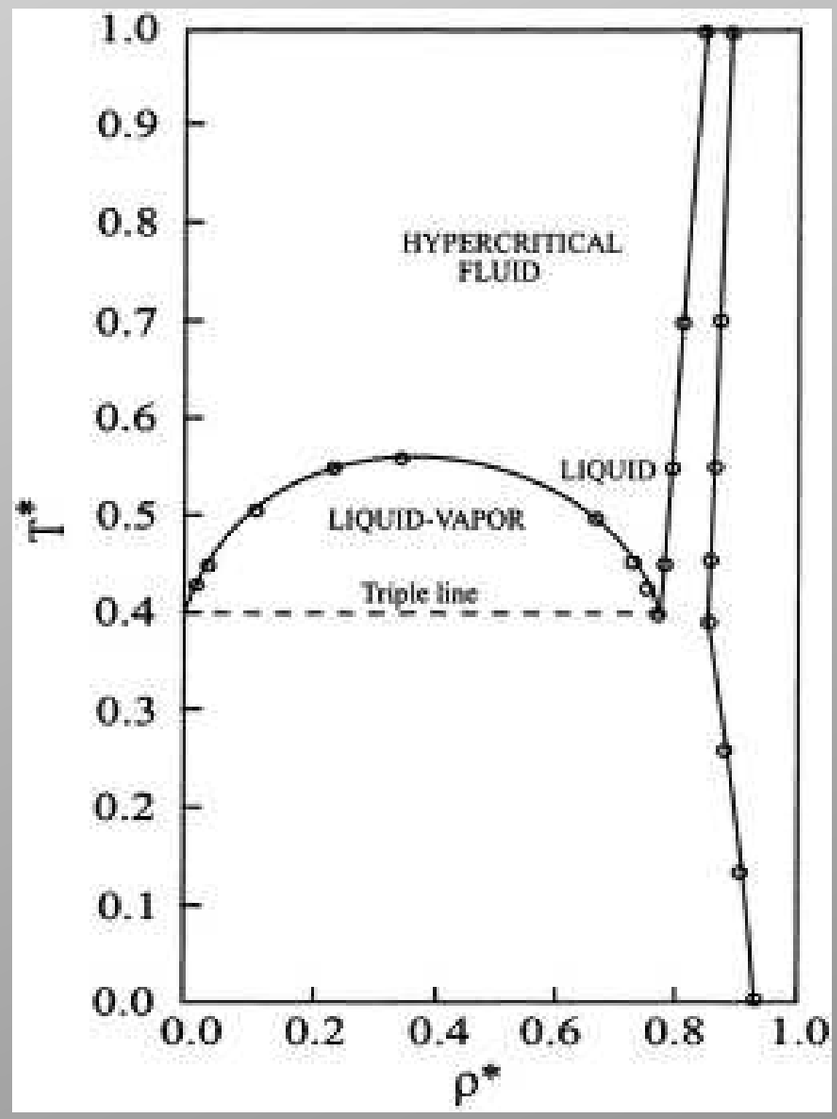
Solid: $\mathbf{r_i(t)}$ are oscillating around $\mathbf{r_i(0)} \rightarrow$ constant MSD.

Gas: $\mathbf{r_i(t)}$ grows rather linearly, MSD behaves as $\propto t^2$.

Liquid: Something between, that is MSD behaves as $\propto t$.

This can be used to determine the phase of the system.





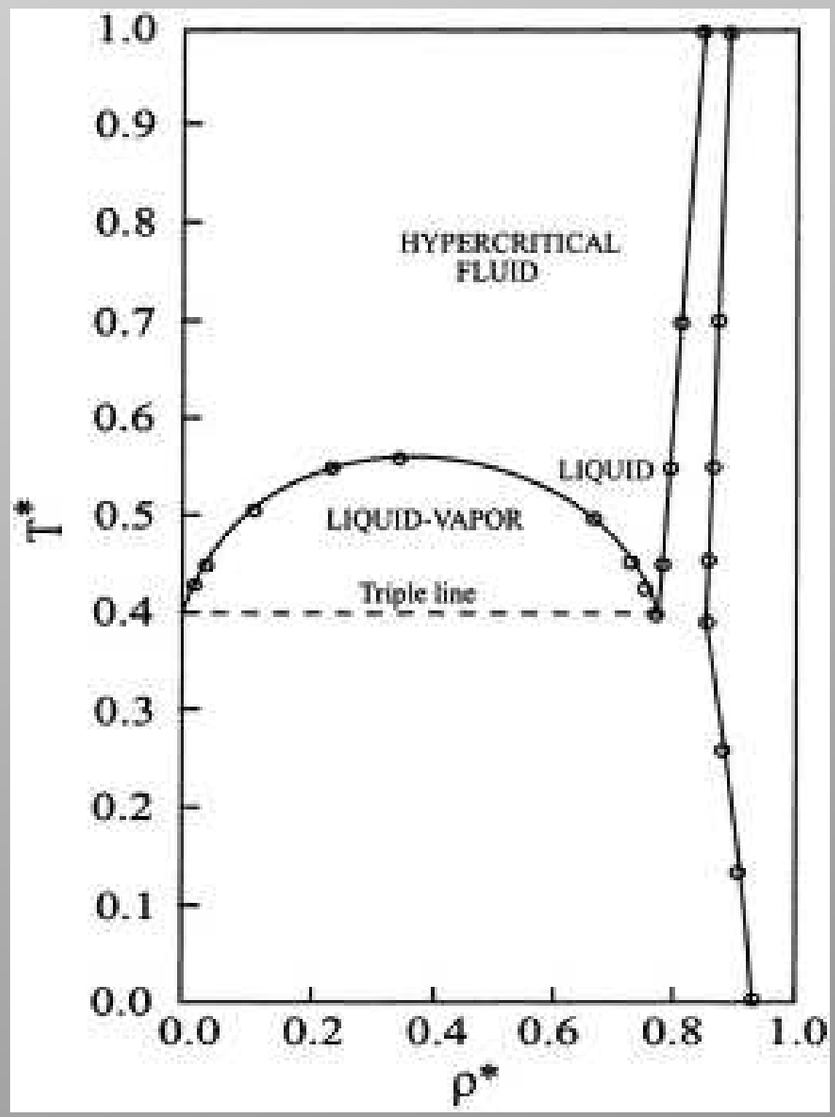
Phase diagram of 2D LJ system.

Velocity autocorrelation function

$$C(t) = \frac{1}{N}\sum_{i}^{N}\langle \mathbf{v}_i(0)\cdot\mathbf{v}_i(t)\rangle$$

- autocorrelation of particle's velocity with past times
- affected by scattering events, atomic vibration etc.
- Connection to the diffusion constant

$$D = \frac{1}{d}\int_0^\infty C(t)dt$$



Phase diagram of 2D LJ system.

INTERACTIVE WEB PAGES

- http://physics.weber.edu/schroeder/software/demos/MDv0.html (HTML5)
- http://physics.oregonstate.edu/~landaur/CPUG/CPlab/MoleDynam/md.html (Java, please note bad browser support)
- Try also the search "molecular dynamics" in Apple's App Store or Google Play

Simple model system for molecular dynamics

N point charges on a plane (2D)

Coulomb repulsion between them

Harmonic external potential to confine the particles

Potential on the first particle:

$$\sum_{i=2}^{N} \frac{1}{\sqrt{(x_i - x_1)^2 + (y_i - y_1)^2}} + x_1^2 + y_1^2$$

Total potential by summing over potentials of each particle

Force on the first particle (-grad):

$$-\left\{ \sum_{i=2}^{N} \frac{x_i - x_1}{\left((x_i - x_1)^2 + (y_i - y_1)^2\right)^{3/2}} + 2x_1, \sum_{i=2}^{N} \frac{y_i - y_1}{\left((x_i - x_1)^2 + (y_i - y_1)^2\right)^{3/2}} + 2y_1, 0 \right\}$$

Simple model system for molecular dynamics

N point charges on a plane (2D)

Coulomb repulsion between them

Harmonic external potential to confine the particles

```
function [ pot, g ] = md_potential( r )
%MD_POTENTIAL calculates potential from coordinates
%   This is now a Coulomb repulsion between particles
%   and a harmonic external potential.

N=length(r(:,1)); % assuming first index to be particle index
pot=sum(sum(r.^2)); % this is x^2+y^2 summed over all particles
for i=1:N,
   for j=i+1:N,
       pot=pot+1/norm(r(i,:)-r(j,:)); % 1/r potential between particles
   end
end
```

Simple model system for molecular dynamics

N point charges on a plane (2D)

We can calculate the force from potential

$$\mathbf{F}_i = -\nabla_{\mathbf{r}_i} V$$

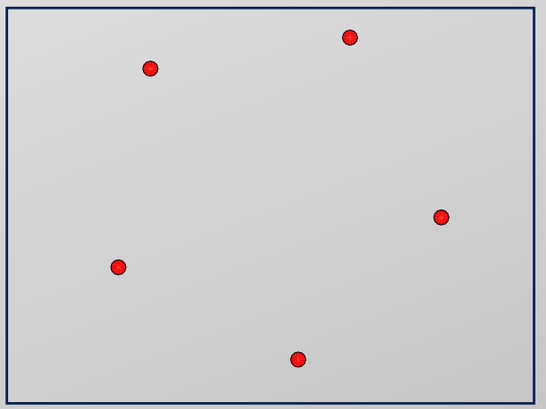
```
function [ force ] = md_force( r )
%MD_FORCE force calculation
%   Coulomb+harmonic

N=length(r(:,1)); % assuming first index to be particle index
force=-2*r; % force from the harmonic potential
ff=zeros(N,N,2);
for i=1:N,
    for j=i+1:N,
        % force between particles i and j
        ff(i,j,:)=(r(i,:)-r(j,:))./norm(r(i,:)-r(j,:))^3;
        ff(j,i,:)=-ff(i,j,:);
    end
end
for i=1:N,
    % Summing force from all other particles:
    force(i,1)=force(i,1)+sum(ff(i,:,1)); % x component
    force(i,2)=force(i,2)+sum(ff(i,:,2)); % y
end
end
```

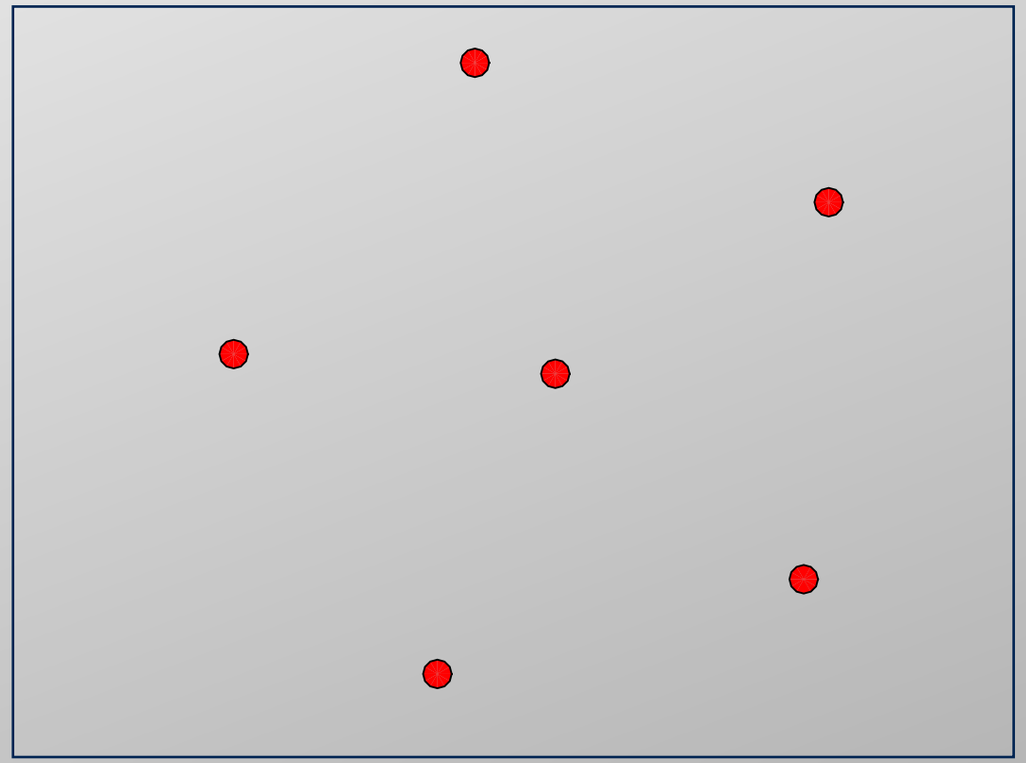
Minimum potential by direct optimisation

r = fminunc(@md_potential,r);

fminunc finds a local minimum of a function of several variables.
X = fminunc(FUN,X0) starts at X0 and attempts to find a local minimizer X of the function FUN. FUN accepts input X and returns a scalar function value F evaluated at X. X0 can be a scalar, vector or matrix.

