Part 2: Molecular Dynamics

- Literature
- History
- Practical Issues

Literature

- A. K. Hartmann
  Big Practical Guide to Computer Simulations
  (World Scientific, 2015)

- M. P. Allen and D.J. Tildesley
  Computer Simulations of Liquids
  (Oxford University Press, 1990)

- D. Frenkel and B. Smit
  Understanding Molecular Simulations
  (Academic Press, San Diego, 2002)

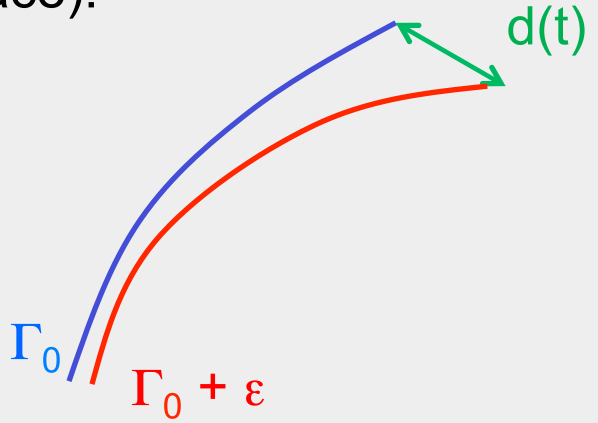
History

- 1940's: Use of computers to solve problems in nuclear physics (bombs) & cryptographie
- 1950's: First use of computers in statistical physics
- 1953: First simulation of a liquid (equation of state) Metropolis, Rosenbluth, Rosenbluth, Teller, Teller; introduction of the "Monte Carlo" method and the "Metropolis algorithm"
- 1955: First simulation of a non-linear crystal (1d) using molecular dynamics; Fermi-Pasta-Ulam
- 1956: First simulation of the dynamics of hard spheres; Alder & Wainwright
- 1960's: Development of algorithms; integrator by Verlet; neighbor lists;...
- 1980's: ab initio calculations; Car-Parrinello; gaussian;...
- 1990's: parallel computers; grand-canonical simulations; histogram-reweighting
- 2000's: Use of GPU's

Molecular Dynamics

In MD one integrates numerically Newton's equations of motion of the system ⇒ trajectory of the system in phase space.

BIG problem: Most equations of motion are non-linear and thus the trajectory is "chaotic": Consider two trajectories that start at points Γ_0 and $\Gamma_0 + \varepsilon$ (in phase space):



Chaotic systems : distance d(t) ~ ε exp(λt) with a $\lambda > 0$, the "Lyapunov exponent". Thus every deviation is amplified exponentially in time **⇒ It is not possible to calculate the exact trajectory over "long" times!!!**

"Solution": Shadowing Lemma: For each finite t there is an exact solution that tracks the approximate one within a given error bar

MD: Algorithms

Properties of a good MD algorithm (=integrator):

1. Good conservation of the total energy of the system, even for very long times

2. Invariance under an inversion of time (thus no algorithms of type predictor-corrector)

3. Conversation of phase space volume, i.e. the Liouville theorem must hold: The algorithm is “symplectic”

$\Rightarrow$ The velocity form of the Verlet algorithm is good

Verlet algorithm

Consider one particle in d = 1 with potential V (r):

Equation of motion:

$$m\ddot{r}(t) = f(t) = -\frac{\partial V(r)}{\partial r}$$

Make Taylor expansion in the position and let *h* be a small time increment:

$$\begin{aligned} r(t+h) &= r(t) + h\dot{r}(t) + \frac{h^2}{2}\ddot{r}(t) + \frac{h^3}{3!}\dddot{r}(t) + O(h^4) \\ &= r(t) + hv(t) + \frac{h^2}{2}\frac{f(t)}{m} + \frac{h^3}{3!}\dddot{r}(t) + O(h^4) \end{aligned}$$

$$r(t-h) = r(t) - hv(t) + \frac{h^2}{2}\frac{f(t)}{m} - \frac{h^3}{3!}\dddot{r}(t) + O(h^4)$$

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

$$v(t) = \frac{r(t+h) - r(t-h)}{2h} + O(h^2)$$

Verlet algorithm

Velocity Verlet algorithm

The Verlet algorithm needs two initial positions (not very practical)

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

$\Rightarrow$ velocity Verlet algorithm

$$\begin{aligned} r(t+h) &= r(t) + hv(t) + \frac{h^2}{2}\frac{f(t)}{m} \\ v(t+h) &= v(t) + \frac{h}{2m}\left[f(t) + f(t+h)\right] \end{aligned}$$

NB1: The trajectories generated by Verlet and velocity Verlet are identical (apart from round-off errors)

NB2: VV uses the force at time t+h. This force can only be calculated if it does not depend on the velocity! Often this is the case but there are situations in which it is not: Lorentz-force, friction due to a solvent, certain type of thermostats,...). In these cases one needs to solve the 2nd equation iteratively. See, e.g., book of Allen and Tildesley for details.

Verlet algorithm: Step size

$$\begin{aligned} r(t+h) &= r(t) + hv(t) + \frac{h^2}{2}\frac{f(t)}{m} \\ v(t+h) &= v(t) + \frac{h}{2m}\left[f(t) + f(t+h)\right] \end{aligned}$$

Choice of time step h: positions of VV are correct up to $O(h^3)$ ⇒ we need that

$h^3/3!\ d^3r/dt^3 \ll h^2/\ 2\ d^2r(t)/dt^2$ Too complicated!

OK if $h^2 f(t) \ll$ typical distances in the system.

Quick estimate: h ≈ 1% of the minimal vibration period in the system

More precisely: The error of *one* step is $O(h^3)$. The number of steps needed to propagate the system for a time t is t/h. Thus the error after time t is $O(h^2)$ ⇒ test of the code

Example: Fluctuation of total energy $\propto h^2$

Periodic Boundary Conditions

Computer resources are limited ⇒ it is not possible to simulate 10^{23} particles

Typical number of particles: 10^3–10^9

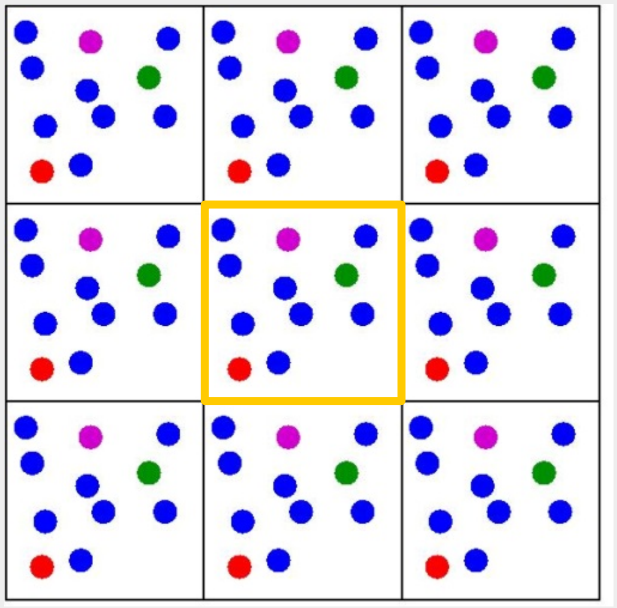
For a system with N particles in a cubic box we have $6 \cdot N^{1/3} \cdot N^{1/3}$ particles at the surface. Thus the fraction of particles that touch the surface is $6/N^{1/3}$.

For $N = 10^3$ this fraction is 0.6 and for $N = 10^6$ it is 0.06!

Thus in such a geometry one simulates mainly the surface and not the bulk!

⇒ use periodic boundary conditions

Periodic Boundary Conditions: 2



NB: Particle #1 (green) interacts with *all* other particles, i.e. all #2's (blue, red,…) and the other #1(green)!

Comments:

• With pbc we do not have open surfaces since all particles are "inside"

• In practice one simulates only one system: The central box. The positions of all other particles can be obtained from the position of the ones in the central box.

• If a particle leaves the central box on one side it re-enters it from the other side.

Periodic Boundary Conditions: 3

- Often the interactions between particles is very weak if their distance is $> L/2 \Rightarrow$ we can set it to zero. In this case particle #1 interacts only with *one* of the images of particles #2.

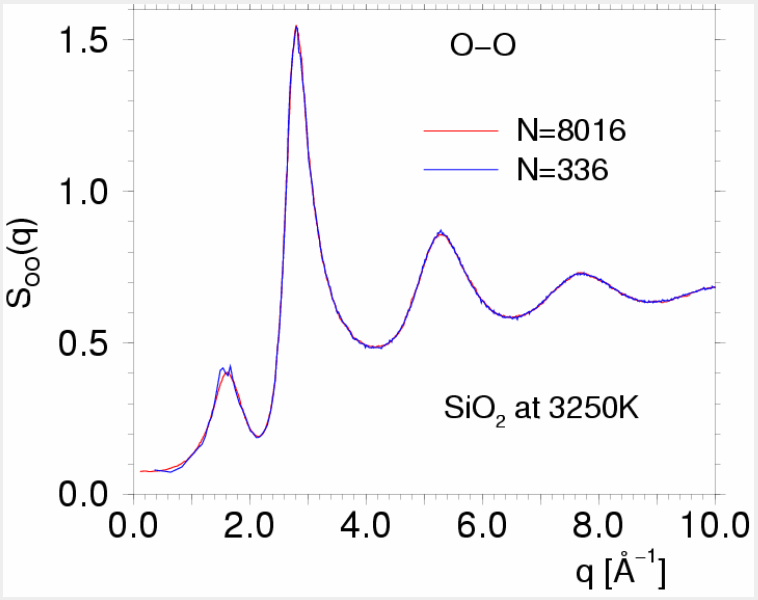
- NB1: Important exception: Coulomb interactions: $V(r) \propto 1/r$. Here one particle interacts with all the other images and this interaction can never be set to zero $\Rightarrow$ We need to sum up this infinite sum and this can be done in an exact way (Ewald summation). Computational cost is order $N^{1.5}$. For large systems one can use schemes (fast multipole methods, particle mesh Ewald,...) that have better scaling properties, i.e. O(N). But these approaches are somewhat more complex to code and the prefactor is not small. Typical cross-over numbers are 10^4 particles.

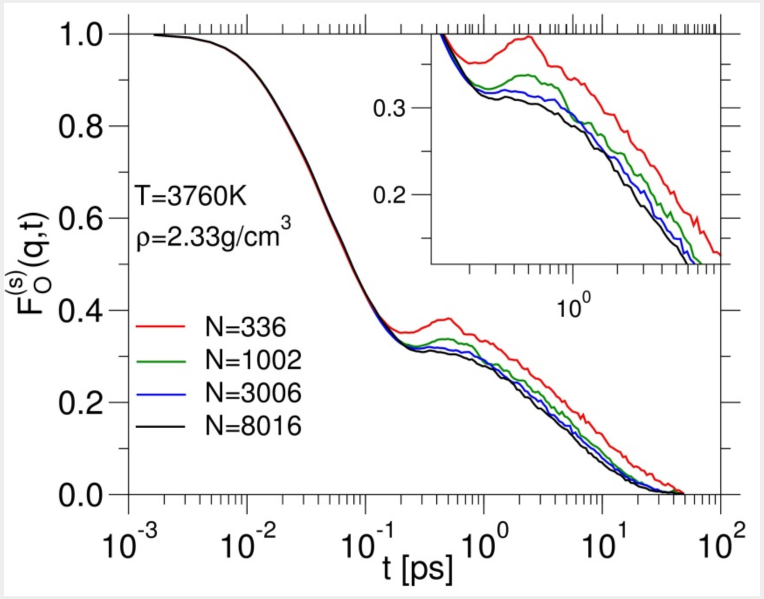
- NB2: Special problem with surfaces: The standard Ewald approach becomes very inefficient. Naive idea: Use normal 3d-Ewald in slab geometry + vacuum: Not very accurate if the spacing is not rather large. $\Rightarrow$ no really satisfactory solution is known $\Rightarrow$ avoid Coulomb!

Periodic Boundary Conditions: 4

- A system simulated with pbc is not really equivalent with a bulk system! Even if the interactions are shorter-ranged than L/2 we can have correlations that extent over larger length scales.

Examples:
1) g(r) in a dense fluid
2) System close to a critical point ⇒ correlation length diverges and one has to do a finite size scaling analysis.





- Even if the static properties do not depend on L, there are situations in which the dynamic properties do (diffusion coefficient,...)..

Neighbor Lists

Assume that particles have an interaction that has range r_c .

