Basics of molecular dynamics

- The basic idea of molecular dynamics (MD) simulations is to calculate how a system of particles evolves in time.

- A little bit of history: why “molecular” ?

  - The method was first used with a realistic potential by Alder and Wainwright in 1959 to calculate the vibrations of atoms in molecules. The name stuck, unfortunately.

- There is an interesting parallel to classical mechanics here. The two-body motion problem was solved by Newton way back then. The three-body problem was solved by a Finnish guy, Sundman, in the early part of the last (20th) century - but the solution is utterly impractical.

- The N-body problem, N>3, can not be solved analytically. MD can also described to be a numerical way of solving the N-body problem. The solution is of course never exact, but if done properly it can be done arbitrarily accurately.

- Consider a set of atoms at positions $\boldsymbol{r}_i$ and some interaction model which gives us the potential energy of the system $V(\boldsymbol{r}_i)$

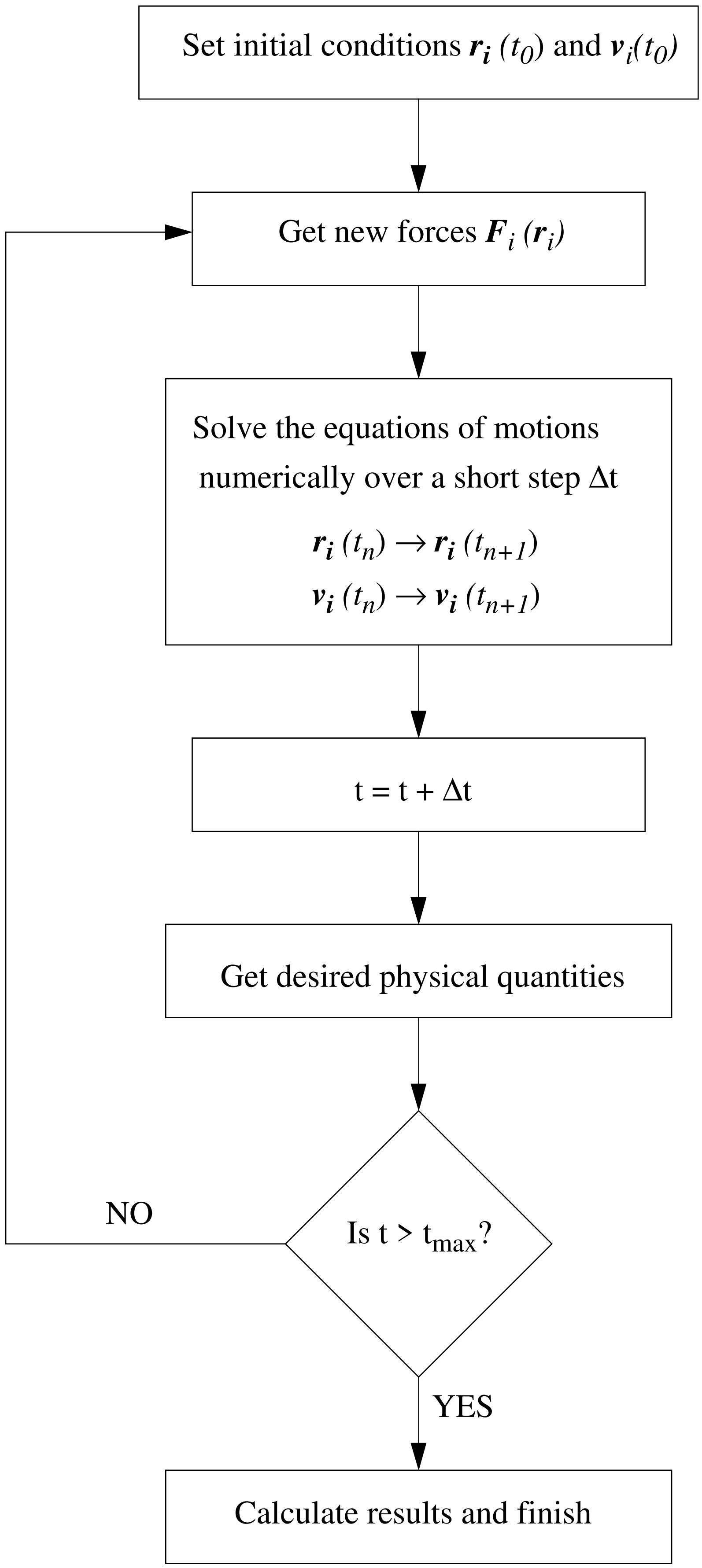
- In Newtonian mechanics we then get:

$$\frac{d\boldsymbol{r}_i}{dt} = \boldsymbol{v}_i$$

$$\frac{d}{dt}(m_i\boldsymbol{v}_i) = \boldsymbol{F}_i = -\nabla V = -\nabla\left(\sum_j V_2(\boldsymbol{r}_i, \boldsymbol{r}_j) + \sum_{j,k} V_3(r_i, r_j, r_k) + \dots\right)$$