`md_min_pot.m`



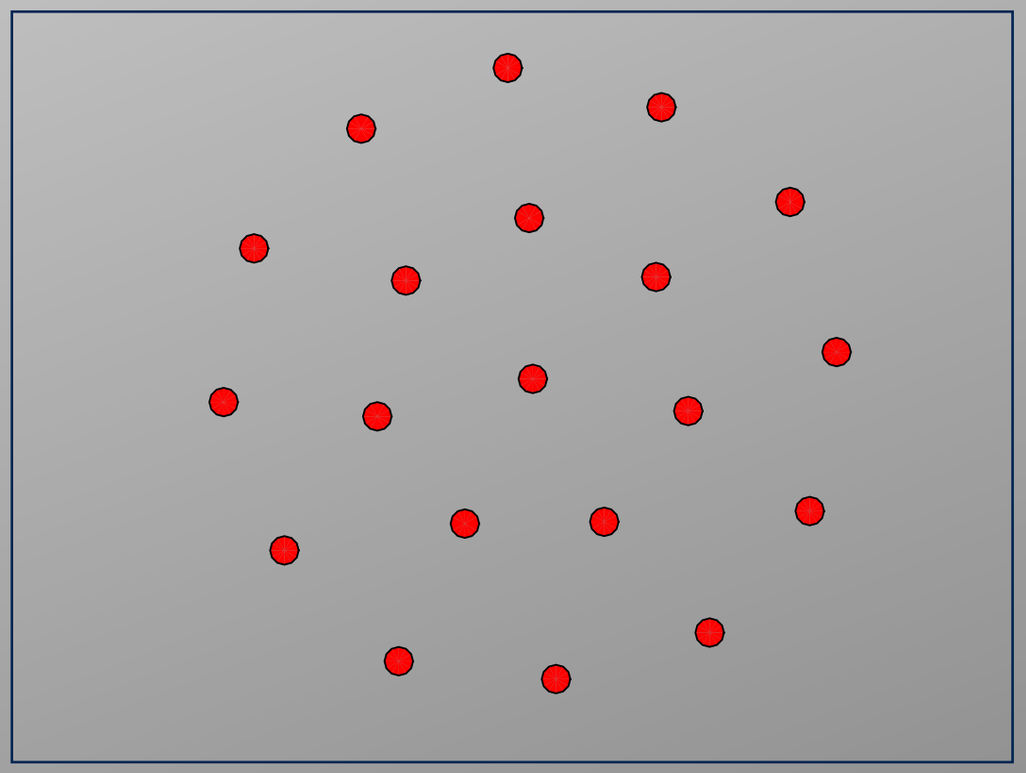


Small systems are polygons

We can use the gradient in optimisation

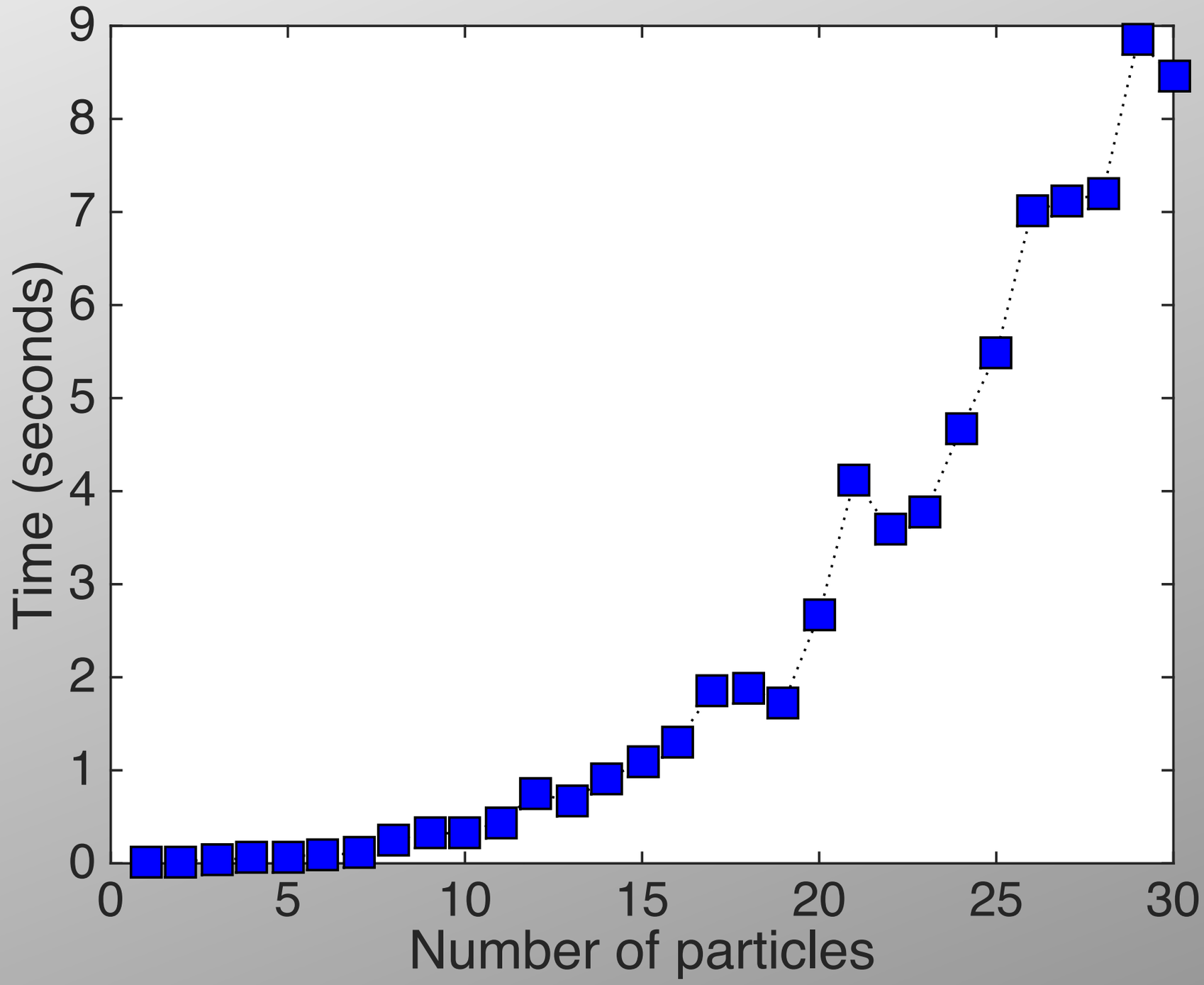
Larger systems start showing rings

In the middle of the cluster,
a triangular lattice can be seen
To find the minimum potential structures is rather easy for small particle numbers



Time of direct optimisation

```
r=randn(N,2); % random positions, 2D system
options = optimoptions(@fminunc,'GradObj','on','Algorithm','trust-region');
tic;
r = fminunc(@md_potential,r,options); % minimise the potential
t(N)=toc;
```



md_min_pot.m

Molecular dynamics

Propagate coordinates and velocities as in the one-dimensional example:

$$
\begin{aligned}
x(t+h) &= x(t) + h\dot{x}(t) + h^2 F[x(t)]/2 \\
\dot{x}(t+h) &= \dot{x}(t) + h(F[x(t+h)] + F[x(t)])/2
\end{aligned}
$$

Coordinates in “r” and velocities in “v”:

```
r=r+dt*v+.5*dt^2*f;
v=v+.5*dt*f;
f=md_force(r);
v=v+.5*dt*f;
```

Time step is “dt”, h in one dimensional formulas

First line assumes the force to be in “f”

Doing this should conserve the energy

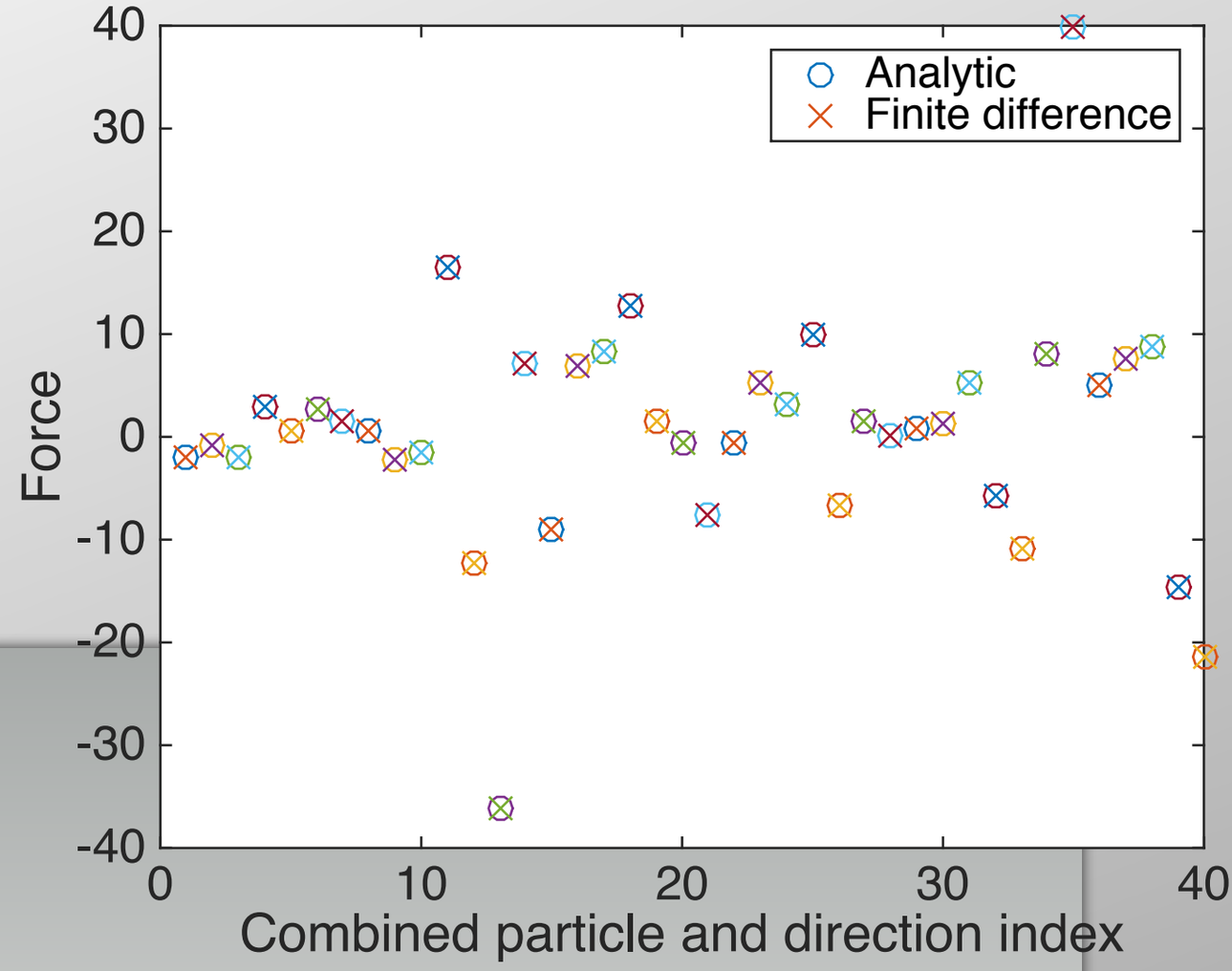
Average kinetic energy is proportional to temperature

Testing the force:

Comparison with a finite difference from potential

md_force_test.m

```
N=20; % Number of particles
r=randn(N,2); % positions, 2D system

potential=md_potential(r);
force=md_force(r);
h=1e-5;
ii=0;
for i=1:N,
   for d=1:2,
       % force by finite difference compared with the md_force function
       r(i,d)=r(i,d)+h;
       pot_plus=md_potential(r);
       r(i,d)=r(i,d)-h;
       ii=ii+1;
       plot(ii,force(i,d),'o',ii,-(pot_plus-potential)/h,'x');hold on;
   end
end
```