We want to calculate force on particle i ⇒ make loop over all particles ≠ i and sum the pair forces

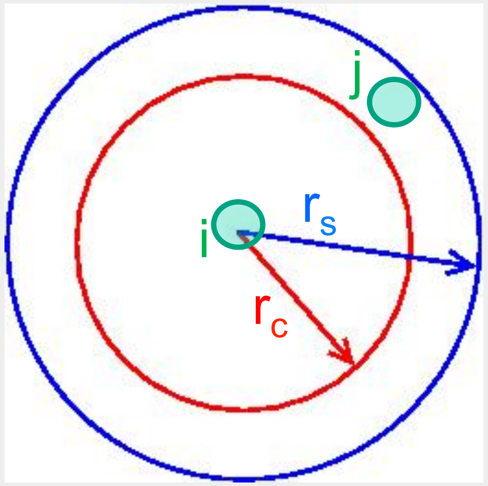
⇒ we need O(N) operations and thus $O(N^2)$ operations to calculate *all* the forces in the system

This is ok if N is not too large, but becomes inefficient for N ≥ O(200)
⇒ neighbor lists.

Two possibilities: "Verlet list" and "cell lists"

Verlet List

Assume that particles have an interaction that has range r_c ; pick a $r_s > r_c$



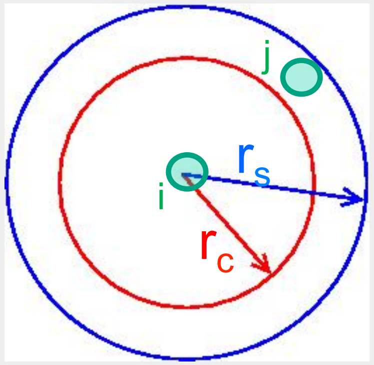
Idea:

•For each particle i one determines all particles j that have $|\mathbf{r}_i - \mathbf{r}_j| \leq r_s \Rightarrow$ list of neighbors of particle i. Cost to create this list is $O(N^2)$ or $O(N)$ (if one uses a cell list, see below).

•The force f_i is calculated by considering only the particles in the neighbor list (NB: Certain interactions will be zero) $\Rightarrow$ Cost for calculating all the forces scales like $O(N)$

•Neighbor list is calculated only rarely $\Rightarrow$ saves CPU time

Verlet List: 2



Reconstruct the list if $\max_k |\vec{r}_k(t+\tau) - \vec{r}_k(t)| \geq \frac{r_s - r_c}{2}$

t is the last time the neighbor list has been created

Comments:

• For "small" systems this approach is not efficient since basically every particle interacts with all the other particles (NB: "small" depends on interaction range and particle density

• Approach is very efficient if the particles move only slowly (crystals, viscous liquids, systems with a fixed topology such as polymer networks,...)

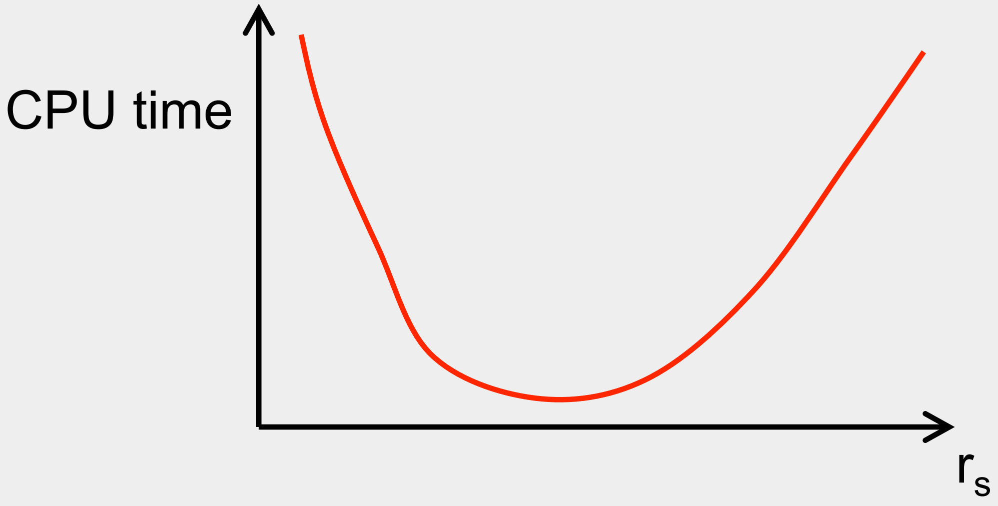
• For very large systems one should avoid the double do-loop to create the list and use a cell list approach (see below) to create a Verlet list.

•Problem of the black sheep: For very large systems ($N \geq O(10^5)$) there is often one particle that moves quite far and thus one has to update the neighbor list. Solution: Update only the neighbor list for the black sheep (and its neighbors) or forget about it.

Verlet List: 3

$$\max_k |\vec{r}_k(t+\tau) - \vec{r}_k(t)| \geq \frac{r_s - r_c}{2}$$

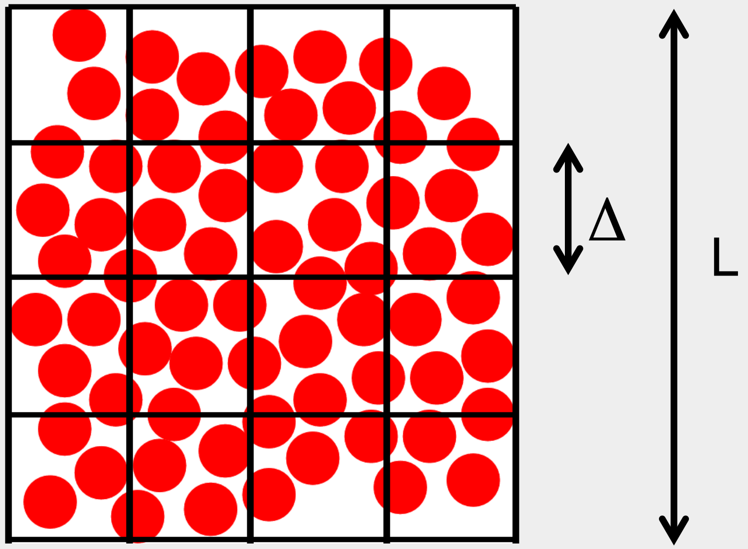
- Choice of r_s: If r_s is too small one has to update the neighbor list very often. If r_s is too large the neighbor lists are large and the calculation of the force f_i is slow. NB: Result depends on T!



E.g. Lennard-Jones fluid around triple point: $r_s \approx 1.3r_c$

Cell List

- Idea: Divide the simulation box of size L into small cubes of size Δ such that $\Delta \geq r_c$.



- Make loop over all the particles in order to determine in which cube they are ⇒ to each cube we can associate a list of particles that are in this cube.

⇒ particles that interact with particle i must be in one of the cubes that are
touching the cube of particle i

Computational cost: O(N). NB: These list can also be used to calculate a Verlet neighbor list.

Cell Lists: 2

- List is used to calculate the force on a given particle i:
  - -see in what cube is particle i
  - -loop over the 26 cubes around the cube of particle i + central cube
  - -loop over all the particles that are in a given cube

  Computational effort for this loop is independent of N $\Rightarrow$ overall algorithm scales like O(N).

•Calculation of forces is not very efficient: One has to test all the particles in the 27 cubes, i.e. all particles that are inside a volume of $(3\Delta)^3 = 27\Delta^3$. The volume in which the interactions are non-zero is $4\pi/3r_c^3$. In the best case we have $\Delta = r_c$ and thus we loose a factor of $27/4 \approx 7$.

- Method is efficient if L/Δ is large (in practice ≥ 5)

Thermostats

Usually we want the properties of the system at a given temperature T $\Rightarrow$ need a method to set the temperature. Recall: Temperature is related to mean kinetic energy of the particles

Thermostat of Andersen

Idea: At each time step each particle has a probability $\nu \leq 1$ to be coupled to an external heat bath that has temperature T , i.e. its velocity is replace by a gaussian random number:

$$p(v) \propto \exp(-mv^2/k_B T)$$

Thus the kinetic DOF will have the right values and due to the coupling between momentum and coordinates the system will go to an equilibrium state at with temperature T .

Problem: The trajectory of the particles becomes a bit erratic, since they are kicked $\Rightarrow$ dynamics is perturbed. Also, the center of mass is not at rest anymore and thus starts to undergo a diffusive motion (inacceptable for very long runs if one wants to get out the dynamics/diffusion constant, so one has to correct for this by rescaling the velocities of the other parti-cles). However, if ν is sufficiently small, the effect is mild. (But the static properties are ok!); Better solution "Nose-Hoover thermostat

Potentials

- Important quantity in a simulation: How do the particles interact with each other (=interaction potential) ?

Two possibilities:

1. Ab initio calculation: Position of the atoms $\Rightarrow$ electronic structure of the system (within approximations!) $\Rightarrow$ Potential; computationally very expensive
2. Effective potential: Form of potential is ad hoc; parameters are obtained from fitting experimental data or results from ab initio calculations; relatively cheap

Potentials: 2

The large majority of interactions goes to zero only if $r \rightarrow \infty$ (Coulomb, van der Waals, Lennard-Jones,...). But most potentials become very small if the distance between particles is large $\Rightarrow$ One can set them to zero beyond a certain cut-off distance r_c . E.g. for a pair potential one has:

$$V(r) = V(r) - V(r_c) \quad \text{if } r < r_c \qquad \text{and } V(r)=0 \ \text{ if } r > r_c$$

"truncation and shifting"

$\Rightarrow$ V(r) is a smooth function but not differentiable at r_c .

Consider therefore the product V(r) S(r) with

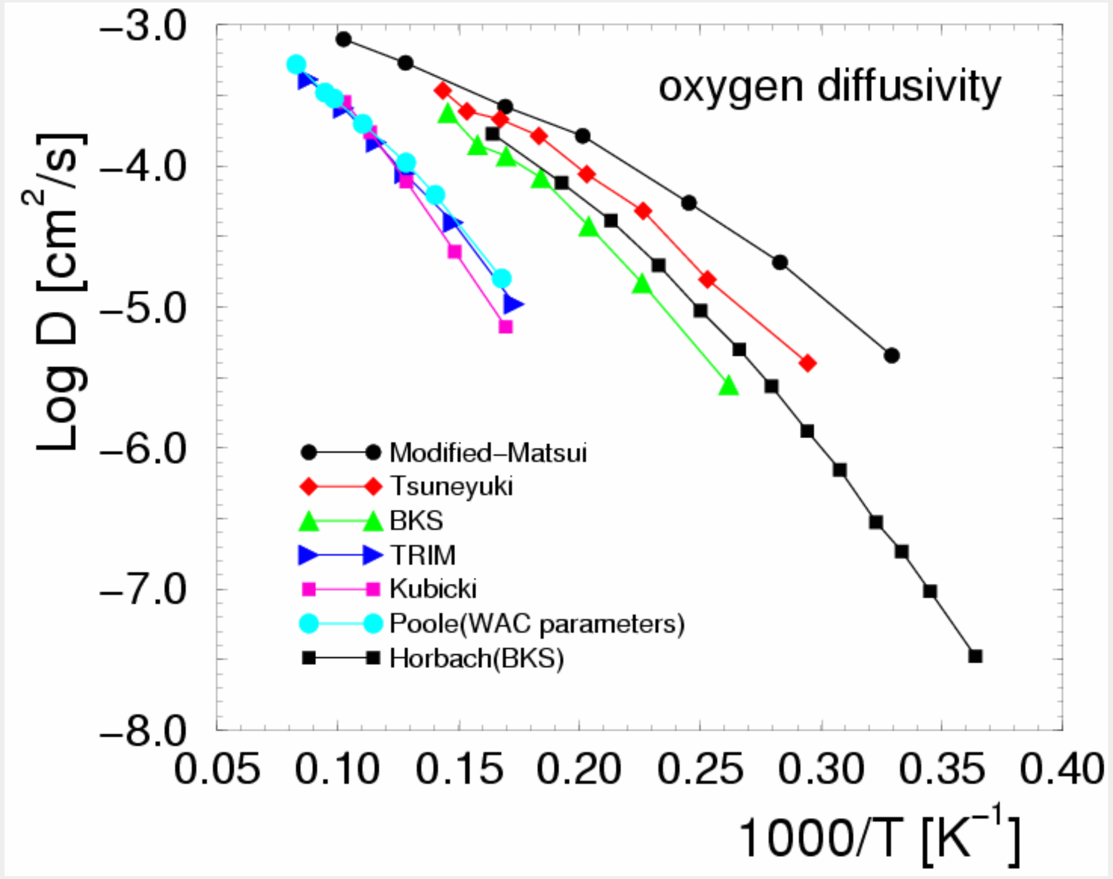
$$S(r) = \exp(-A^2/(r_c - r)^2)$$

Product is differentiable and smooth $\Rightarrow$ numerical integration can be done with larger time steps

Importance of potential

- What is the influence of the potential on the results?