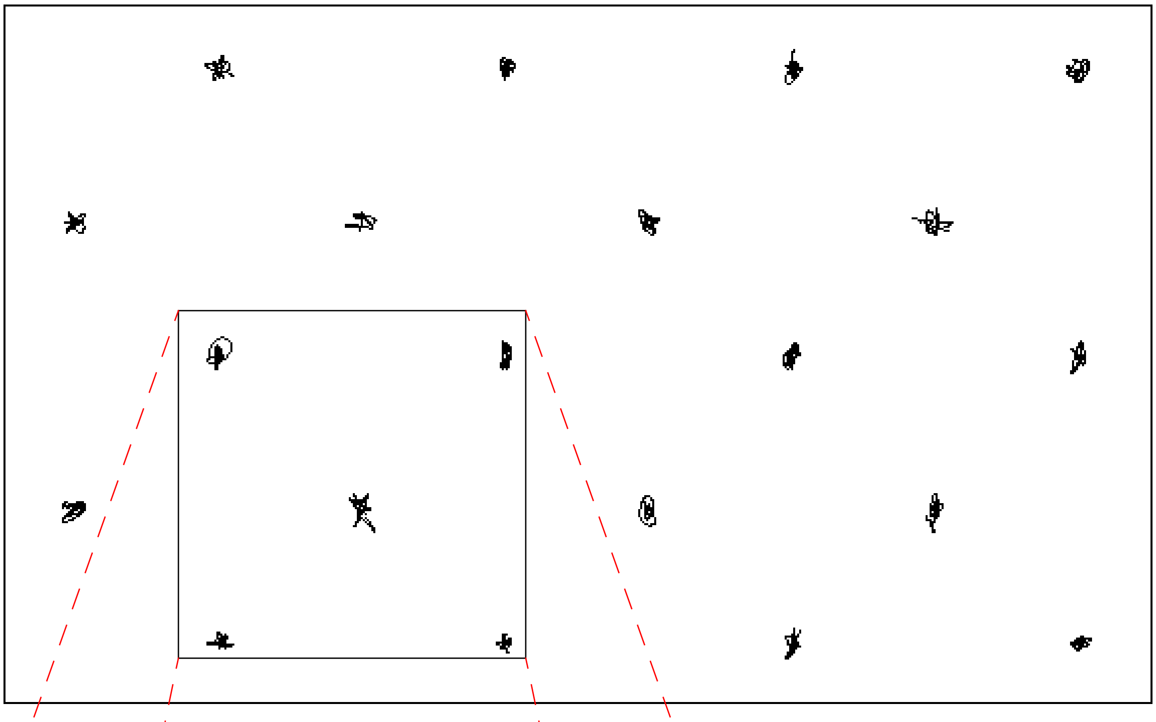
- By solving the above set of equations numerically we can derive $\boldsymbol{dr}$ over some short time interval dt

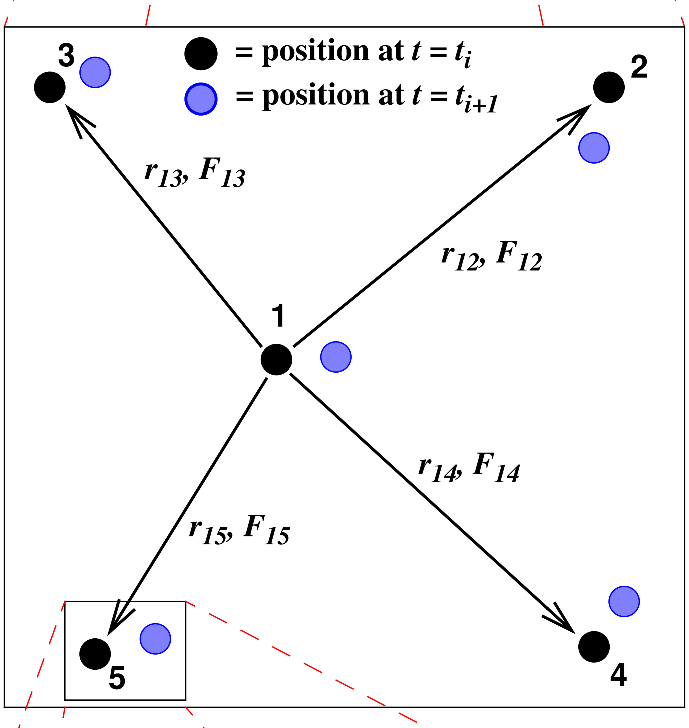
Basic MD algorithm (slightly simplified)