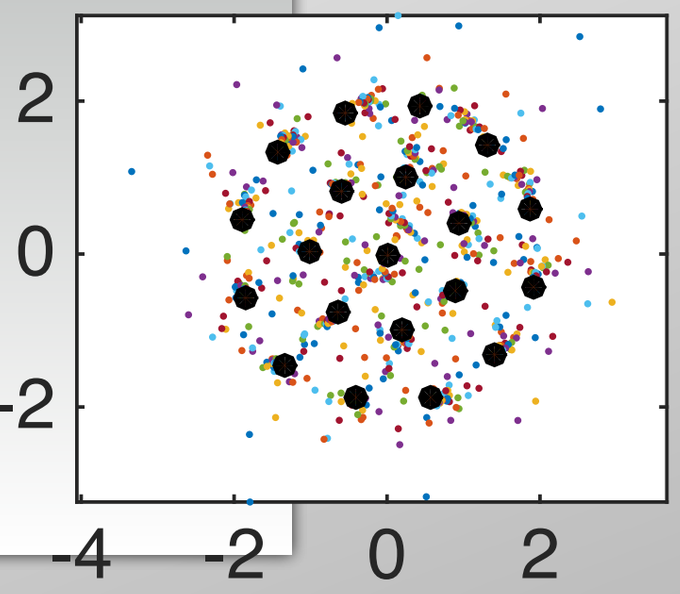


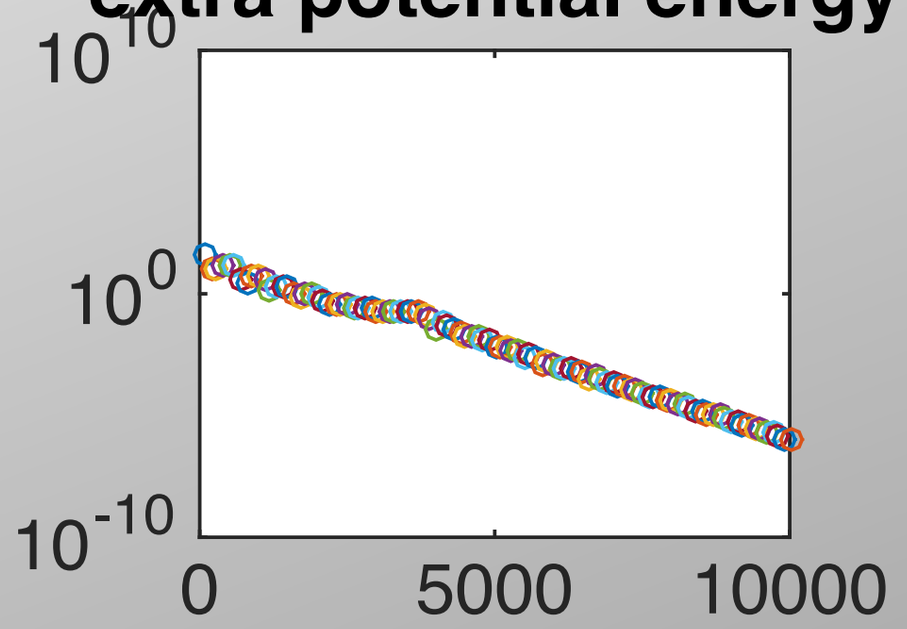
Scaling down velocities to zero temperature

```
T=10000;% number of time steps
dt=1e-2;% time step size
f=md_force(r);
for t=1:T,
    r=r+dt*v+.5*dt^2*f;
    v=v+.5*dt*f;
    f=md_force(r);
    v=v+.5*dt*f;
    if mod(t,100)==0
        v=.8*v;
    end
end
```

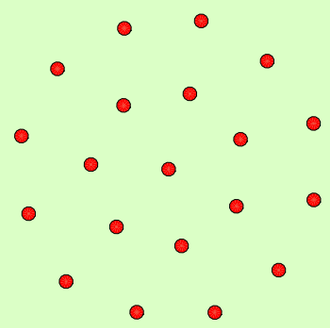
md_cool.m

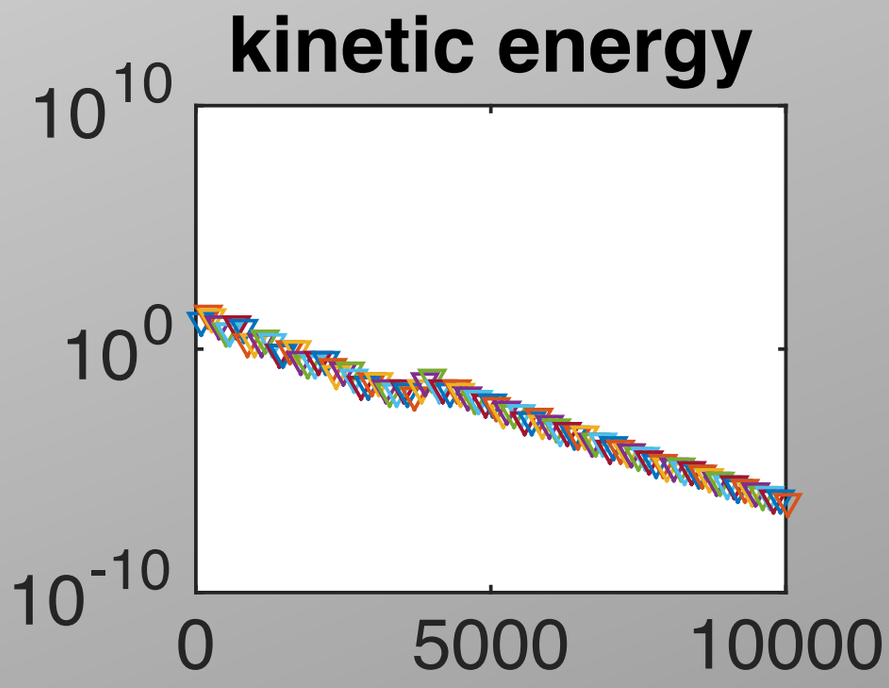
Small dots are snapshots of positions, large spots the final steady state

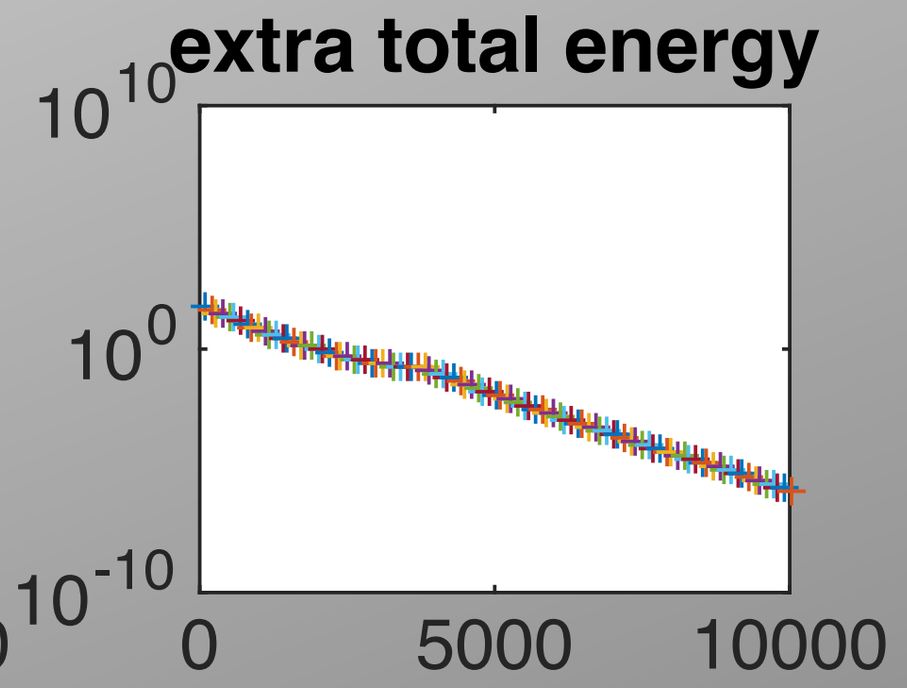




Compare with the optimised positions:







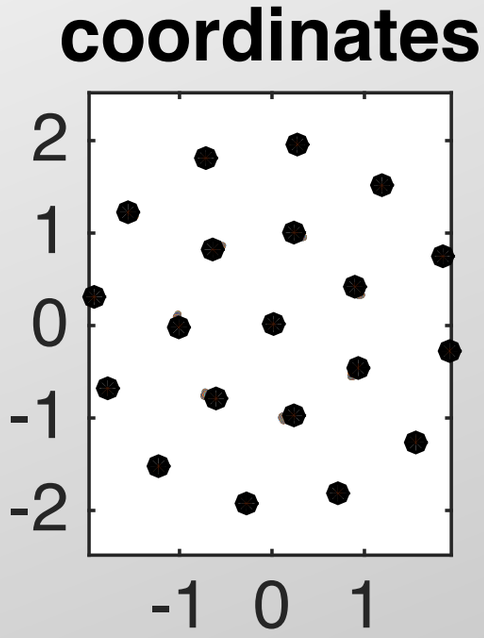
Time step index on x axis (except in coordinates)

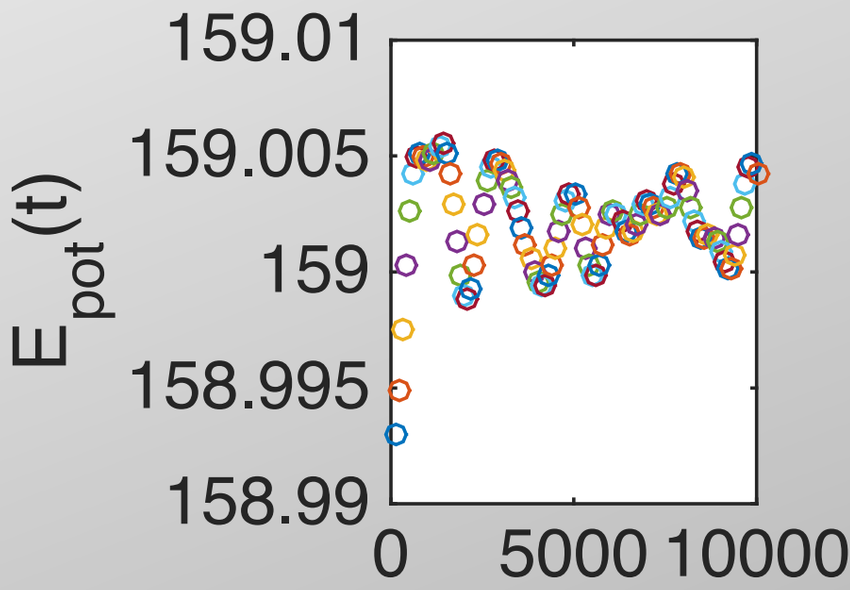
Constant energy, initial velocities tiny

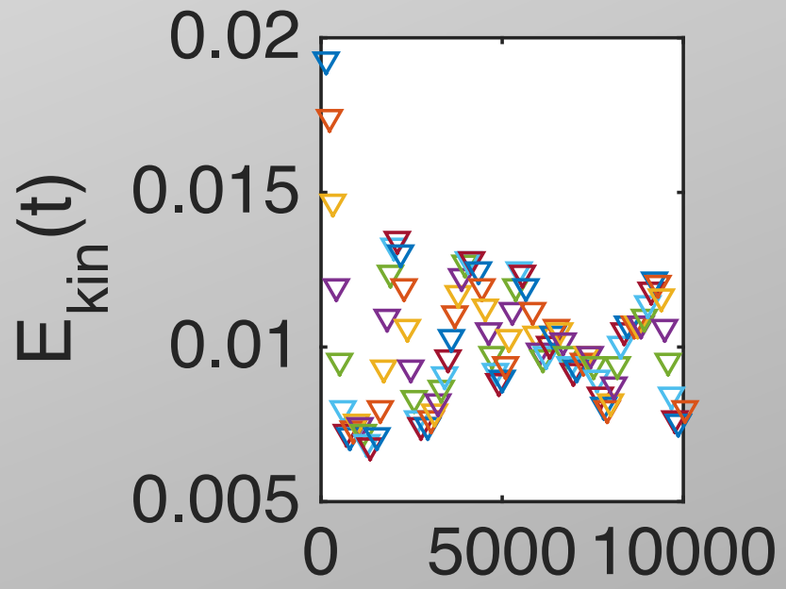
md_constEnergy.m

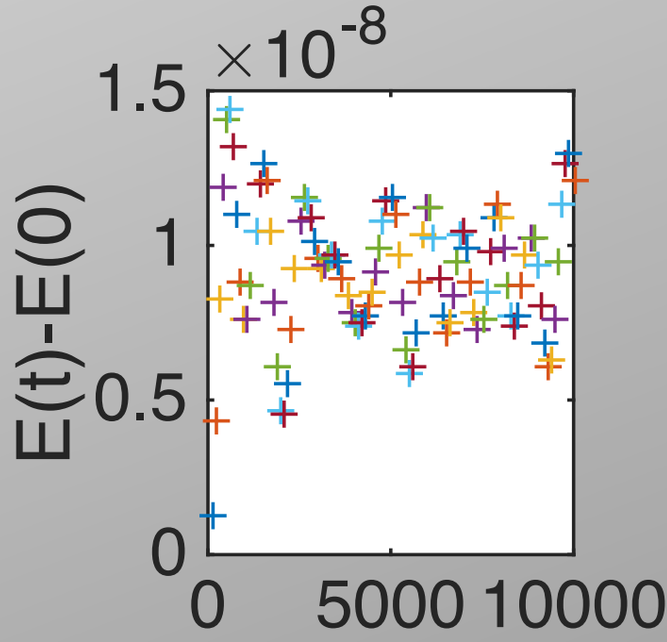
Tiny fluctuation in coordinates

Solid?