- Hemmati and Angell have calculated for various models for SiO_2 the diffusion constant at different temperatures
- NB: structure for the different models is quite similar!!



- Large discrepancy between the different potentials

⇒ It is important to use *reliable* potentials

How to make a simulation

1. Think!

- What do I want to learn from the simulation?
- Can I do it?

Power of one processor: 10^{12} steps/particle/month

$\Rightarrow$ 10^3 particles for 10^9 steps (= (20Å)^3 for 1μs) or

10^6 particles for 10^6 steps (= (200Å)^3 for 1ns).

- Problems with system size?
- Problems with equilibration?
- Can/should I use MC to equilibrate?
- Need parallel computer?
- What are the results?!!!

How to make a simulation: 2

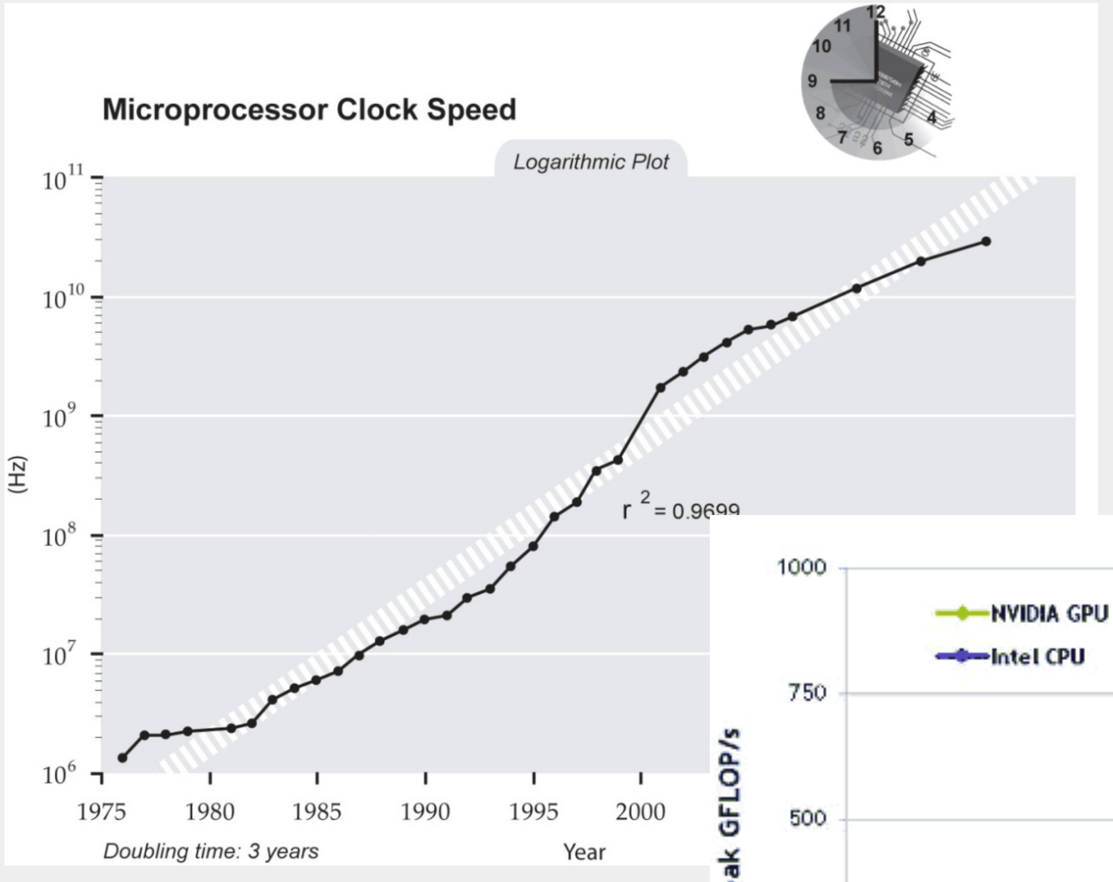
2. Write code

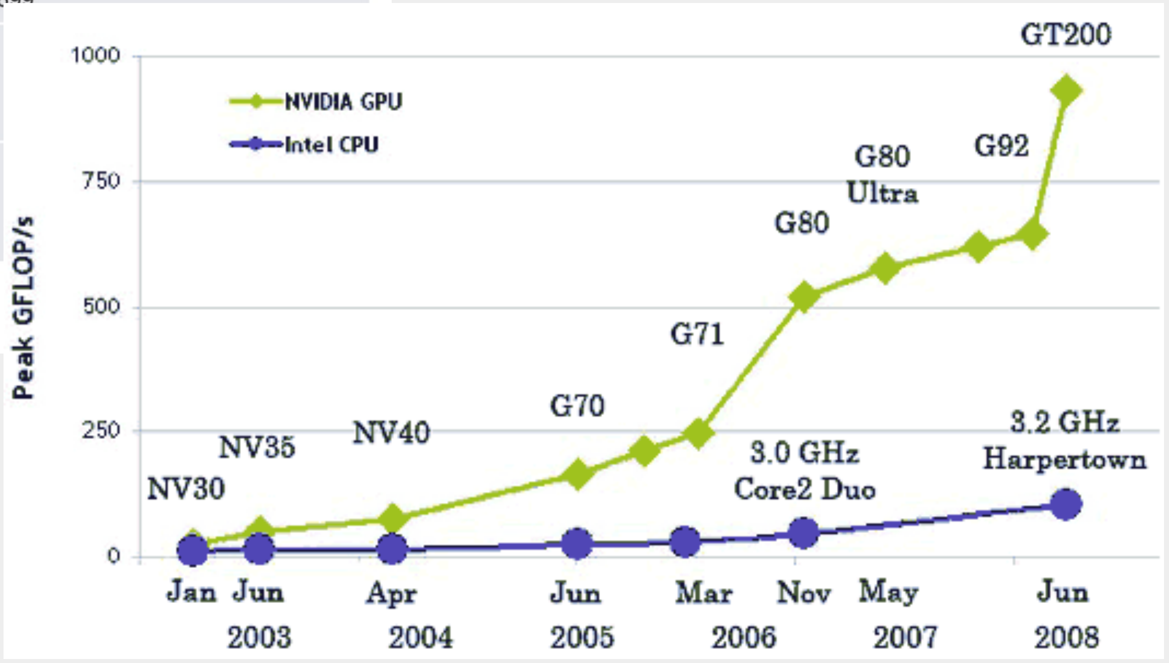
```
•Read input parameters
• Read start configuration
• Write input parameter to “log-file”
• do time=1,$t_{end}$
      − make one step
      − influence the system if needed (energy, temperature, pressure)
      − write configuration if needed
      − calculate certain quantities on the fly
      − write to “log-file”
  enddo  time
•  write final configuration
```

• Check code
• Make production
• Analyze the data
•Estimate the error!
  A smooth curve is not necessarily correct!
•Average over several samples/runs (that are at the same state point!)
• Are the results the ones you expected? Be critical!!!

GPUs

GPU = Graphic Processor Unit : The graphic card is used as a parallel computer (and it is relatively cheap)





Today: GPU's are about 20 times faster than a high end processor: BUT..

Computer Simulations of glass-forming liquids and glasses

- Observables
  - Structure
  - Dynamics
  - Vibrations

Structure: Pair distribution function

Define radial distribution function G(**r**) : Probability that two particles in the system are a vector **r** apart.

$$\begin{aligned} G(\mathbf{r}) &= \frac{N}{V}\delta(\mathbf{r}) + \frac{1}{V}\sum_i \sum_{j\neq i} \langle \delta(\mathbf{r} + \mathbf{r}_i - \mathbf{r}_j)\rangle - \rho^2 \\ &= \rho\delta(\mathbf{r}) + \rho^2 g(\mathbf{r}) - \rho^2 \end{aligned}$$

$$g(\mathbf{r}) = \frac{1}{\rho N}\sum_i \sum_{j\neq i} \langle \delta(\mathbf{r} - \mathbf{r}_i + \mathbf{r}_j)\rangle$$

For an isotropic system g(**r**) depends only on r=|**r**| ; divide g(**r**) by the phase space factor $4\pi\ r^2$:

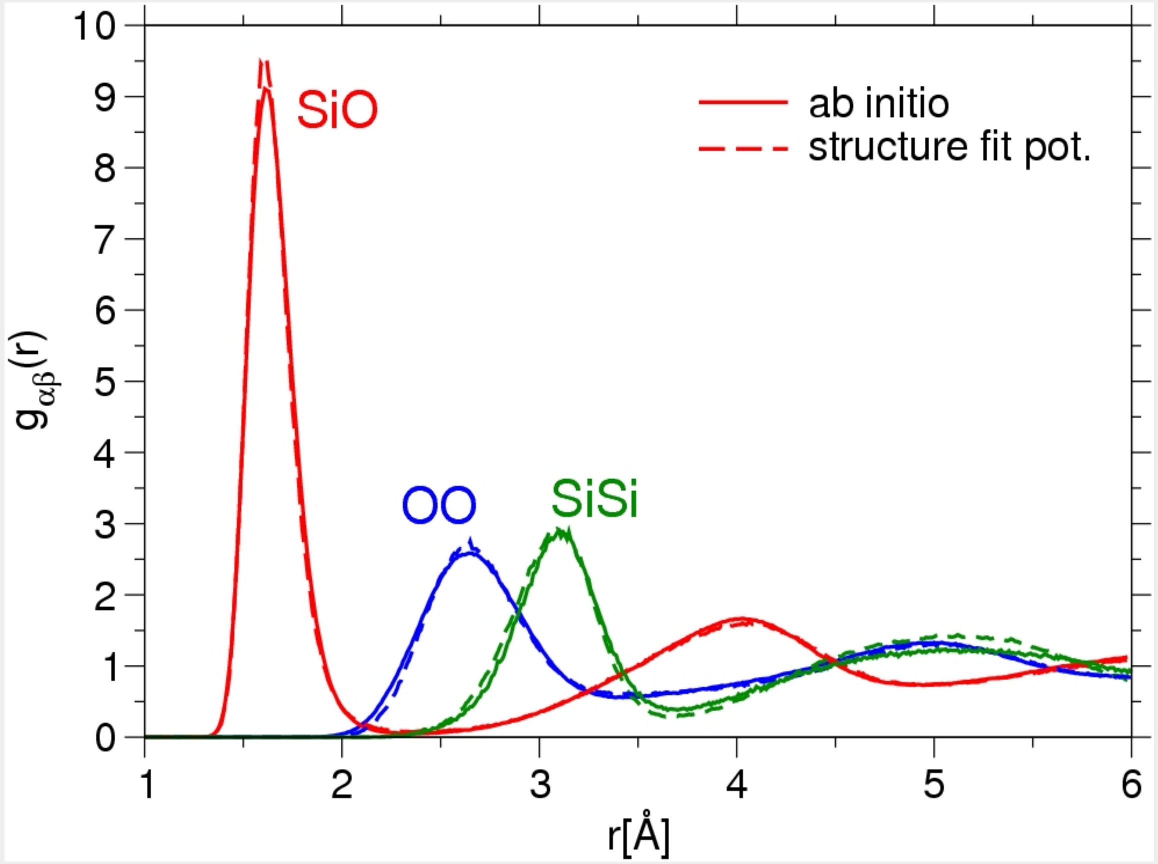
$\Rightarrow$ pair distribution function g(**r**): $$g(r) = \frac{1}{4\pi r^2 \rho N}\sum_i \sum_{j\neq i} \langle \delta(r - |\mathbf{r}_i - \mathbf{r}_j|)\rangle$$

g(r): relative probability that there is a particle at distance r from the origin *if* there is a particle at the origin; $g(r \rightarrow \infty) = 1$

Structure: Radial distribution function

- Definition of g(r) is easily generalized to multicomponent systems:

$$g_{\alpha\beta}(\mathbf{r}) = \frac{N}{\rho N_\alpha N_\beta} \sum_{i}^{N_\alpha} \sum_{j \neq i}^{N_\beta} \langle \delta(\mathbf{r} - \mathbf{r}_i + \mathbf{r}_j) \rangle$$



- Example of SiO_2
- position of peaks give the most probable distance
- NB: g(r) is a very useful quantity but usually not accessible in experiments (but for colloidal systems it is)

Structure: Structure factor

- g(r) is useful to characterize the *local* structure; for structures on larger scales it is not useful $\Rightarrow$ Use static structure factor

Define $\rho(\mathbf{k}) := \sum_j \exp(-i\mathbf{k}\cdot\mathbf{r}_j)$

The vector **k** is called “wave-vector”