An alternative view

MD-simulation of thermal motion over 100 time steps

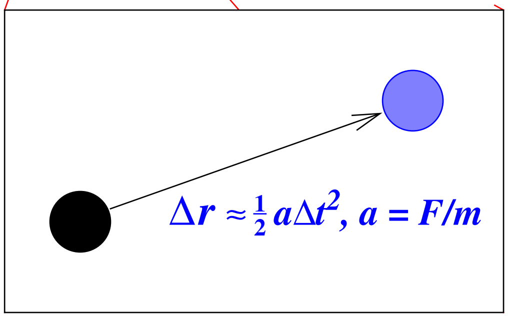




Zoom in on 2 time steps:

At time t the distances r_{ij} and hence forces F_{ij} between nearby atoms are calculated

From these forces we can solve the equations of motion, and hence get new positions and velocities.



The displacement over a time step Δt is denoted with Δr. Δr has to be much smaller than the distance between nearby atoms.

General considerations

- The above was the simplest possible example, the so called microcanonical or NVE ensemble. This means that the approach preserves the number of atoms N, the volume of the cell V and the energy E. Other ensembles will be dealt with later on in the course. But the NVE ensemble is the most natural one in that it is the true solution of the N-body problem, and corresponds to the real atom motion.

- The possible size of the system for MD (2003):

  - N ~ 1 - 10^9 atoms

- First MD simulation:

  - B. J. Alder, T. E. Wainwright: *Phase transition for a Hard Sphere-System,* J. Chem. Phys. **27** (1957) 1208

- State-of-the-art (2003):

  - SPaSM (Scalable Parallell Short-range Molecular Dynamics) in Los Alamos: 1 000 000 000 atoms. http://bifrost.lanl.gov/MD/MD.html

  - In Finland: CSC IBM SC: some 10 000 000 atoms with a realistic potential easily possible for thousands of time steps in a 32 processor queue. We are routinely now running 2-4 million atom jobs.

- If all *N* atoms interact with all atoms, one has to in principle calculate N^2 interactions. This is prohibitively expensive for millions of atoms.

- Fortunately in practice most atomic interactions decrease rapidly in strength as r -> ∞. In that case it is enough to calculate only interactions to nearby atoms.

  - E.g. in diamond-structured semiconductors (Si, Ge, GaAs...) atoms have 4 covalent bonds, so the calculation can be reduced to 4 neighbours => 4 N interactions.

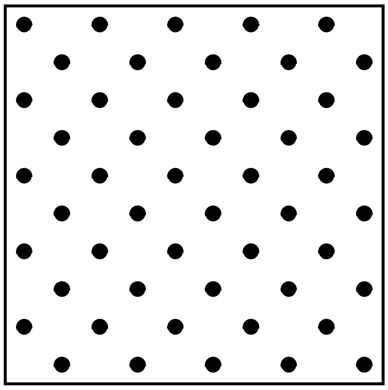
  - In metals atoms more than ~ 5 Å far can usually be neglected => about 80 N interactions

  - In ionic systems the interaction ~ *1/r*, i.e. decreases very slowly. It can not be cut off, but there are smart ways around this, as we shall see later.

Simulation cell

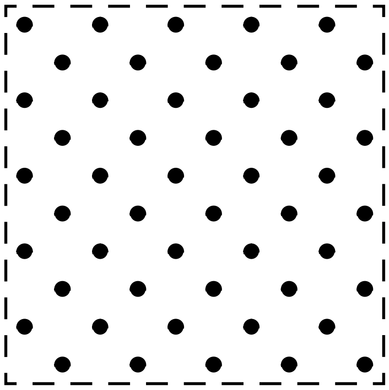
- In practice in most cases the atoms are arranged in a orthogonal **simulation cell** which has a size $S_x \times S_y \times S_z$

  - It is also perfectly possible to use a simulation cell with axes than are not orthogonal, but in practice this is rarely needed or done.



- Problem: what should we do with the atoms at the borders.