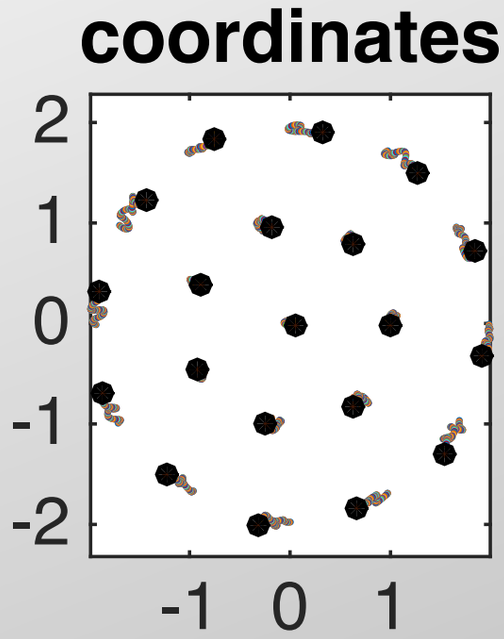
Total energy nearly constant

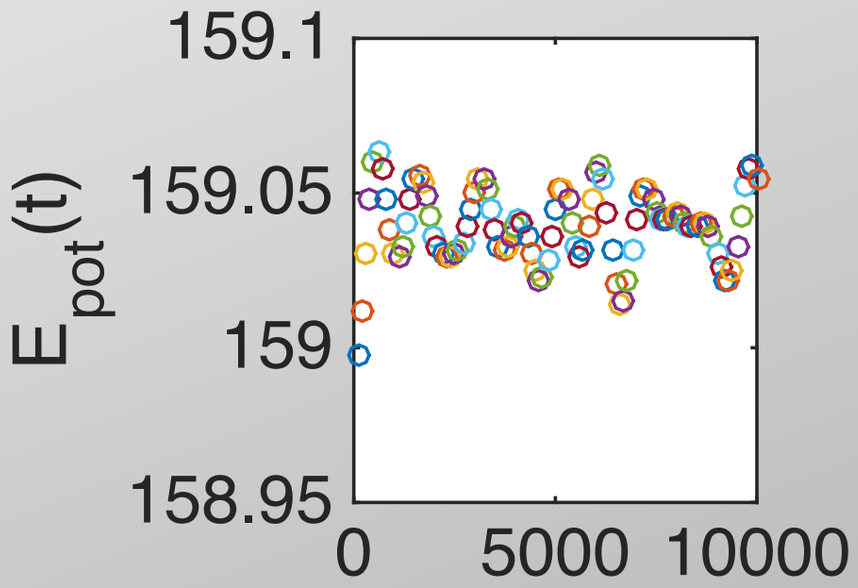
Kinetic and potential energies show fluctuation!

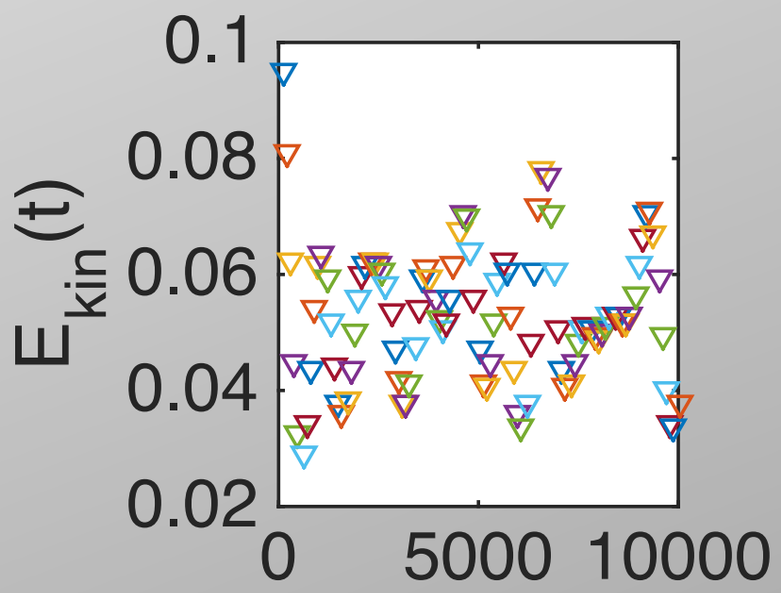
Constant energy, initial velocities small

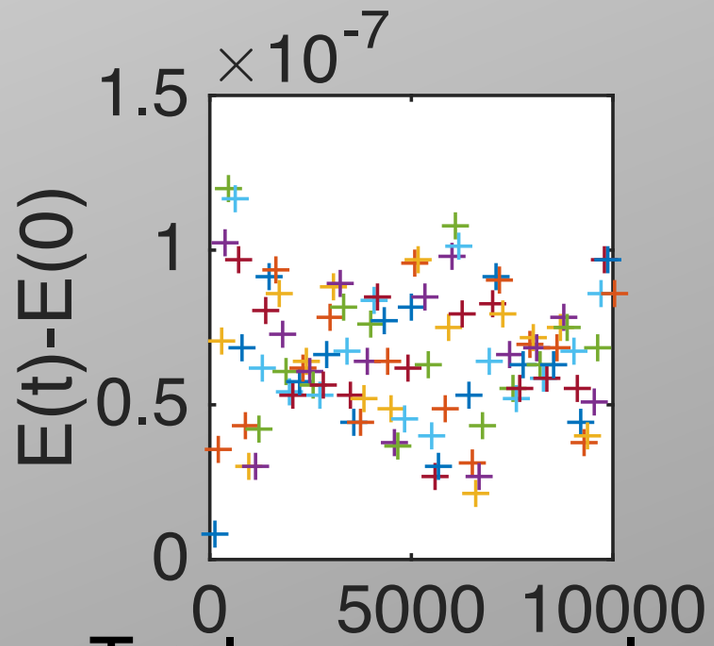
md_constEnergy.m

Small fluctuation in coordinates









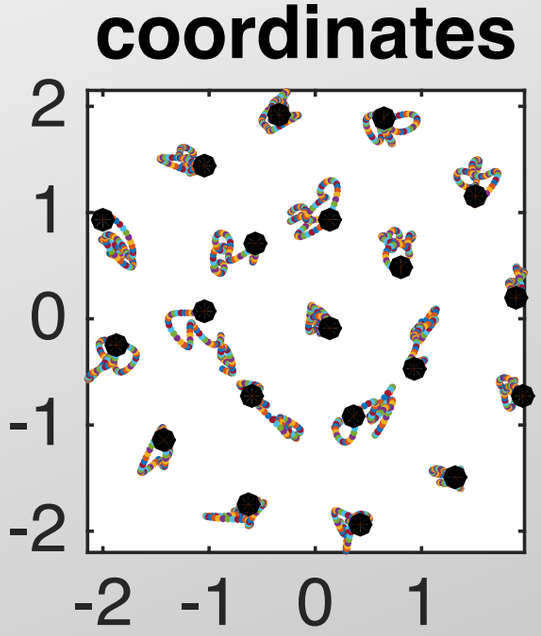
Total energy nearly constant

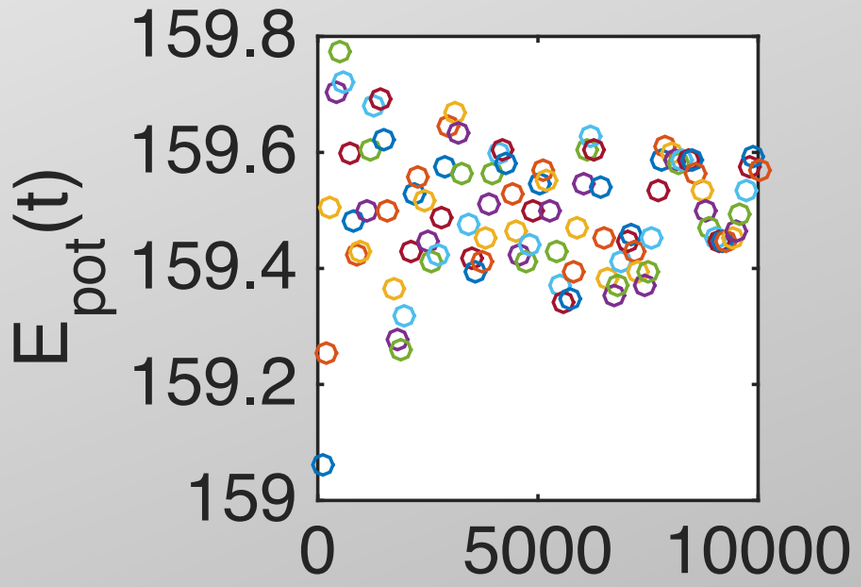
Average kinetic energy around five times larger, five times higher temperature

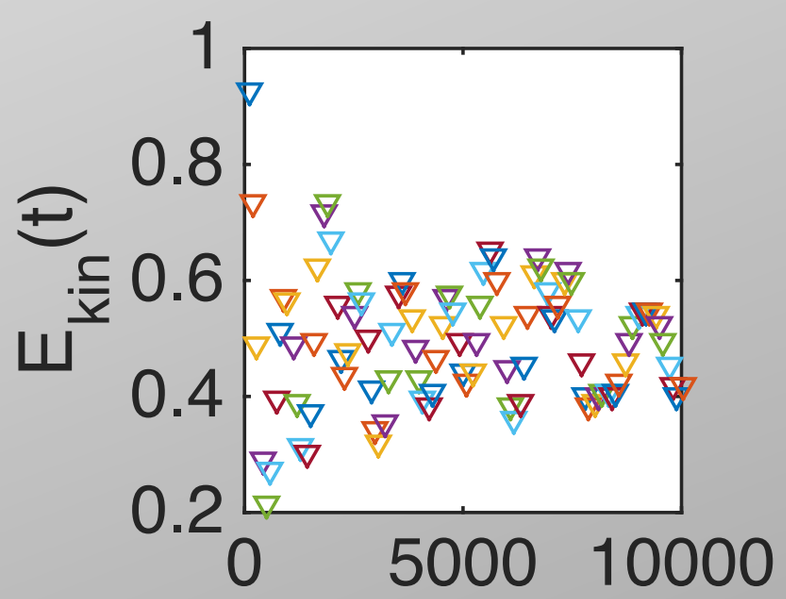
Constant energy, initial velocities modest

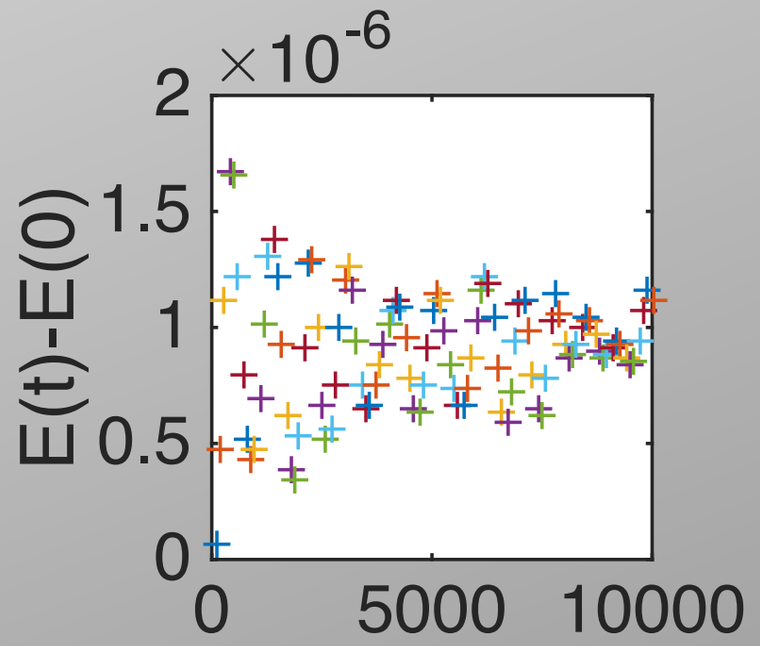
md_constEnergy.m

More fluctuation in coordinates









Total energy nearly constant

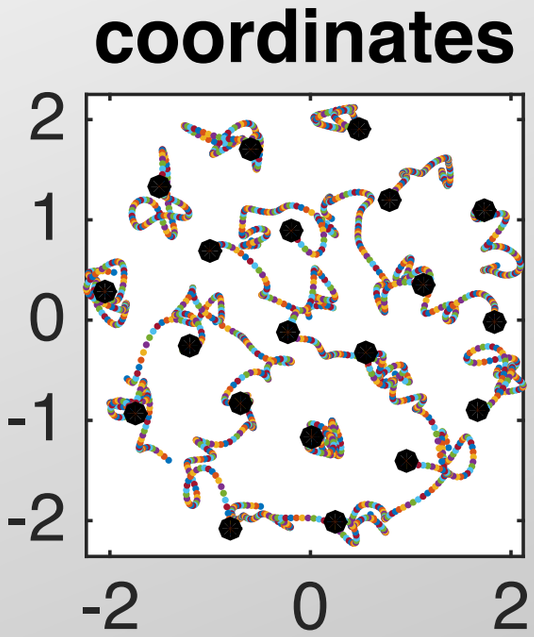
Around ten times higher temperature

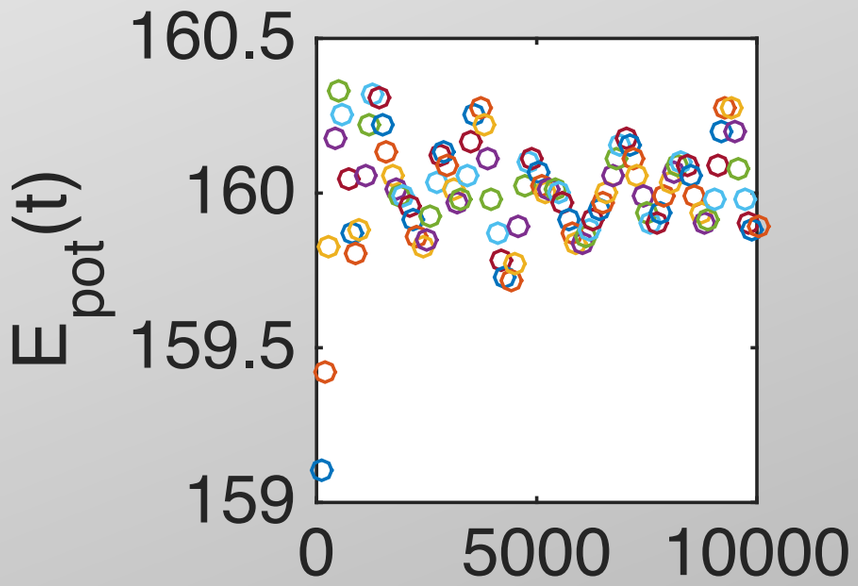
Constant energy, initial velocities larger