- Define structure factor

$$S(\mathbf{k}) := \frac{1}{N}\langle \rho(\mathbf{k})\rho(-\mathbf{k})\rangle = \frac{1}{N}\sum_j\sum_l \langle \exp[-i\mathbf{k}\cdot(\mathbf{r}_j - \mathbf{r}_l)]\rangle$$

NB: S(**k**) is reel

Simple manipulations $\Rightarrow$

$$S(\mathbf{k}) = 1 + \rho\int d\mathbf{R}\exp[-i\mathbf{k}\cdot\mathbf{R}]g(\mathbf{R})$$

$\Rightarrow$ S(**k**) is the space FT of g(**R**)

For isotropic systems g(**R**) depends only on R

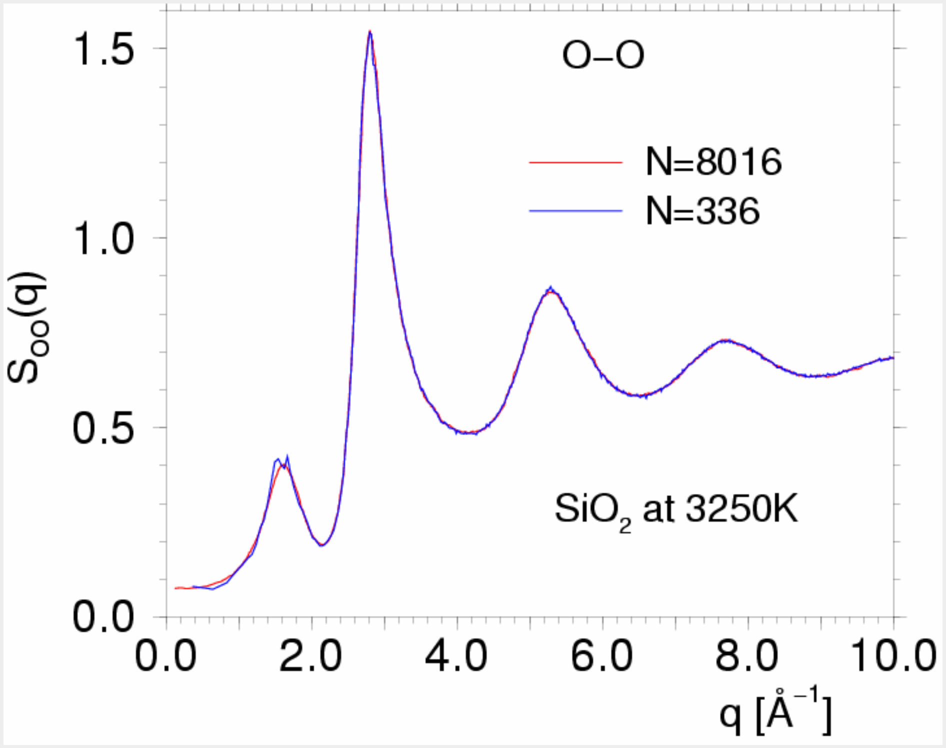
$$S(k) = 1 + \rho\int_0^\infty g(R)\frac{\sin(kR)}{k}4\pi R dR$$

Structure: Structure factor 2

Multicomponent systems: ⇒ partial structure factors

$$S_{\alpha\beta}(\mathbf{k}) := \frac{f_{\alpha\beta}}{N} \sum_{j}^{N_\alpha} \sum_{l}^{N_\beta} \langle \exp[-i\mathbf{k}\cdot(\mathbf{r}_j - \mathbf{r}_l)] \rangle$$

$f_{\alpha\alpha}$=1; $f_{\alpha\beta}$=1/2 for $\alpha \neq \beta$

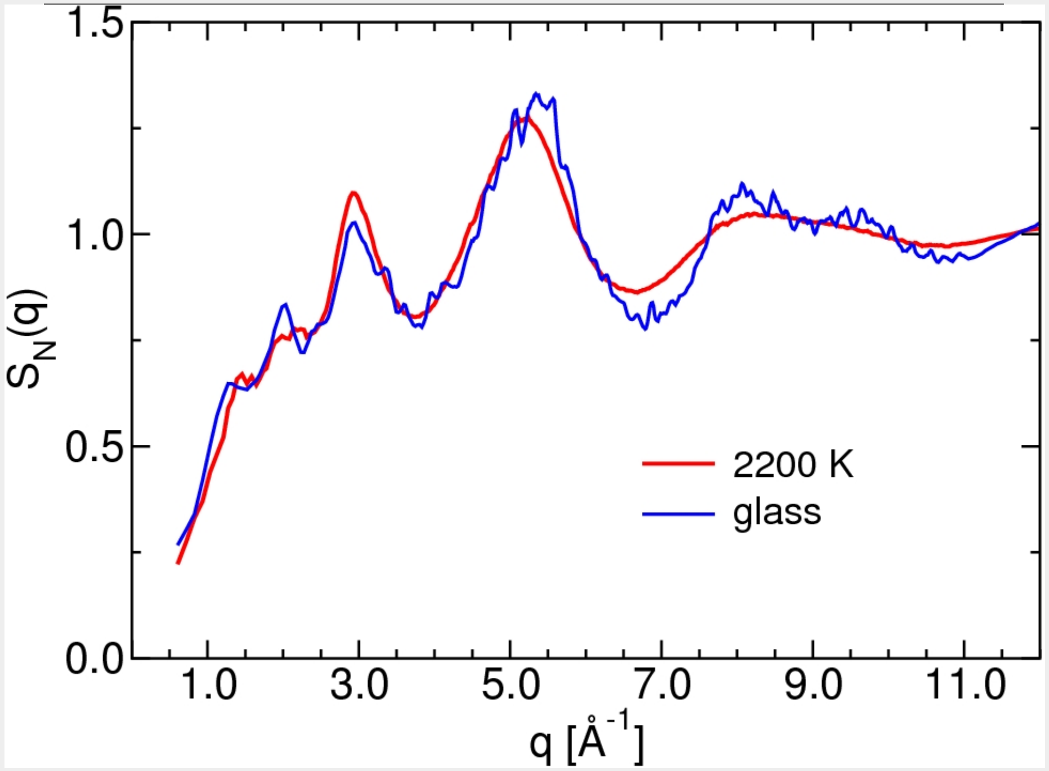


Structure: Neutron structure factor

- Experiments do usually not allow to measure the partial structure factors directly; within a neutron scattering experiments one measures

$$S^{\mathrm{neu}}(\mathbf{k}) = \frac{N}{\sum_\alpha N_\alpha b_\alpha^2} \sum_{\alpha,\beta} b_\alpha b_\beta S_{\alpha\beta}(\mathbf{k})$$

where the constant b_α is the "neutron scattering cross section" for an element of type α (see www for values of b_α)



Na-Si-O system (=NBS)

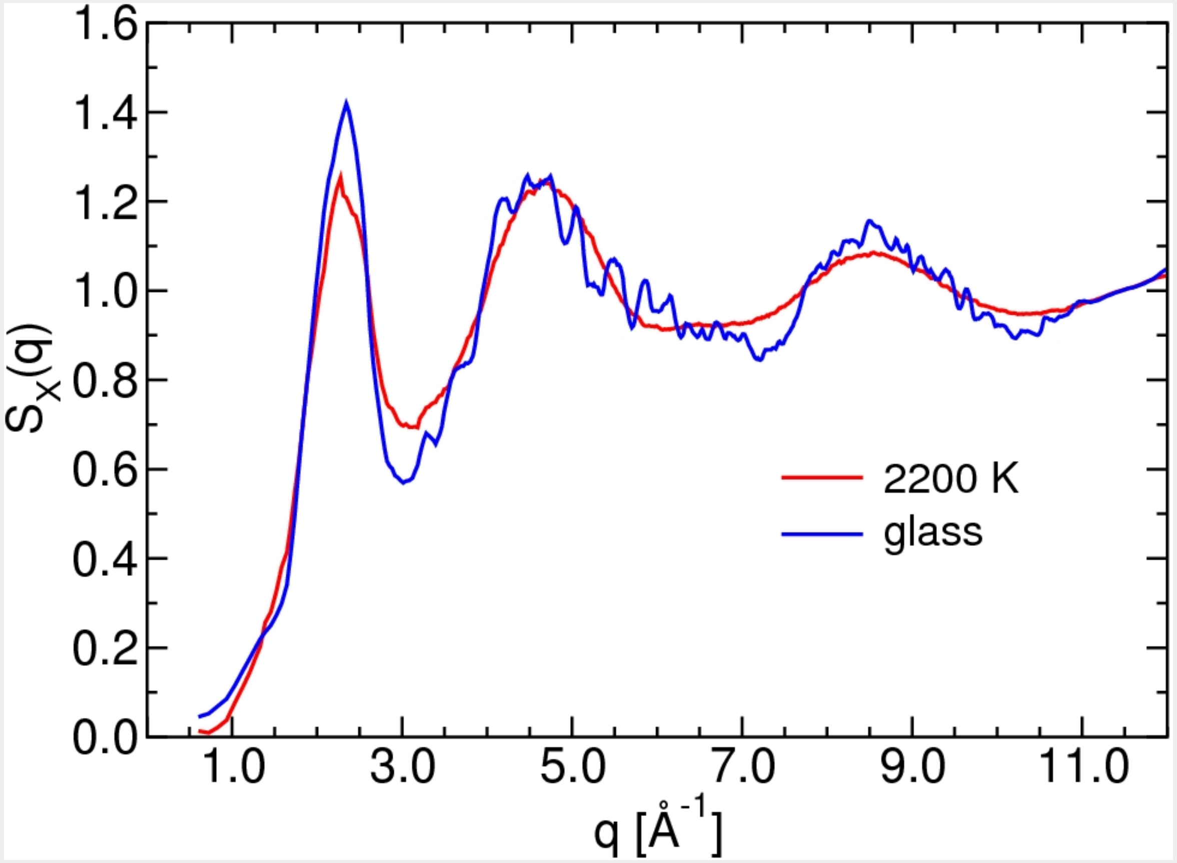
- SiO_2 with BKS potential (Horbach et al.)

Structure: X-ray static structure factor

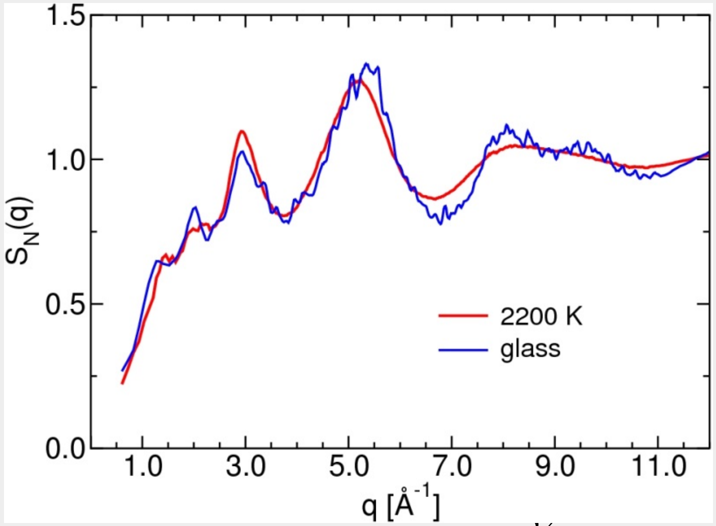
- Similarly for X-rays:

$$S^{\mathrm{xr}}(\boldsymbol{q}) = \frac{N}{\sum_{\alpha} N_\alpha x_\alpha^2(q)} \sum_{\alpha\beta} x_\alpha(q) x_\beta(q) S_{\alpha\beta}(\boldsymbol{q})$$

where the functions $x_\alpha(q)$ are known



- NBS system
- $S_X(q)$ has much less structure than $S_N(q)$



Structure: S(k) and g(r)

Since S(**k**) is the space FT of g(**R**) we have that g(**R**) is the FT of S(**k**)

$$g(\mathbf{R}) = 1 + \frac{1}{(2\pi)^3\rho}\int [S(\mathbf{k}) - 1]\exp(i\mathbf{k}\cdot\mathbf{R})d\mathbf{k}$$

For an isotropic system we can make the angular average:

$$g(R) = 1 + \frac{1}{2\pi^2\rho}\int [S(k) - 1]\frac{\sin(kR)}{R}kdk$$

From S(k) one can *in principle* obtain g(r) WARNING!!! Very unreliable!!!