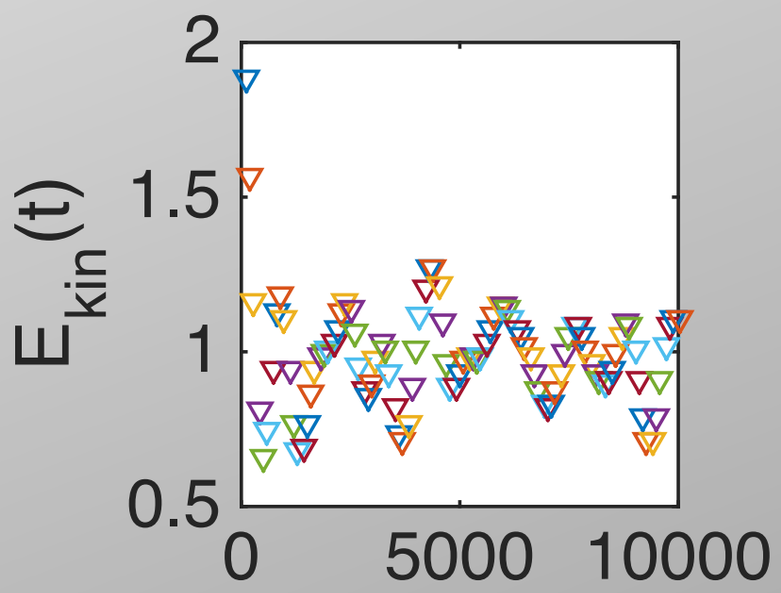
md_constEnergy.m

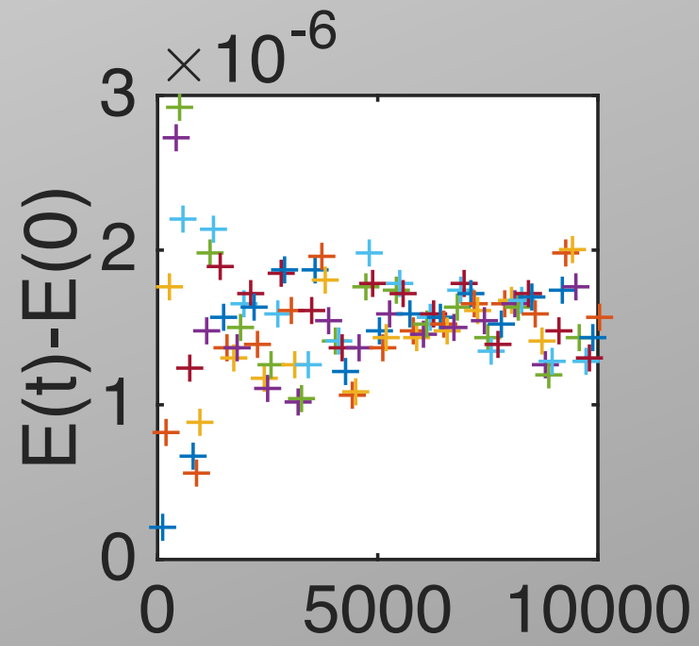
Even more fluctuation in coordinates

Liquid?









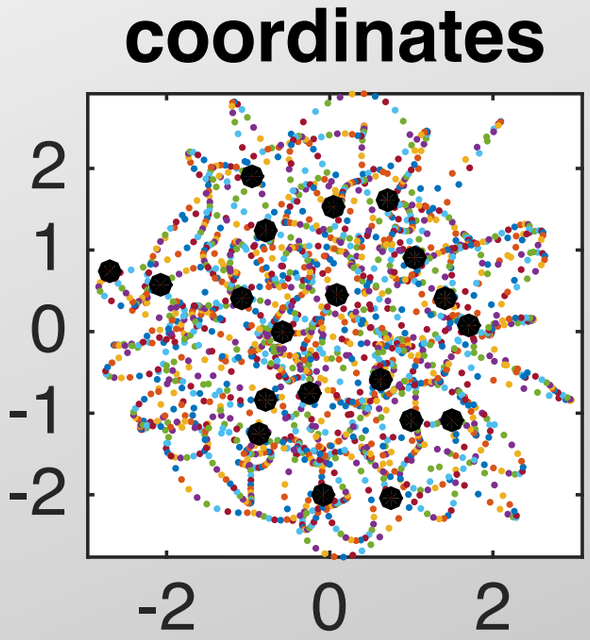
Around double temperature from the previous one

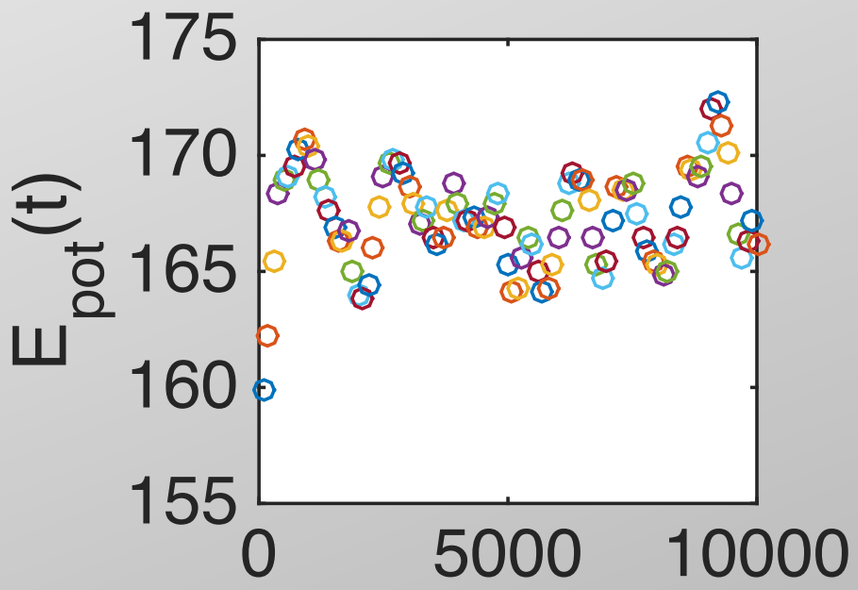
Constant energy, initial velocities huge

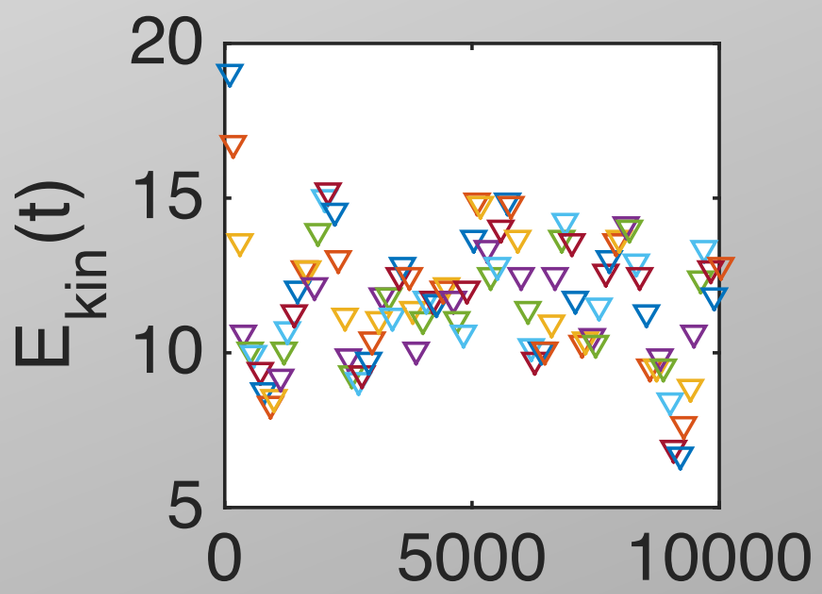
md_constEnergy.m

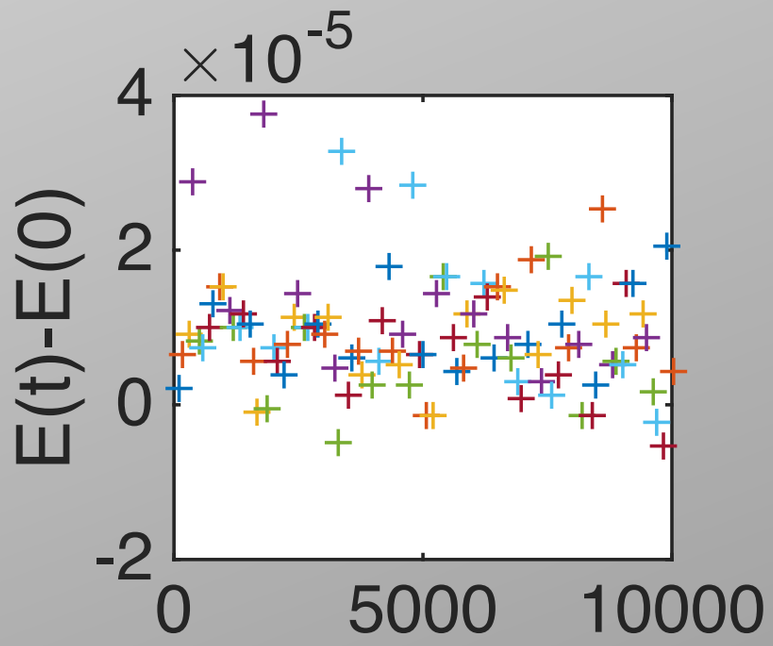
"Ballistic" coordinates

Gas?





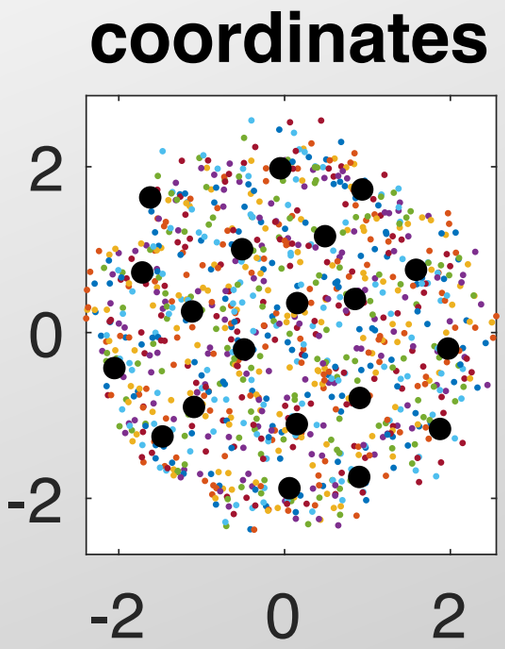


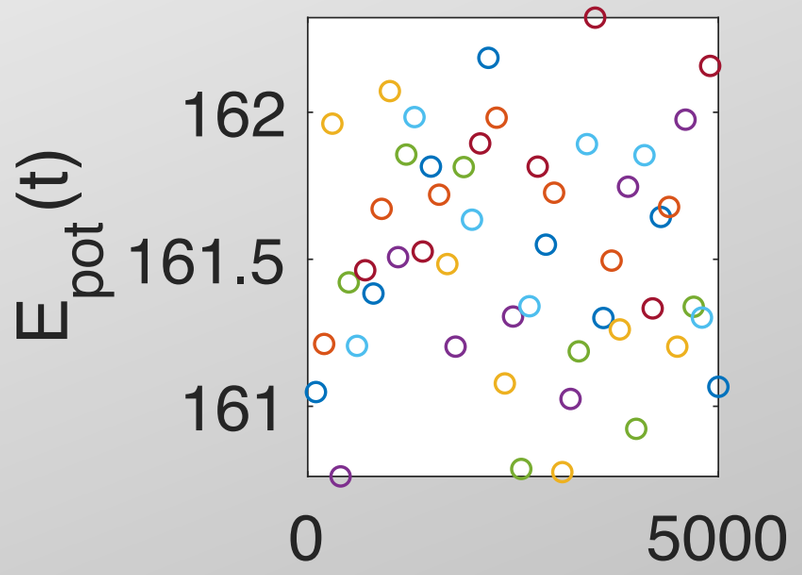


Around ten times higher temperature

Constant temperature

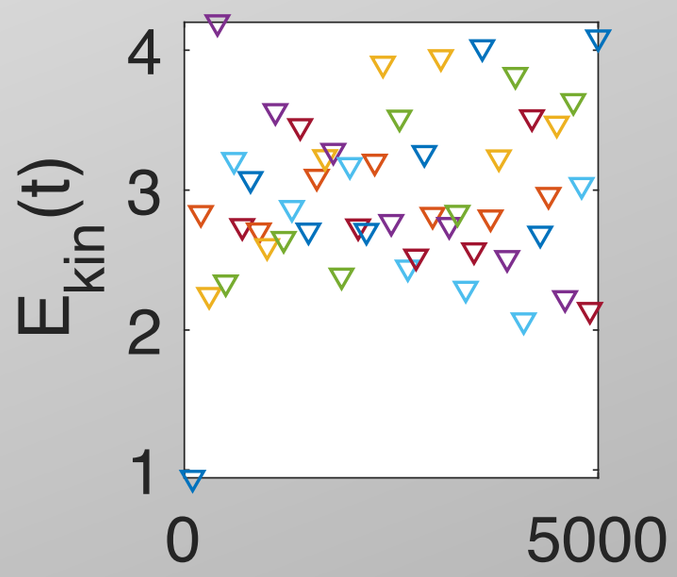
md_constT.m

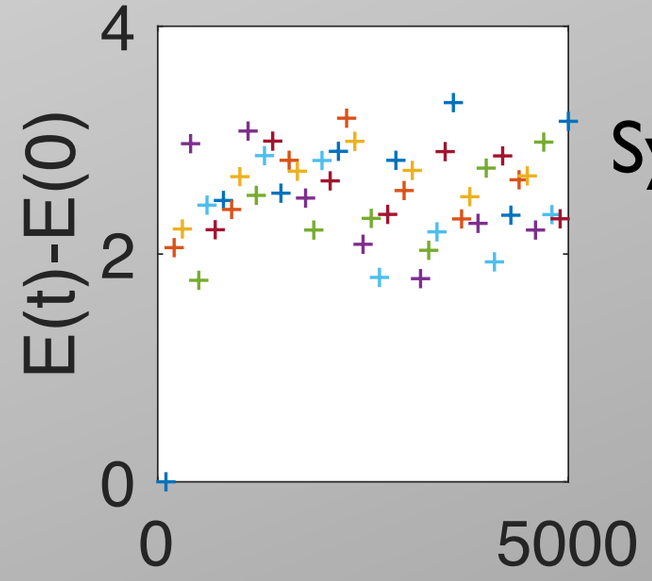




$$\langle K \rangle = \frac{D}{2} N k_B T$$

v=v.*sqrt(temperature/(kin/N));



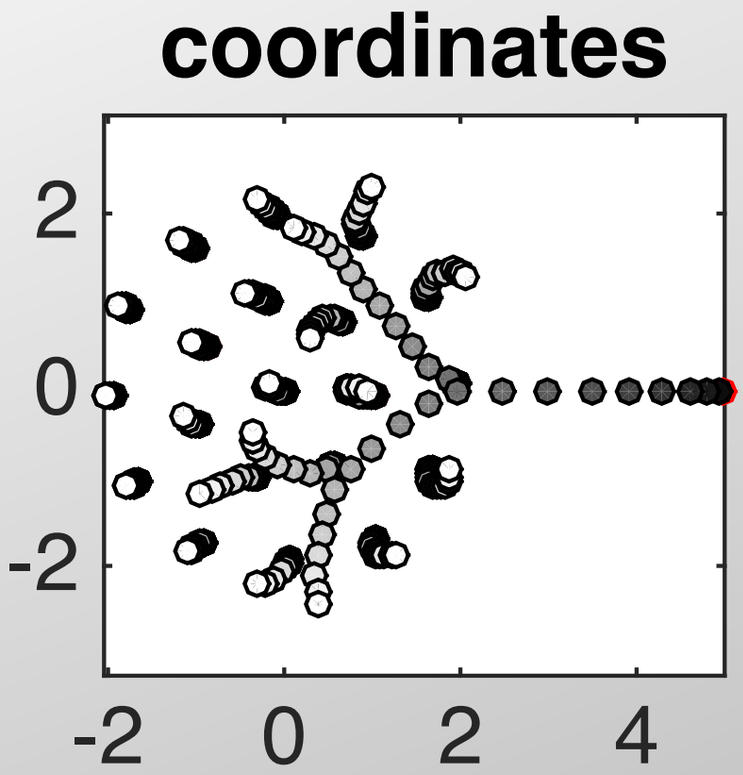


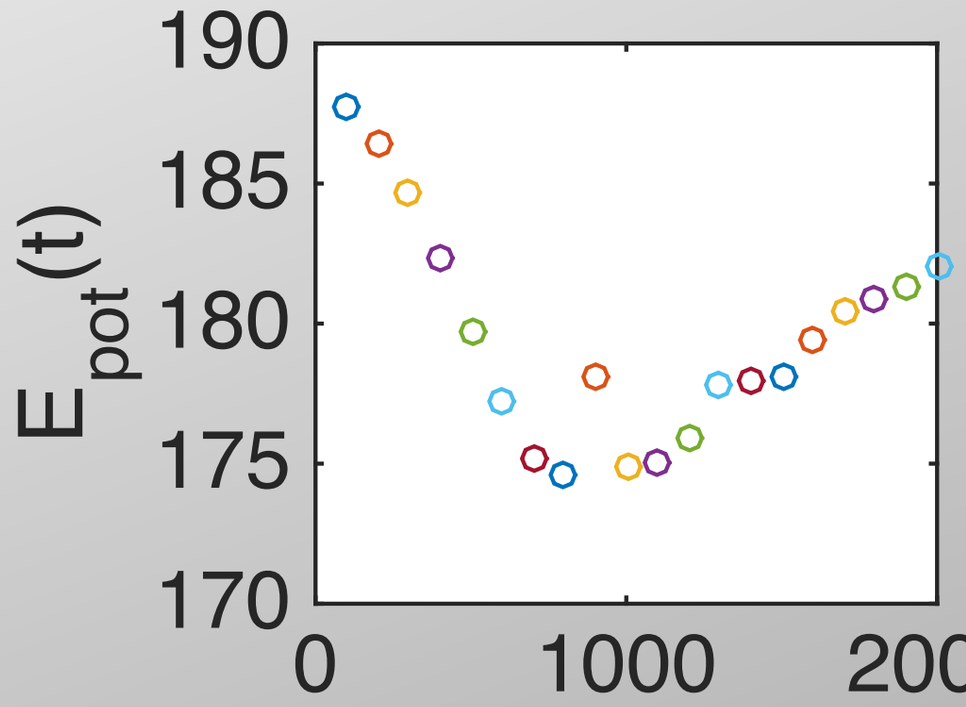
System exchanges energy with the ”reservoir”

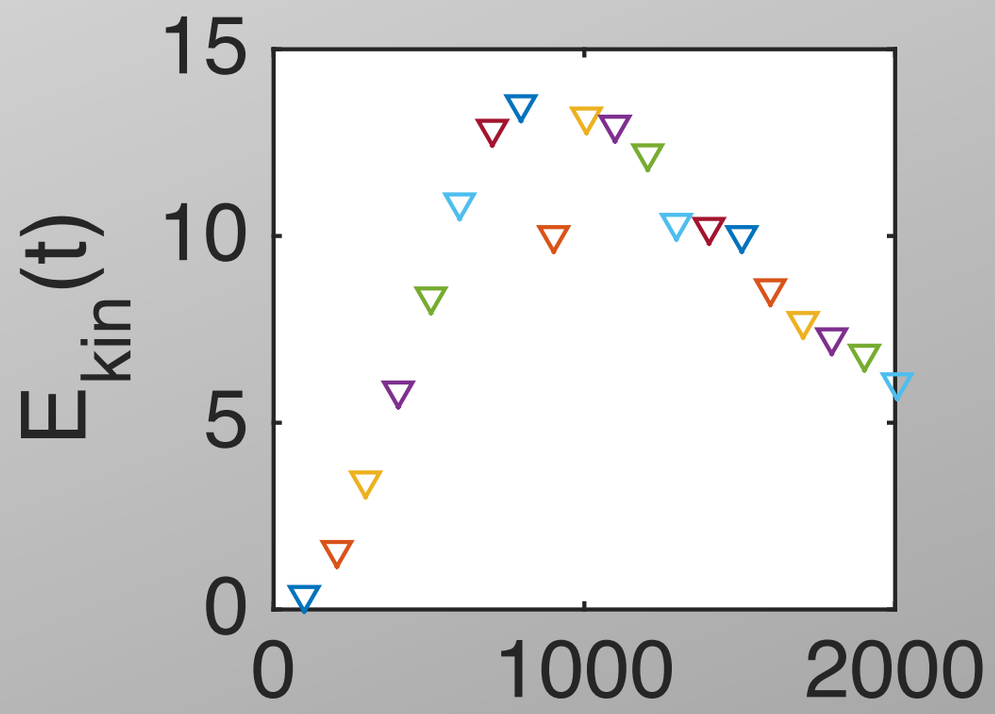
Code scales velocities periodically (every 100 steps) in order to satisfy the equipartition theorem (a crude ”thermostat” to enable NVT simulations)

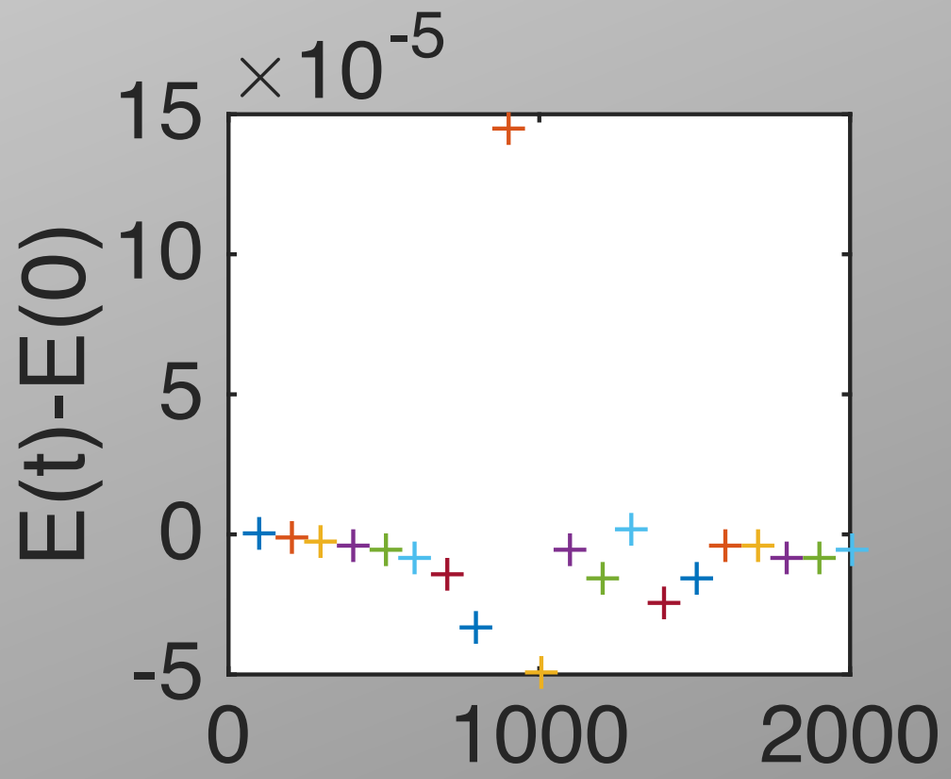
Constant energy, billiards

md_billiards.m









Time reversal / chaotic systems (project topic?)

If you reverse the direction of time at the middle of the simulation, you should obtain the starting state at the end of the simulation.

This can be done by changing the sign of the time steps of by reversing all the velocities “v=-v”.