Same issue: Should not calculate S(k) from FT of g(r). Instead use the definition of S(k):

$$S(\mathbf{k}) := \frac{1}{N}\langle\rho(\mathbf{k})\rho(-\mathbf{k})\rangle = \frac{1}{N}\sum_j\sum_l\langle\exp[-i\mathbf{k}\cdot(\mathbf{r}_j - \mathbf{r}_l)]\rangle$$

Note: use only **k**-vectors of the form

$$\mathbf{k} = \frac{2\pi}{L}\begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix}$$

Structure: Compressibility

- From S(k) one can obtain the compressibility κ_T

$$S(|\mathbf{k}| \to 0) = \rho k_B T \kappa_T$$

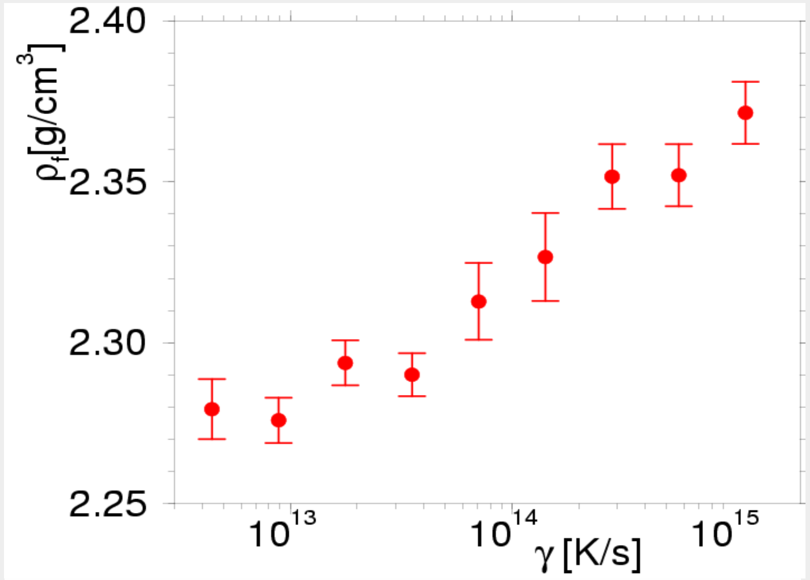
- Multicomponent systems:

$$\sum_{\alpha,\beta} S_{\alpha\beta}(|\mathbf{k}| \to 0) = \rho k_B T \kappa_T$$

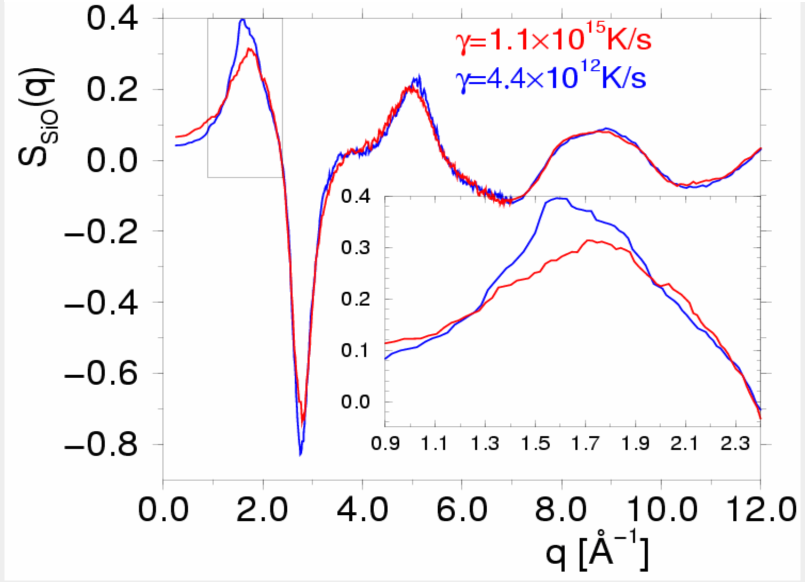
Not all glasses are the same!

- To produce a glass a liquid has to be cooled below the glass transition temperature with a certain cooling rate

  $\Rightarrow$ properties of glasses depend on their history



•Cooling rate dependence of the density of amorphous SiO_2 after a quench to 0K at constant pressure



•Cooling rate dependence of the structure factor of amorphous SiO_2 after a quench to 0K

•One needs to be able to equilibrate also at very low T or one must take into account these effects

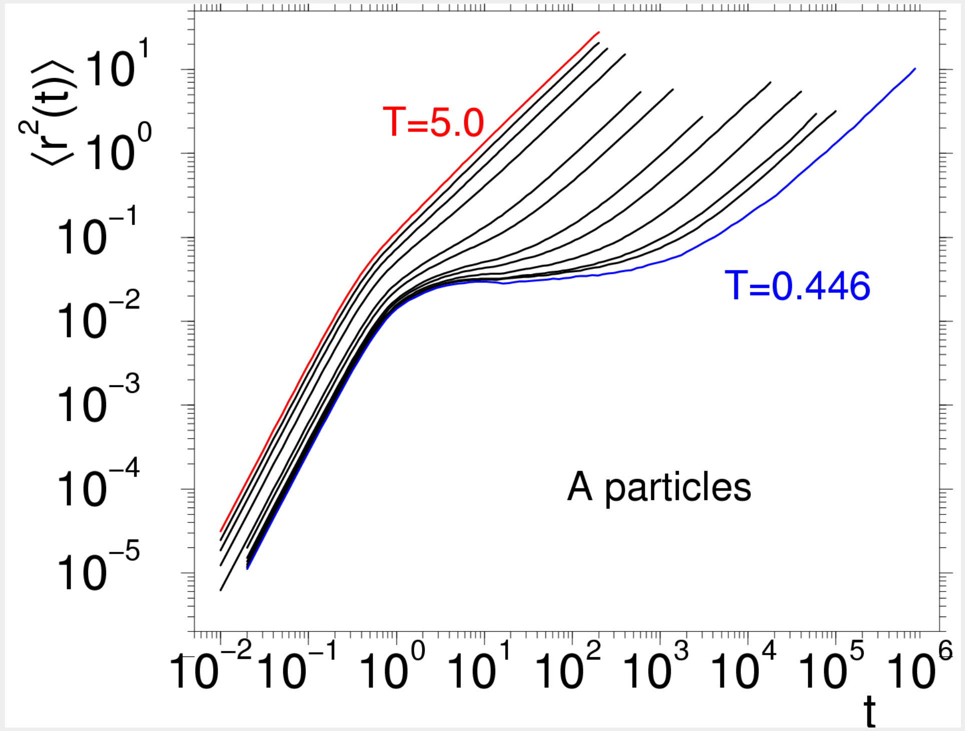
Dynamics: MSD

- Use mean squared displacement to characterize the dynamics

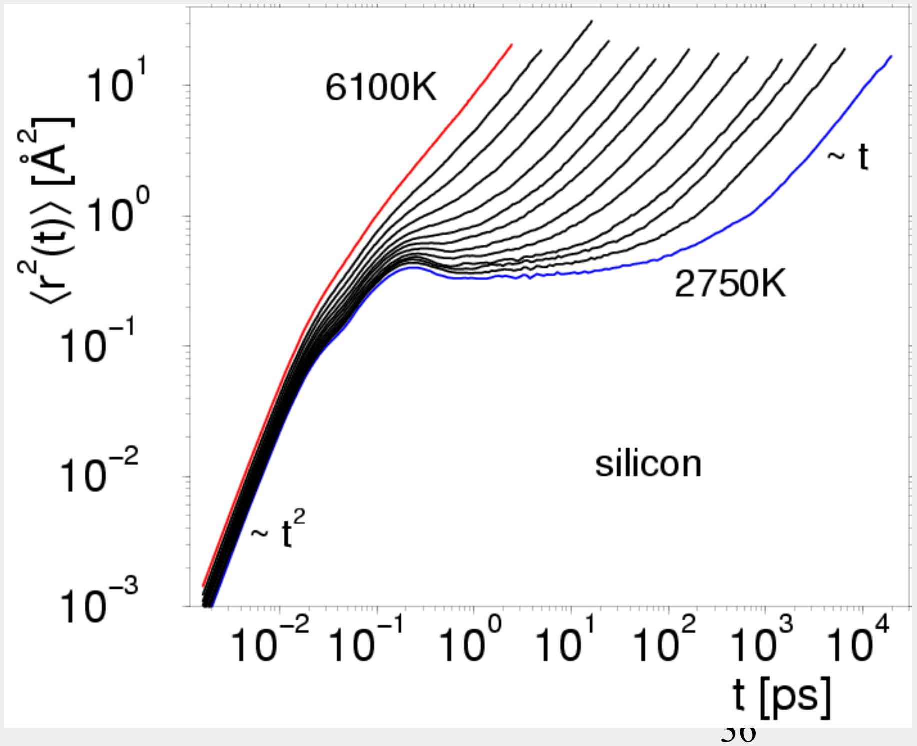
$$\langle r^2(t)\rangle = \langle|\mathbf{r}_i(t) - \mathbf{r}_i(0)|^2\rangle$$

- MSD at short times: $\mathbf{r}_i(t) \approx \mathbf{r}_i(0) + \mathbf{v}_i t + \mathbf{f}_i(t)\frac{t^2}{2}$

$\Rightarrow$ ballistic motion $\Delta^2(t) = \langle|\mathbf{r}_i(t) - \mathbf{r}_i(0)|^2\rangle \approx \langle \mathbf{v}_i^2\rangle t^2 = \frac{3k_BT}{m}t^2$



Ni in $Ni_{80}P_{20}$



Si in SiO_2

Time dependent correlation functions

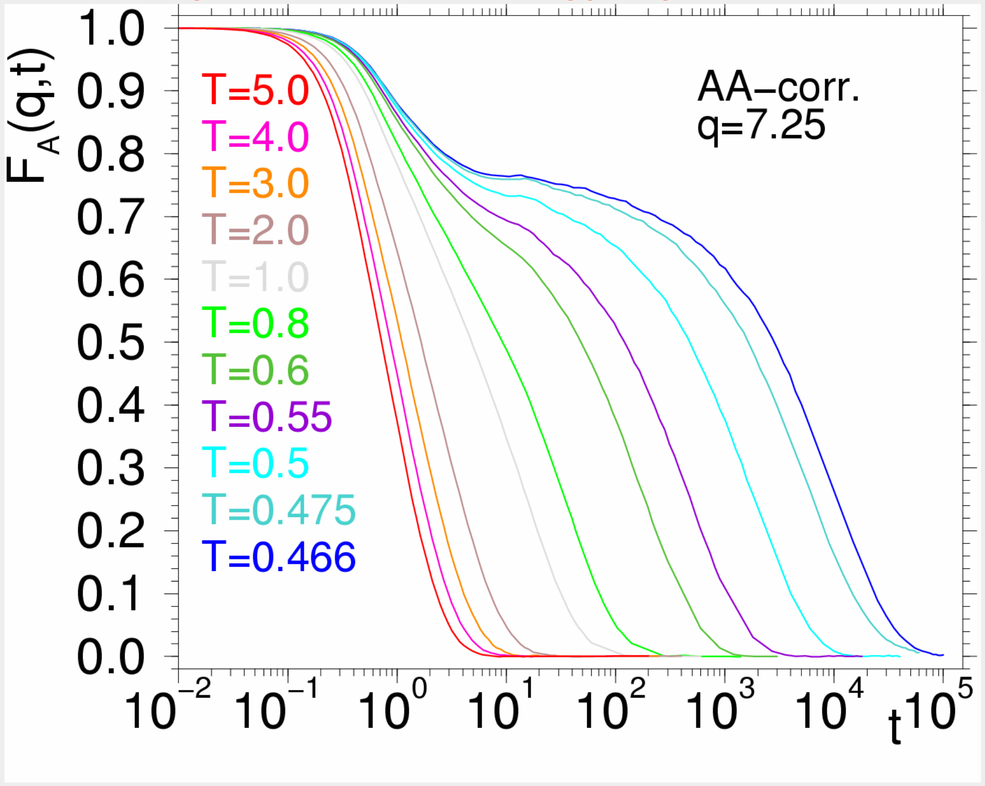
- At any time there are equilibrium fluctuations in the density distribution; how do these fluctuations relax?

- Consider the incoherent intermediate scattering function $F_s(q,t)$

$$F_s(q,t) = N^{-1} \left\langle \delta\rho_s(-q,t)\, \delta\rho_s(q,0) \right\rangle \quad \text{with} \quad \delta\rho_s(q,t) = \exp(i\mathbf{q}\cdot\mathbf{r}_k(t))$$

- $F_s(q,t)$ can be measured in inelastic n- and X scattering

$F_s(q,t)$ for Ni in $N_{i80}P_{20}$



- high T: after the microscopic regime the correlation decays exponentially

- low T: existence of a plateau at intermediate time (reason: cage effect)

- N.B.: plateau can only be seen on a logarithmic time axis!