It turns out that many systems are chaotic and small numerical inaccuracy is enough to make trajectories deviate at later time, see:
http://en.wikipedia.org/wiki/Lyapunov_exponent and
http://en.wikipedia.org/wiki/Chaos_theory

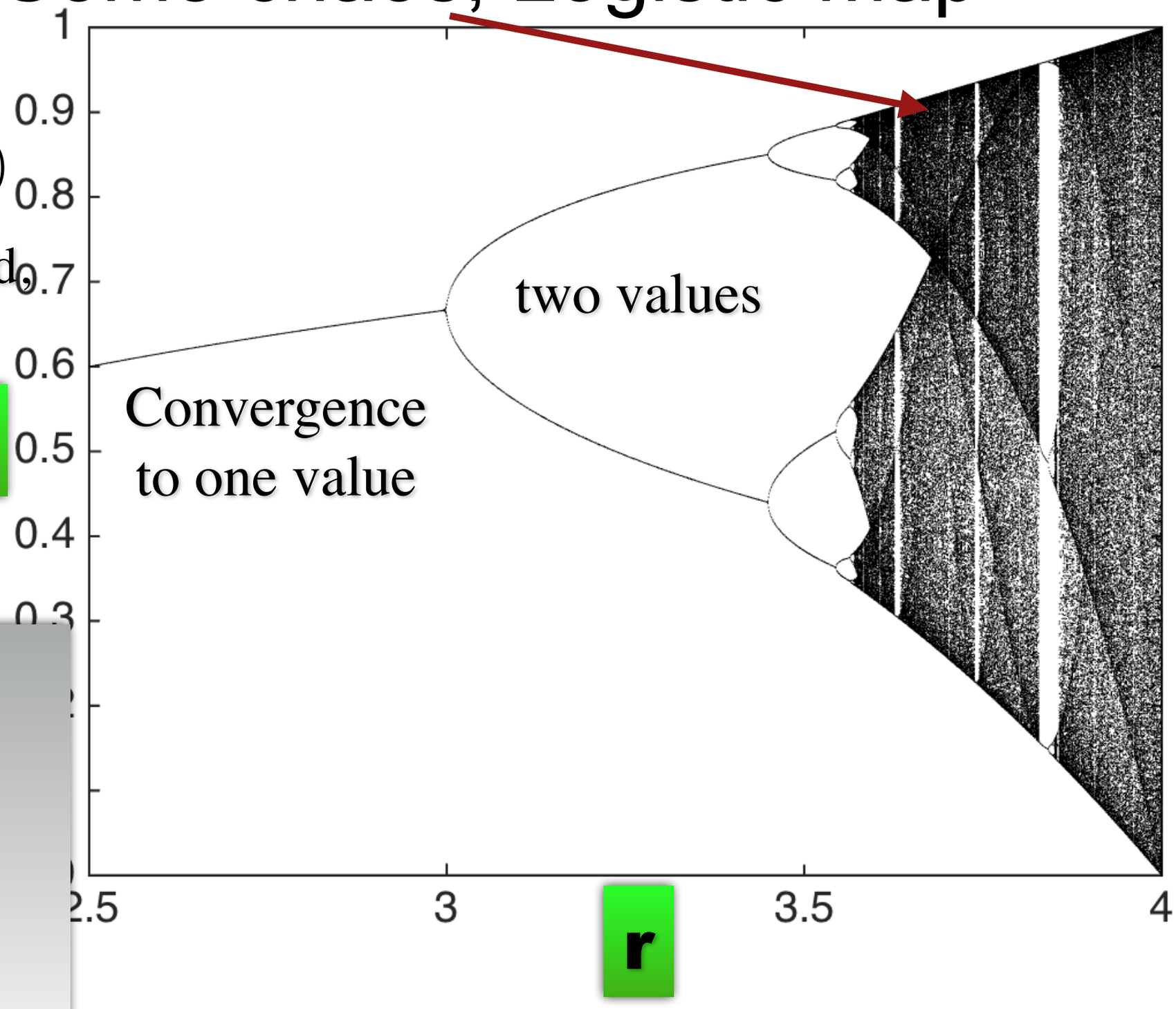
In some cases, however, the time reversal is possible, and you can continue the simulation to the early seconds of the universe, assuming the laws of physics are given by the formulas your code has :-)

Some chaos, Logistic map

Repeat

$$x_{n+1} = r\ x_n\ (1 - x_n)$$

Plot x after initial period, different r values:

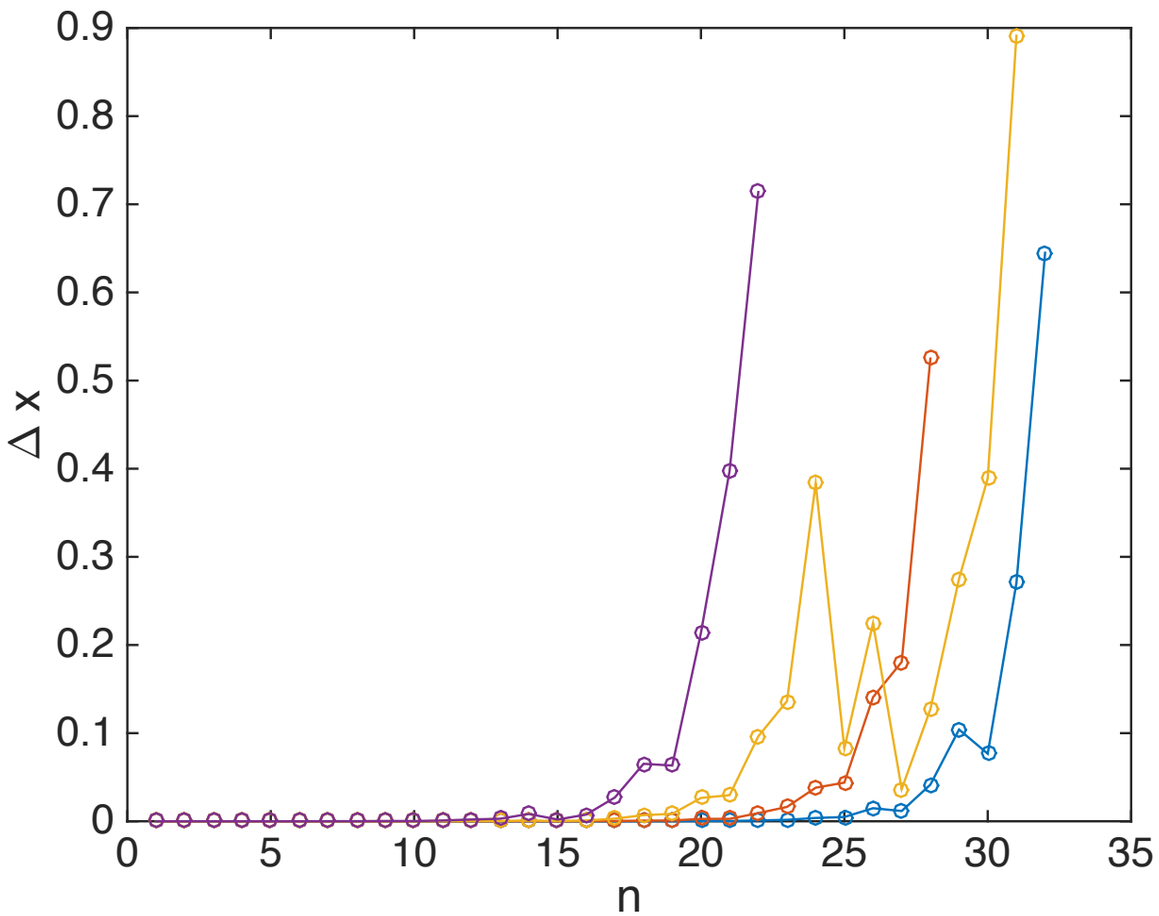


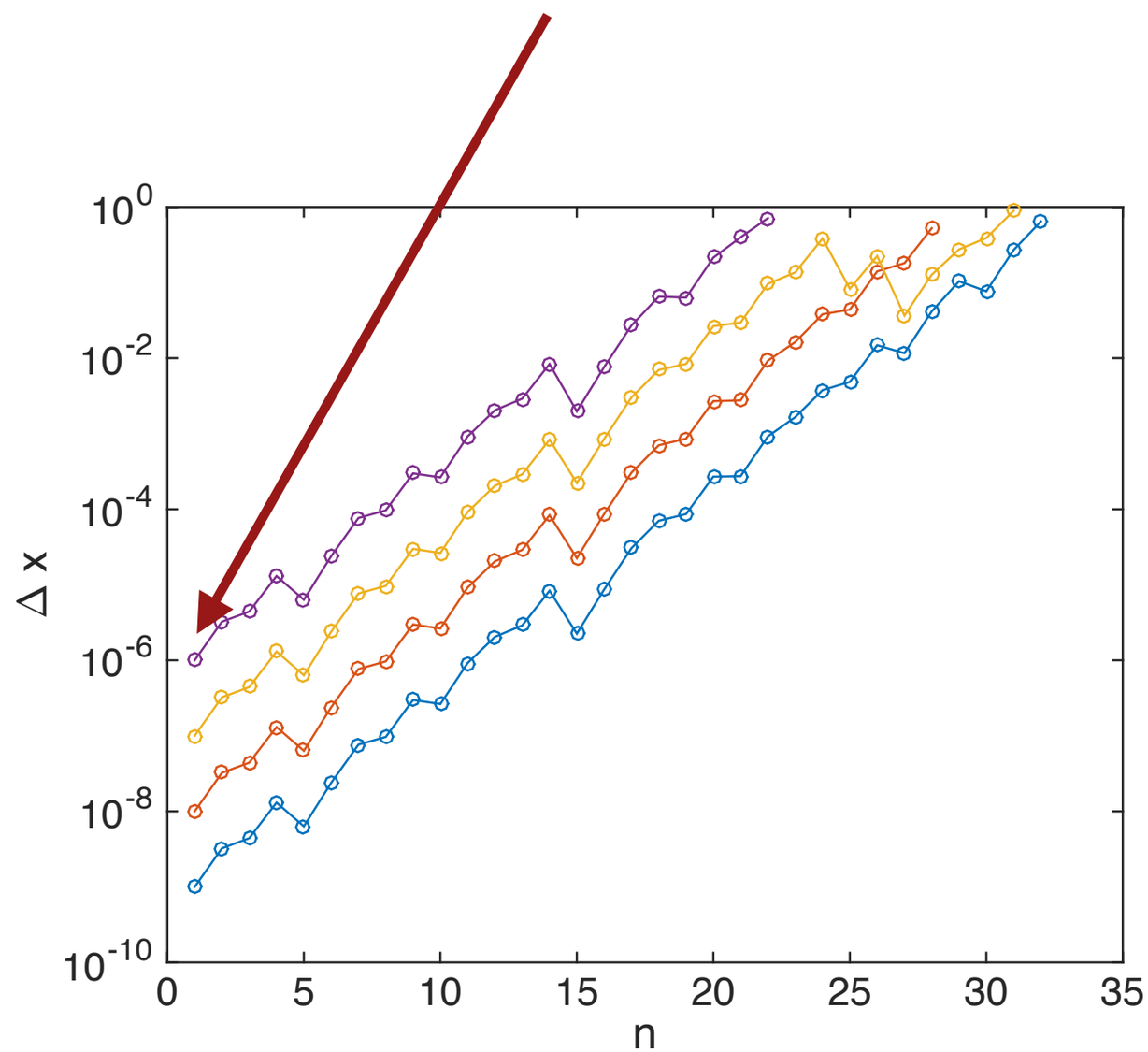

```
Nout=1000;% Omitted points
Nplot=500;% Plotted points
x=zeros(Nplot,1);
for r = 2.5:0.001:4.0
  x(1)=rand;
  for n = 1:Nout
    x(1)=r*x(1)*(1-x(1));
  end
  for n = 1:Nplot-1
    x(n+1) = r*x(n)*(1 - x(n));
  end
  plot(r+0*(1:Nplot), x, 'k.', 'markersize', 1)
  hold on;
end
```

Initial value changed

r=4-.1*rand considered here, chaotic area.

Small change in the initial value of x(1), see value from right figure, at n=1

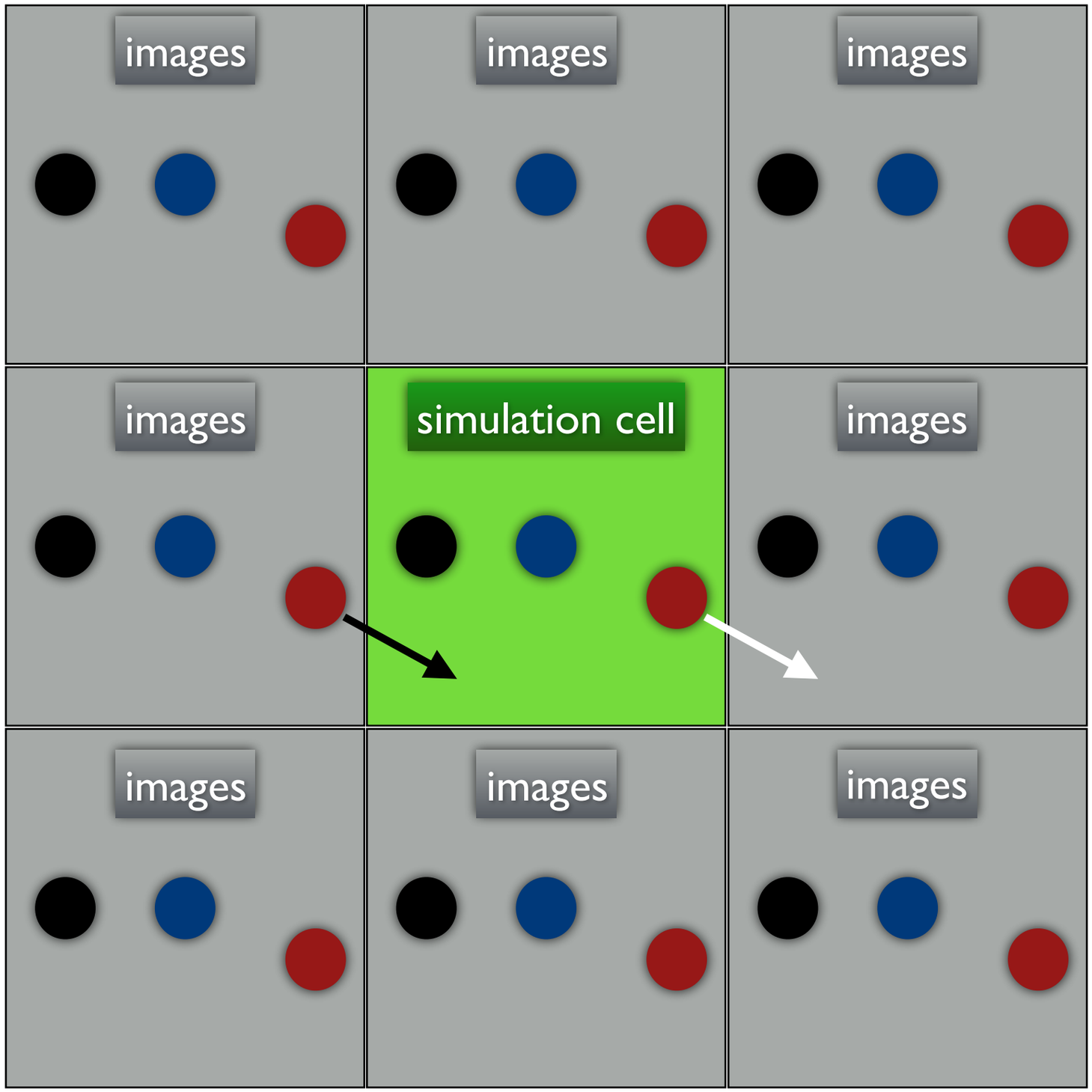




Clearly exponential deviation later. It means that predicting chaotic trajectory is “demanding”.