Vibrational density of states

- At low T the atoms in the glass oscillate around their equilibrium position. If T is sufficiently low these oscillations are harmonic.
- A system of N particles in d=3 has 3N harmonic modes with frequencies ω_i, the eigen-frequencies

$$\mathbf{r}_j(t) = \sum_{k=1}^{3N} A_{jk} \exp[i(\omega_k t + \phi_k)]$$

- The density of states is $g(\omega) = \frac{1}{3N} \sum_{i=1}^{3N} \delta(\omega - \omega_i)$

- Experimentally one has access to $\tilde{g}(\omega) = C(\omega) g(\omega)$

where the function $C(\omega)$ depends on the type of experiment (neutron, infrared,…)

Vibrational density of states: 2

- Equation of motion $\mathbf{M}\ddot{\mathbf{r}} = \mathbf{F} = -\nabla V$

  with the mass matrix **M** $\mathbf{M} = \begin{pmatrix} m_1 & 0 & 0... \\ 0 & m_2 & 0... \\ & .... & \\ 0 & .... & m_N \end{pmatrix}$

- Consider deviations from the local minimum $\mathbf{R}_0$:

$$\mathbf{r}(t) = \mathbf{R}_0 + \mathbf{u}(t) \to \ddot{\mathbf{r}}(t) = \ddot{\mathbf{u}}(t)$$

- Taylor expansion of the potential

$$V(\mathbf{r}) = V(\mathbf{R}_0) + \nabla V(\mathbf{R}_0) \cdot \mathbf{u} + \frac{1}{2}\mathbf{u}\nabla^2 V(\mathbf{R}_0)\mathbf{u}$$

$$\Rightarrow \mathbf{M}\ddot{\mathbf{u}} = -\nabla^2 V\mathbf{u}$$

Ansatz $\mathbf{u}(t) = \mathbf{u}_0 \exp(-i\omega t)$ $\Rightarrow$ $\mathbf{M}\omega^2\mathbf{u}_0 = \mathbf{D}\mathbf{u}_0$

with the dynamical matrix $\mathbf{D} := \frac{\partial^2 V}{\partial u_i \partial u_k}$

$\Rightarrow$ eigenvalues ω_i and DOS ; NB the eigenvectors $\mathbf{u}_0$ give information on the nature of the vibrations

Vibrational density of states: 3

- Large systems (N> O(10^4)): dynamical matrix becomes too large to be inverted

$\Rightarrow$ use connection between DOS and time-FT of velocity autocorrelation function

$$g(\omega) = \frac{1}{N k_B T} \sum_j m_j \int_{-\infty}^{\infty} dt \exp(i\omega t) \langle \mathbf{v}_j(t) \cdot \mathbf{v}_j(0) \rangle$$

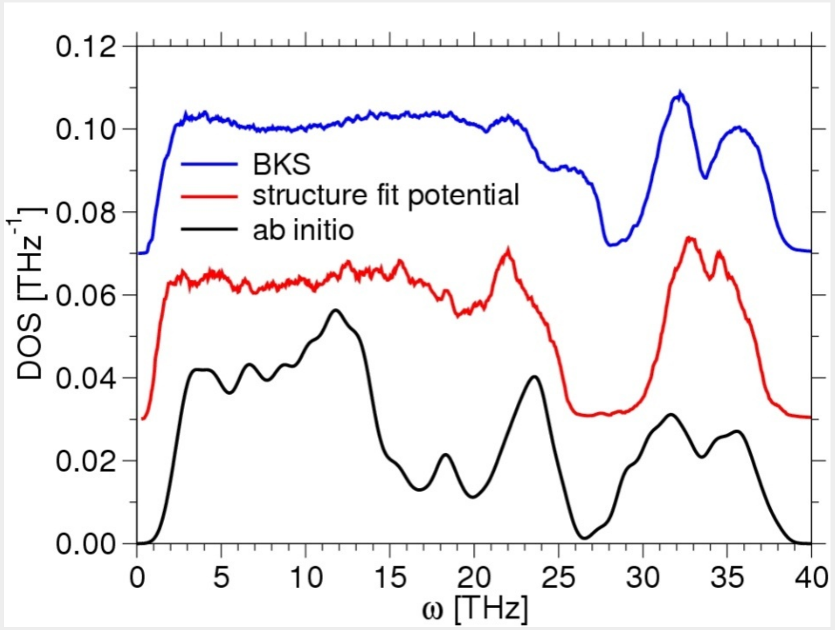
NB:

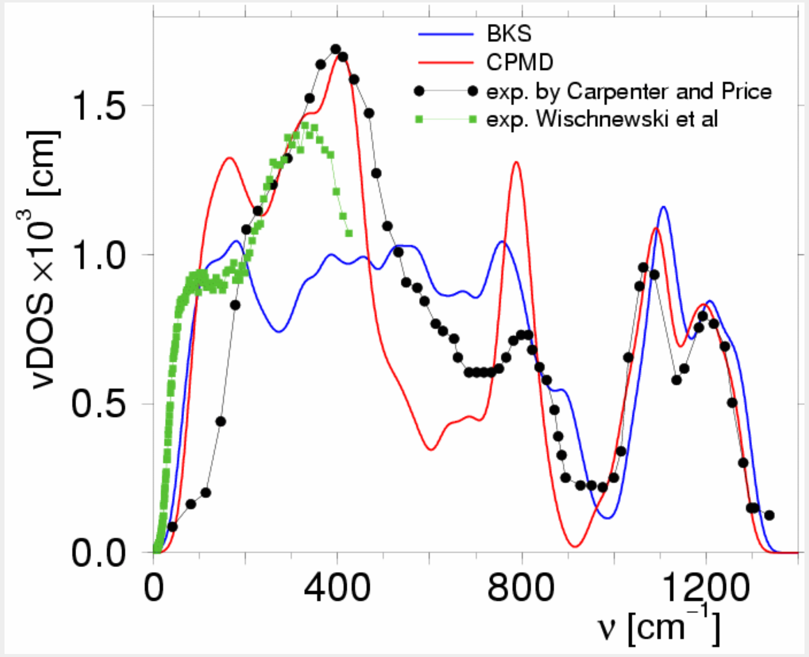
1) In practice one does not calculate the correlation function but makes the FT of the velocities (Wiener-Kinchine theorem)

2) Formula allows to decompose the DOS into partial DOS for the elements

3) Approach gives no information on the eigenvectors

vDOS for silica





- shape of vDOS depends on potential

- for SiO_2 most effective potentials do not give a good description of the vDOS at intermediate frequencies

Bottom-line

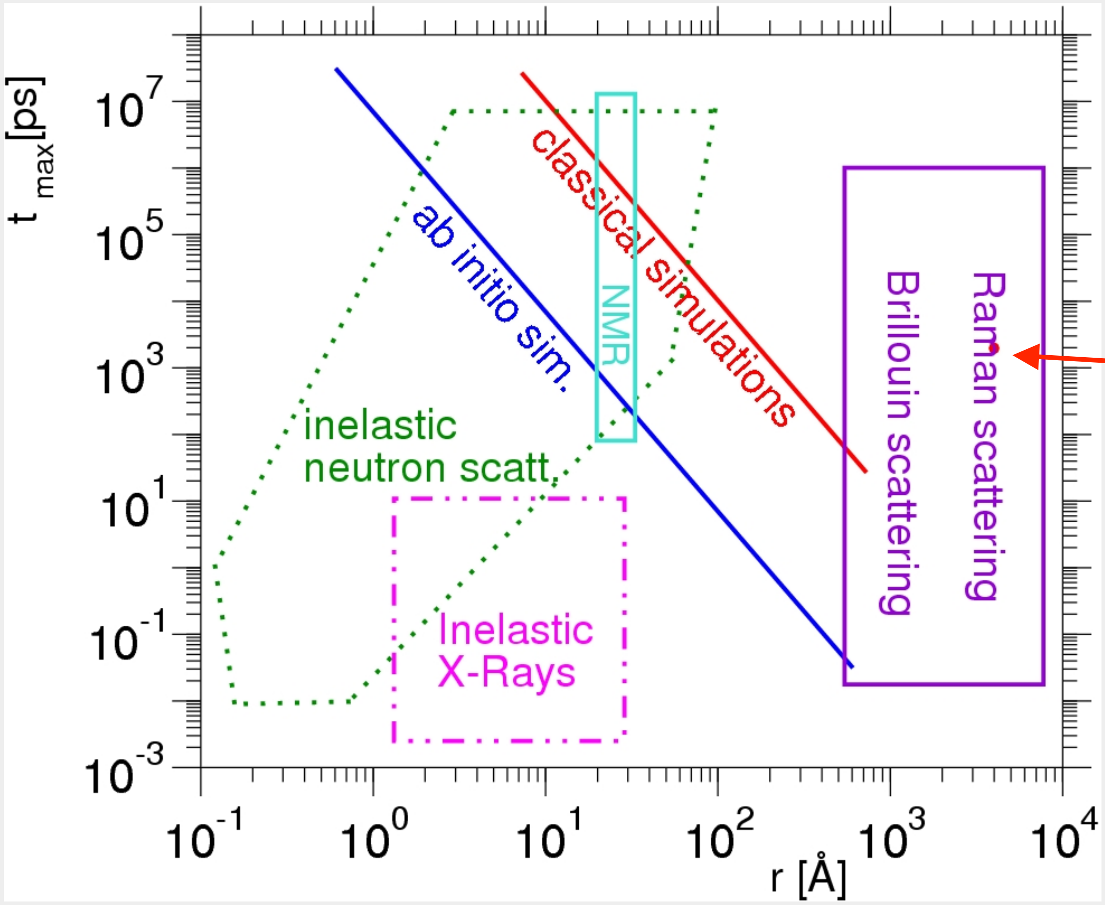
Two major problems in simulations of glasses:

1) Accuracy of interaction potentials
   - We need accurate, transferable (and simple) potentials that allow to describe reliably also complex systems on large length and time scales; thanks to the *ab initio* simulations there is (slow) progress

2) Time scales:
   - For some problems we need to be able to access longer time scales (seconds, hours,…) since the processes of interest are slow
   - We need to be able to generate glasses that have fallen out of equilibrium at lower temperatures
   - Some ideas are around, but so far progress is *very* slow (parallel tempering, genetic algorithms, ...)

Present day computer simulations

- Processor speed still (!) increases with time + multicore processors



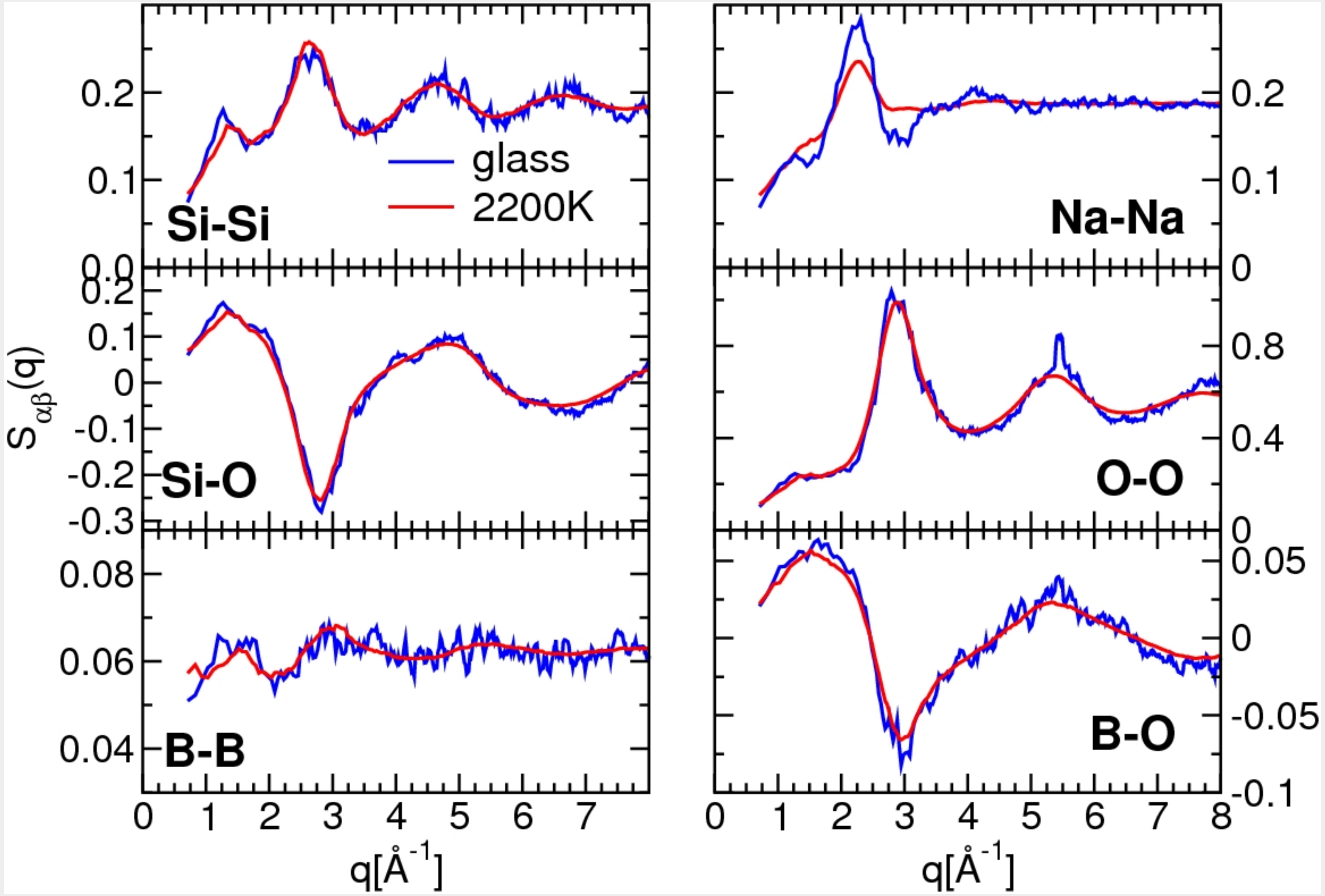
4000 years of CPU time 10^{10} particles

Large scale computer simulations (= one month of CPU time)

- 10^{10} time steps for 100 particles (10μs, 10Å) **classic!!**
- 10^5 time steps for 10^7 particles (100ps, 400Å)

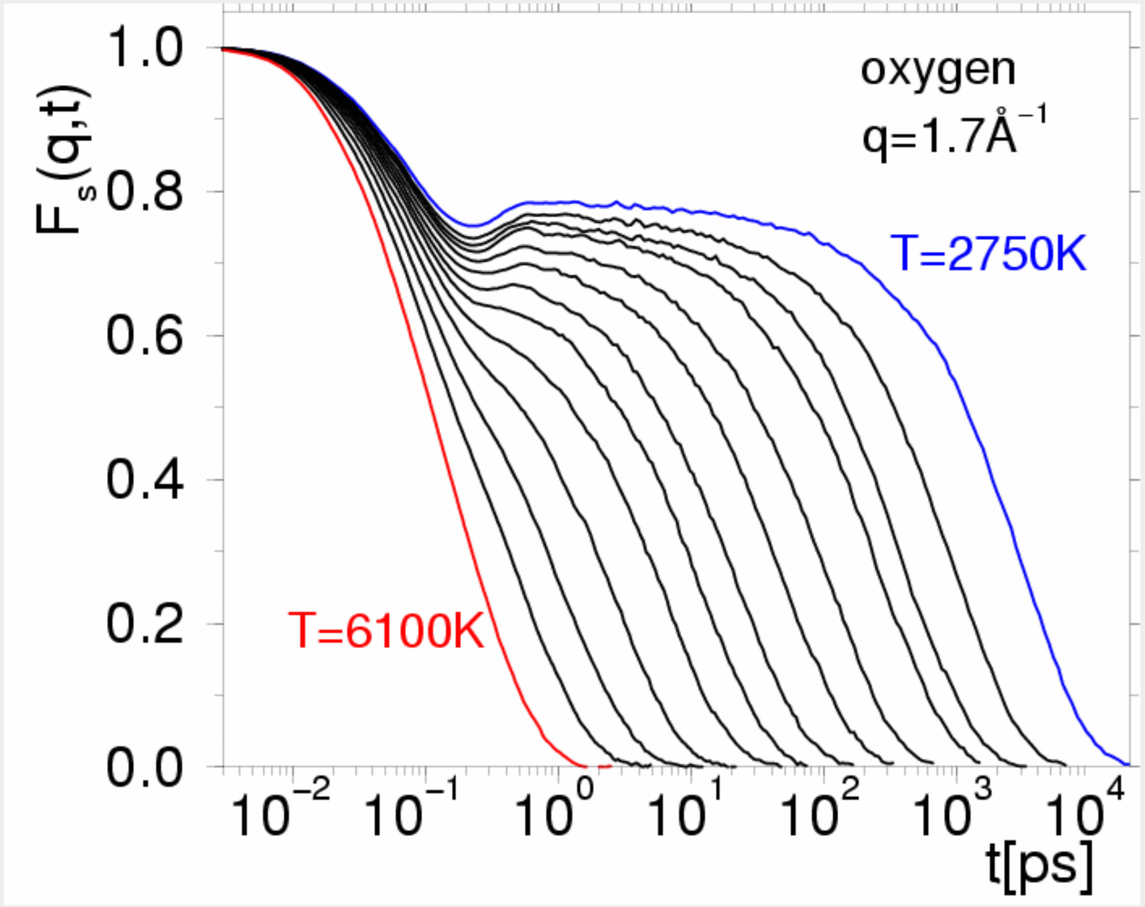
Structure: Structure factor 3

- $3Na_2O$-B_2O_3-$6SiO_2$; ab initio, 320 atoms ⇔ 60 Si, 180 O, 60 Na, 20 B; 60 ps trajectory; 45 years of CPU time



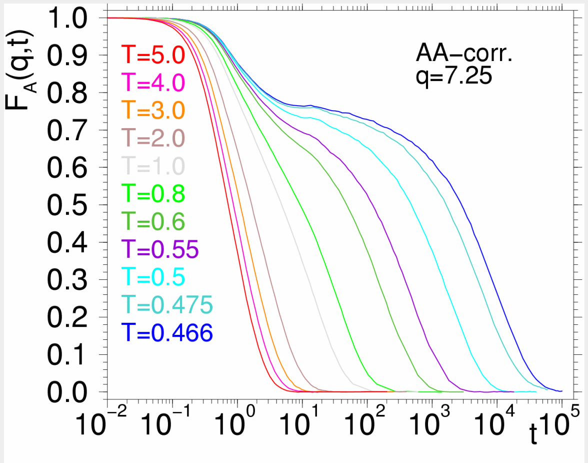
Time dependent correlation functions: 2

- $F_s(q,t)$ in SiO_2



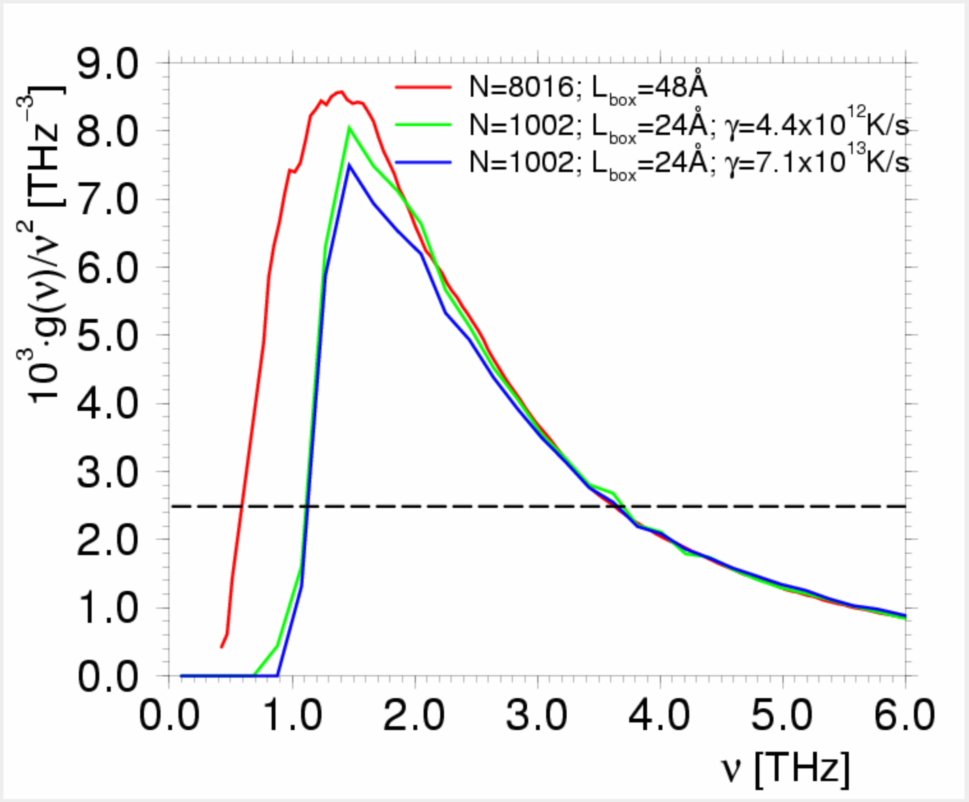
- high T: exponential relaxation
- low T: relaxation in two steps (because of cage effect); long times: relaxation is stretched
- intermediate times: viscoelastic effects, Boson peak

NB: we are above the melting temperature T_m = 2000K !



Importance of system size

• At small ν, g(ν) is expected to scale like ν^2 (Debye); many glass-forming systems shown an anomalous increase of g(ν) over the Debye-level ⇒ Boson peak



•Even for the largest systems g(ν) does not show the expected Debye behavior at small ν

•g(ν) depends strongly on system size and on cooling rate

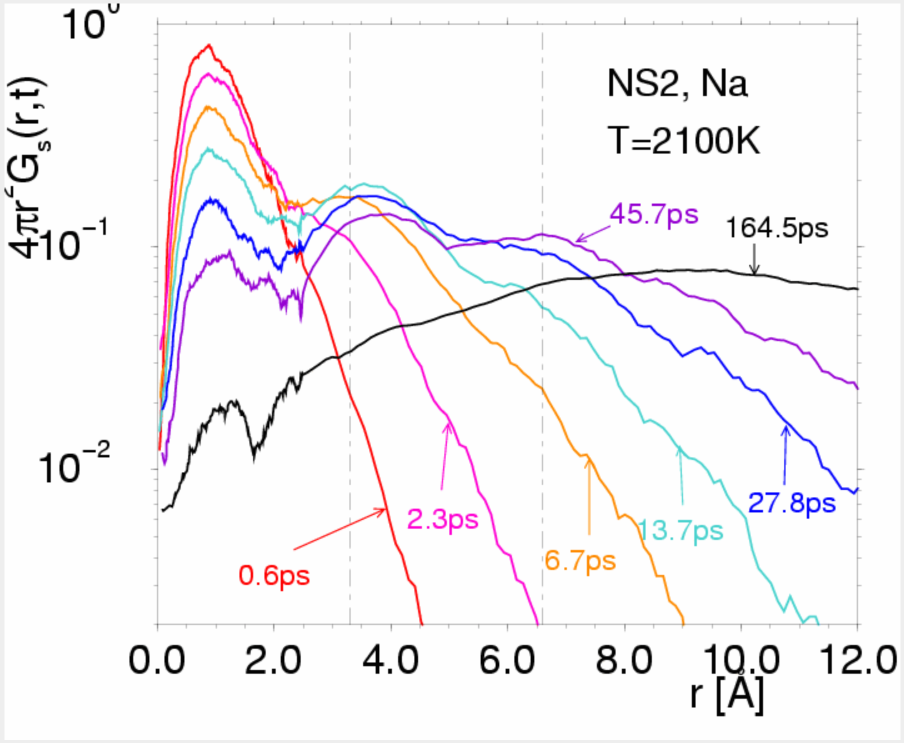
⇒ One has to be careful when one compares such results with experimental data

Dynamics of individual Na atoms in NS

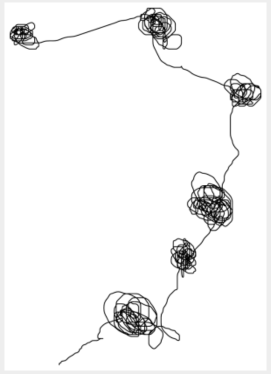
- Characterize the dynamics by means of the self part of the van Hove correlation function $G_s(r,t)$:

$$G_s(r,t) = \frac{1}{N}\sum_{j=1}^{N}\langle\delta\left(\mathbf{r} - (\mathbf{r}_j(t) - \mathbf{r}_j(0)\right)\rangle$$

- N.B.: $G_s(r,t)$ is just the Fourier transform of $F_s(q,t)$



- low T: rattling and hopping motion on the length scale of nearest neighbors ≈ 3.4Å