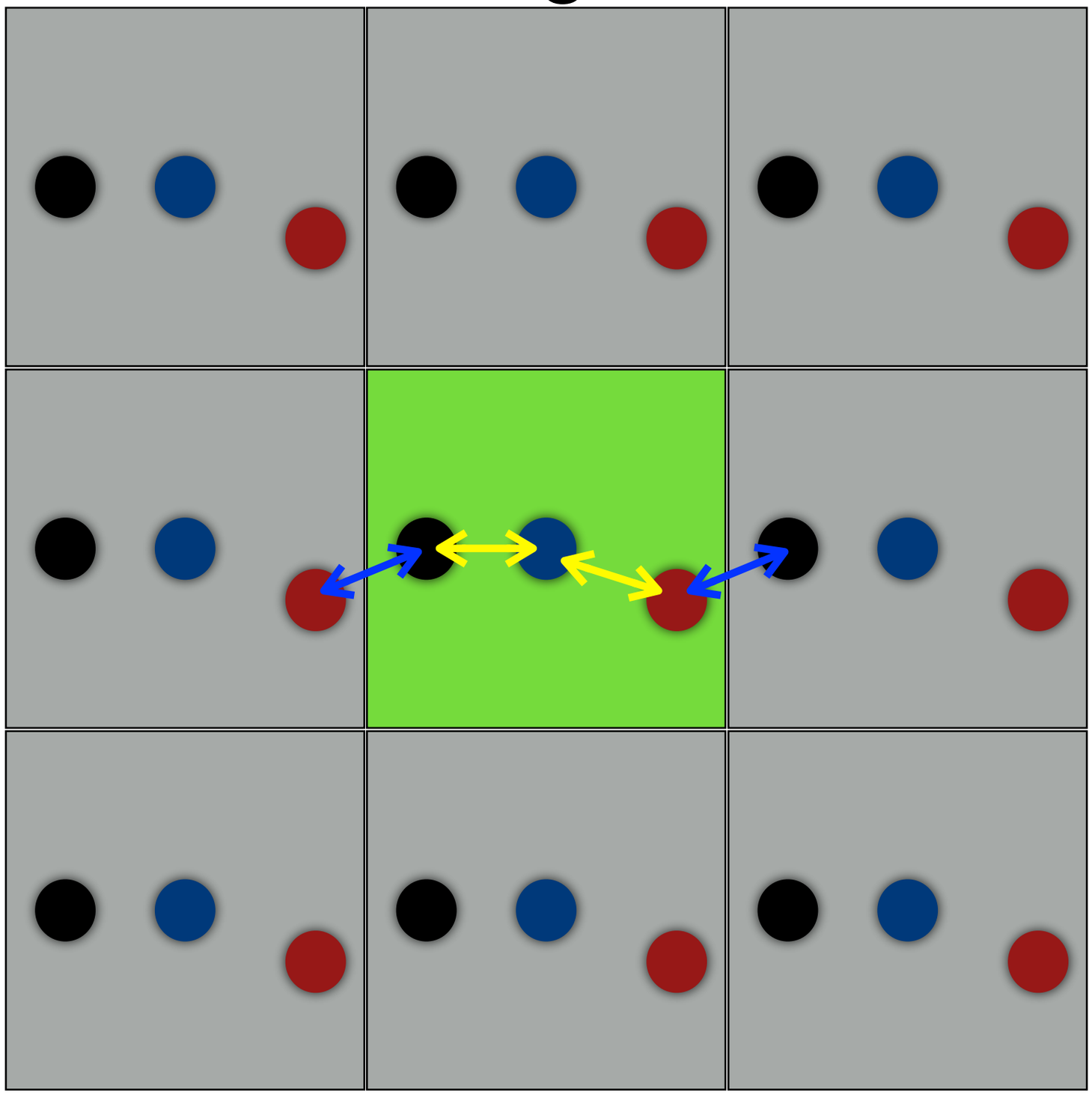
Finite size effects

It might be better to use periodic boundary conditions to reduce finite size effects.



Moving out (**white arrow**) from the simulation cell corresponds to moving in as **black arrow**

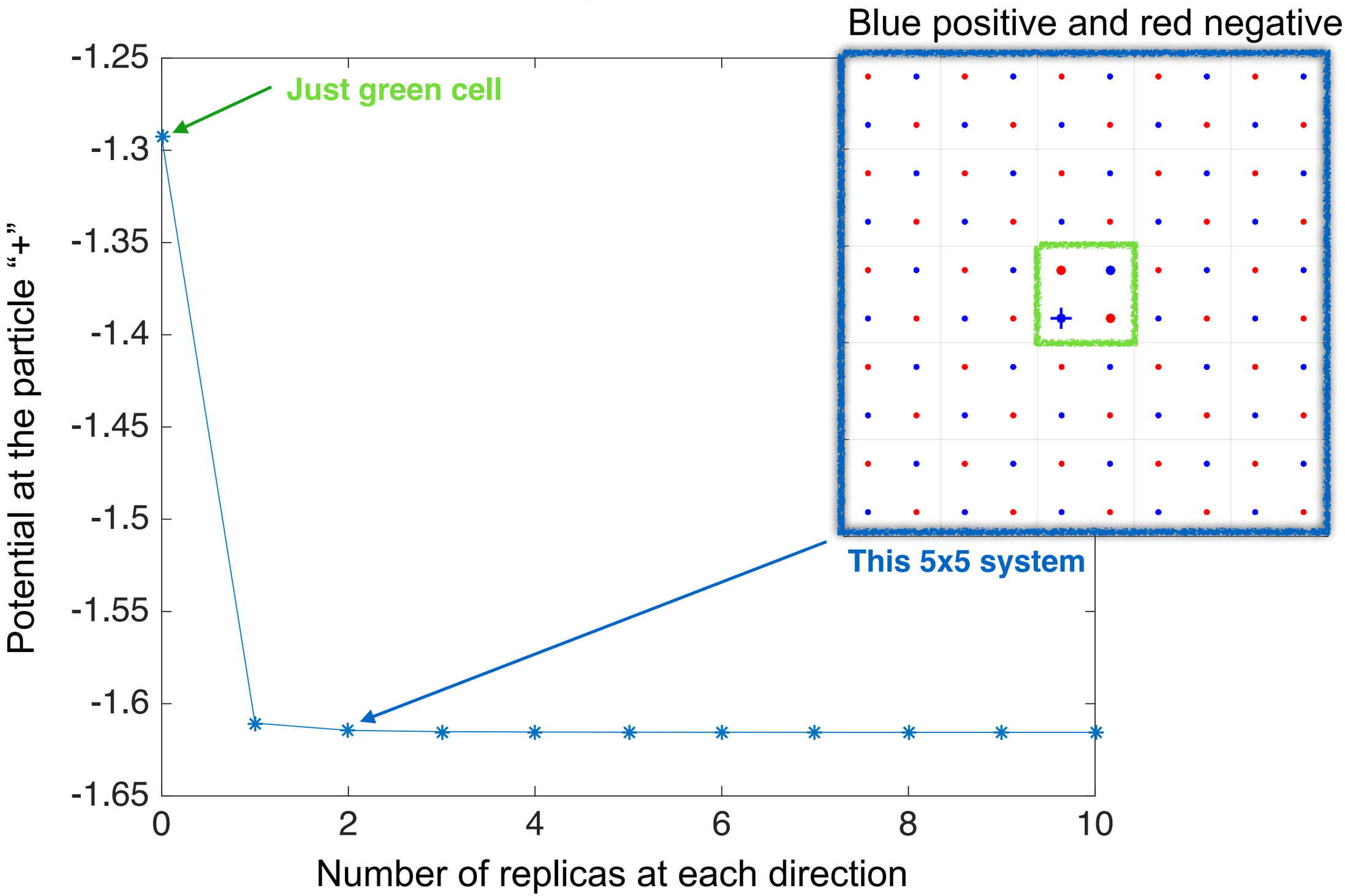
Minimum image convention



Interaction with the closest image (**blue**) or the original particle (**yellow**).

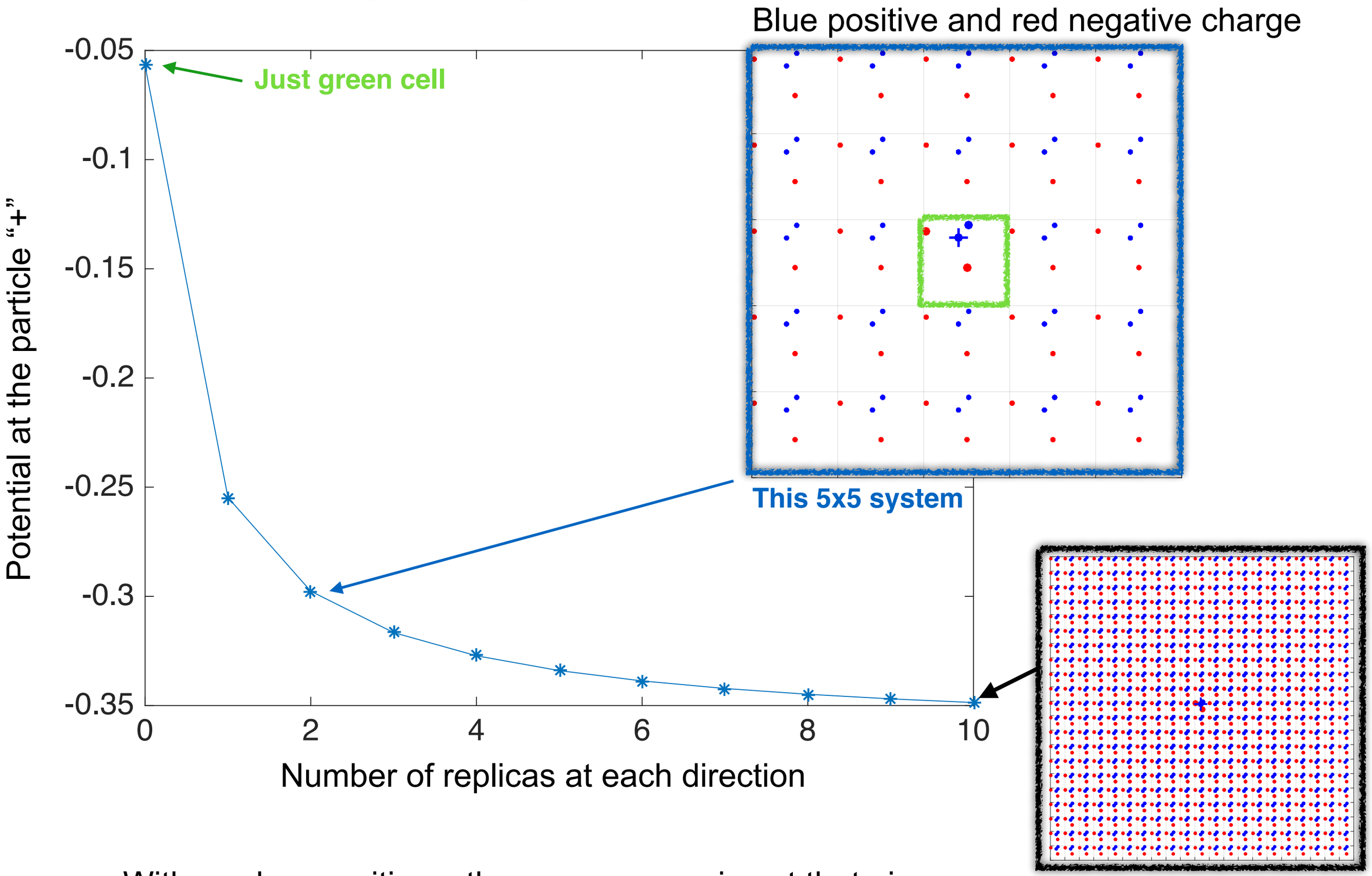
Note that long-range interaction would reach further images, causing some trouble…

Long-range Coulomb potential



Seems to converge nicely. (This is actually a Madelung constant calculation)

Long-range Coulomb potential



With random positions, the convergence is not that nice…

the Ewald summation gives a nicer convergence!