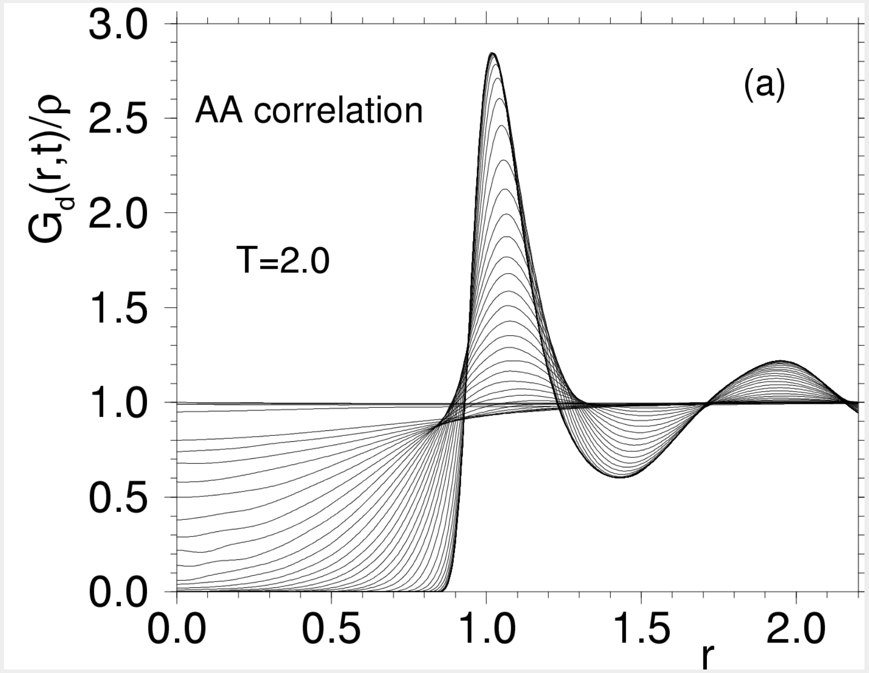
- dynamics of Si and O show only a very weak signature of hopping

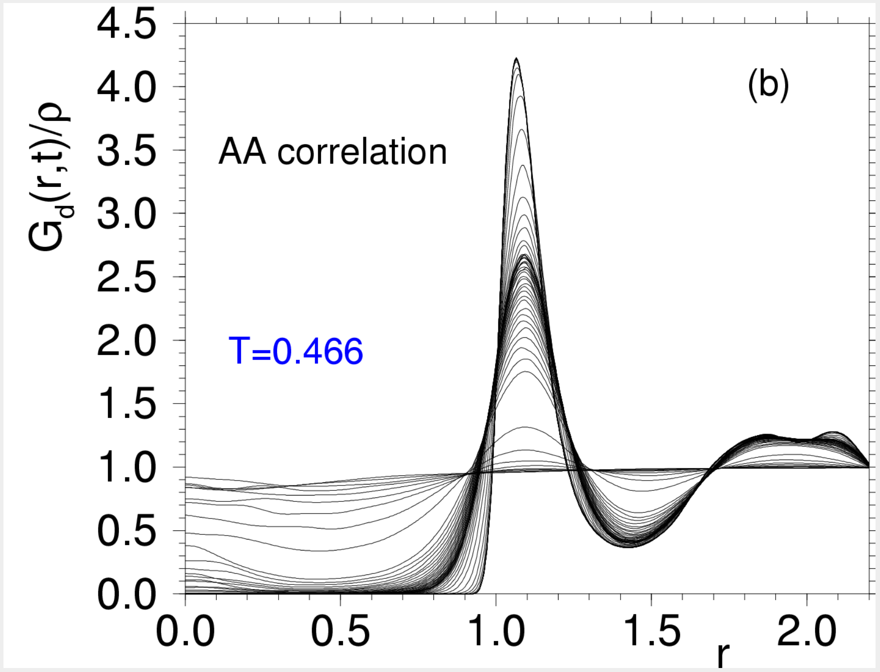
The van Hove correlation function (distinct part)

- Distinct part of van Hove correlation function $G_d(r,t)$ ∝ probability to find at time t a different particle at a distance r from a place at which at time t=0 there was a particle:

$$G_d(r,t) = N^{-1} \sum_i \sum_{j\neq i} \left\langle \delta(r - |\mathbf{r}_i(t)-\mathbf{r}_j(0)|) \right\rangle \quad \text{N.B. } G_d(r,0) = g(r)$$

Distinct van Hove correlation function for A particles





high T: correlation hole at r=0 is quickly filled up

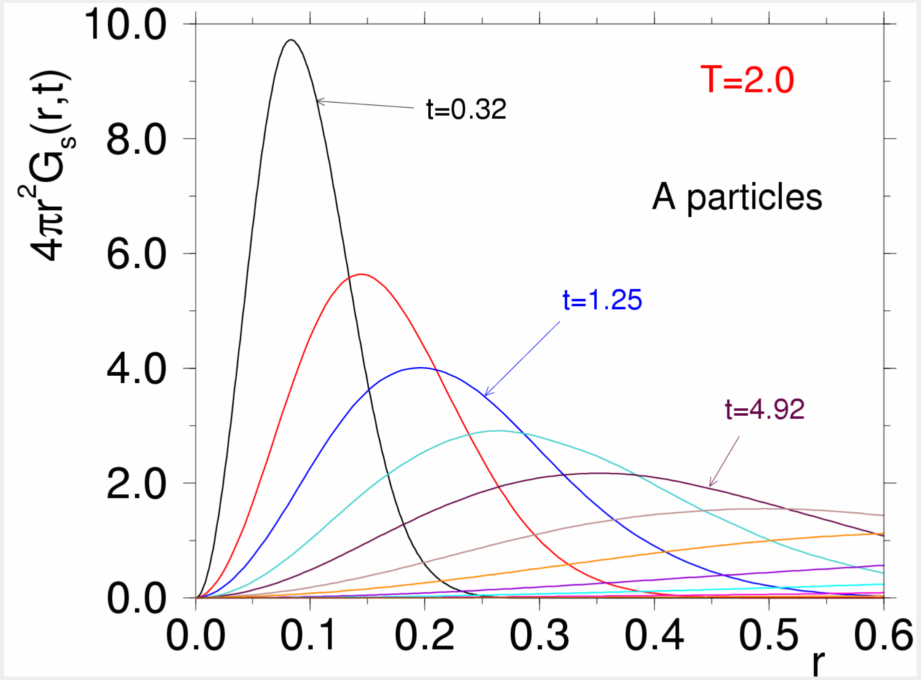
low T: correlation hole at r = 0 survives for a long time (note small peak at r=0!)

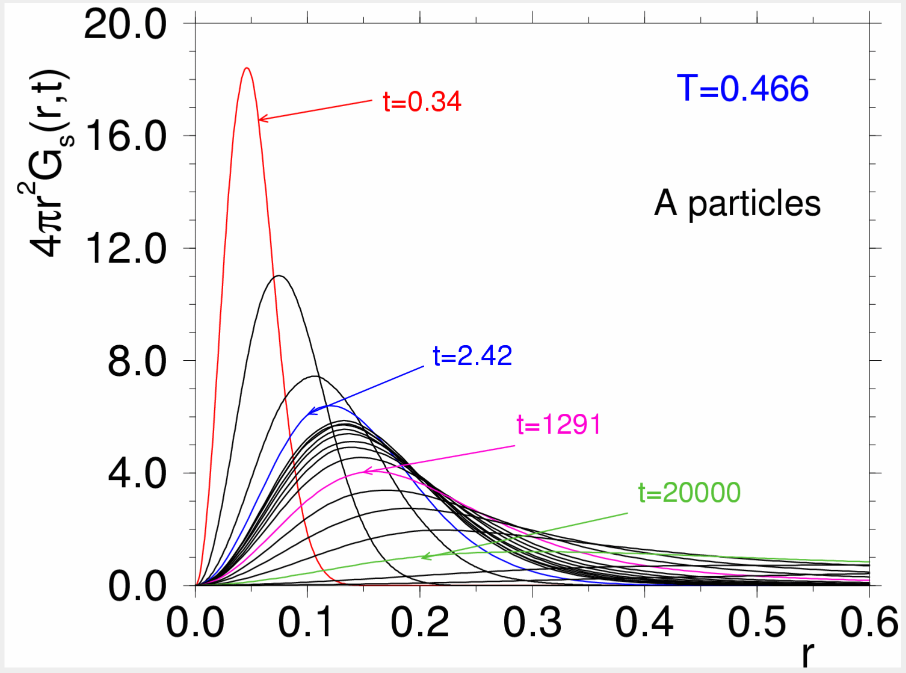
The van Hove correlation function (self part)

- Self part of van Hove correlation function $G_s(r,t)$ = probability that a particle has moved a distance r in time t:

$$G_s(r,t) = N^{-1} \sum_i \left\langle \delta(r - |\mathbf{r}_i(t) - \mathbf{r}_i(0)|) \right\rangle$$

self van Hove correlation function for A particles





high T: no cage effect

low T: cage effect
N.B. cage is quite small!
(Lindemann criterium of melting)

Structure

Define local particle density

$$\rho(\mathbf{r}) = \sum_{i=1}^{N} \delta(\mathbf{r} - \mathbf{r}_i)$$

$\Rightarrow$ Mean density ρ is given by

$$\langle \rho(\mathbf{r}) \rangle = \frac{N}{V}$$

Define a two point density correlation in space:

$$\begin{aligned} G(\mathbf{r}', \mathbf{r}'') &:= \langle [\rho(\mathbf{r}') - \rho][\rho(\mathbf{r}'') - \rho] \rangle \\ &= \langle \rho(\mathbf{r}')\rho(\mathbf{r}'') \rangle - \rho^2 \end{aligned}$$

N.B.: One subtracts the average density ρ, i.e. we look at fluctuations

$$G(\mathbf{r}', \mathbf{r}'') = \sum_{\mathbf{i}} \sum_{\mathbf{j}} \langle \delta(\mathbf{r}' - \mathbf{r_i}) \delta(\mathbf{r}'' - \mathbf{r_j}) \rangle - \rho^2$$

Structure: Coordination number

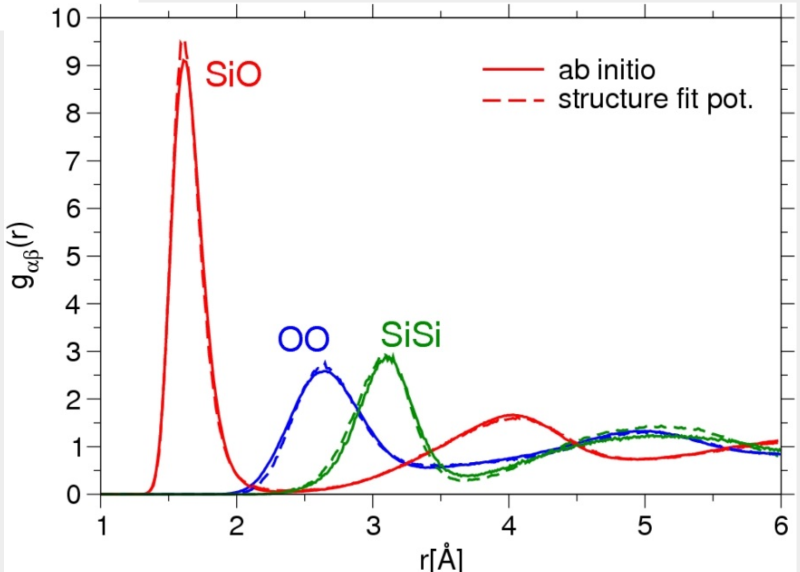
- Coordination number: Average number of nearest neighbor particle of a given particle; Ex.: cubic lattice: Z=6

- The integral $\int_0^{2\pi} d\phi \int_0^{\pi} d\theta \int_0^{R} dr g(\mathbf{r})$ gives the number of particles

that are within distance R of a given particle

- If g(r) has a well defined first minimum at $R_{\min}$, we can define the coordination number Z:

$$Z = \int_0^{2\pi} d\phi \int_0^{\pi} d\theta \int_0^{R_{\min}} dr g(\mathbf{r}) = \int_0^{R_{\min}} 4\pi g(r) dr$$



MD: Constraints

In molecular systems the intra-molecular interactions are rather strong (e.g. CH_4, polymers with H,...) ⇒ rapid vibrations ⇒ h has to be very small ($h^2 f(t) \ll \sigma$) ⇒ we loose time by simulating something we don't care about (vibrations) instead of moving the molecules.

⇒ Solution: We consider the intra-molecular bonds as being completely frozen, i.e. rigid ⇒ constraints (distances, angles,...). The Verlet/velocity Verlet algorithm does not take into account such constraints. But these algorithms can be generalized so that constraints are considered ⇒ algorithms SHAKE/RATTLE ⇒ also complex molecules can be simulated efficiently.

Other solutions:
i)Introduce effective potentials that mimic, e.g., the hydrogen molecules ⇒ embedded atom potentials
ii)Methods using multiple time scales.