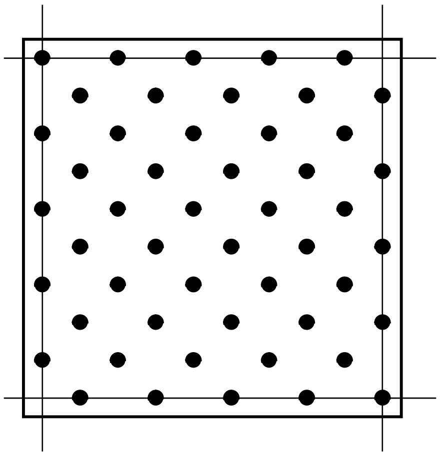
  1. Nothing => “free” boundaries



  - This works fine if we want to deal with e.g. a molecule or nanocluster in vacuum.

  - If we want to describe a continuous medium, this does not work: the atoms are left “hanging” on the surface as if they would be on the surface.

2. Fix the boundary atoms:

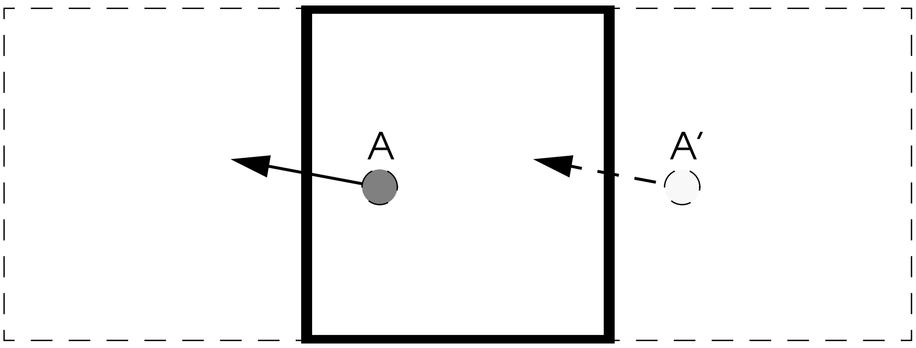


- completely unphysical, this should be avoided if possible. Sometimes it is needed and with a fairly large “sacrificial” region next to the fixed one can be perfectly OK.

3. Periodic boundary conditions

- To implement this very important boundary condition two things have to be done

1. An atom which passes over the cell boundary comes back on the other side:



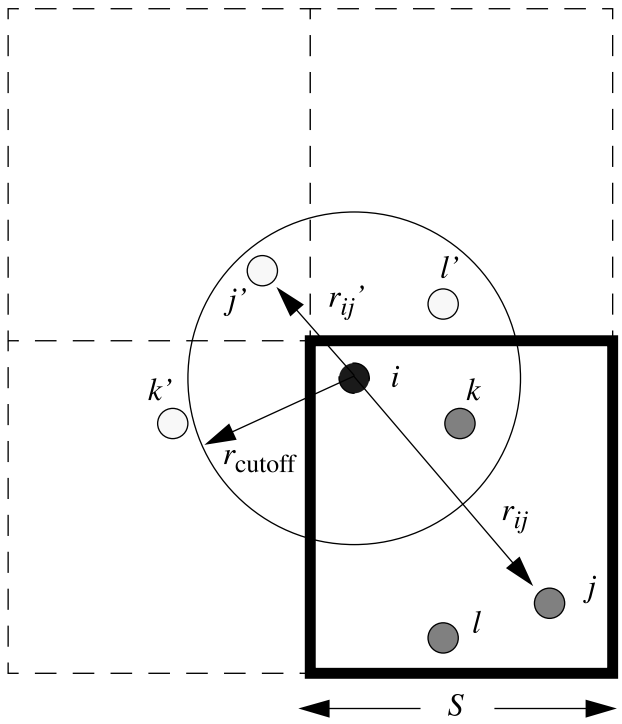
- In practice this can be implemented as follows (Fortran 90):

```
!     x            : particle coordinate
!     periodicx    : = true periodic, false free
!     xsize        : MD cell size

  if (periodicx) then
      if (x <  -xsize/2.0) x = x + xsize
      if (x >=  xsize/2.0) x = x - xsize
  endif

      and the same for y- and z-coordinates.
```

2. When distances between atoms are calculated, the periodic boundaries have to be taken into account:



- The solid box is the simulation cell, with atoms *i*, *j*, *k* and *l*. Because of the periodic boundaries, all atoms have image atoms in the repeated cells, for instance *j'*, *k'*, *l'*.

- When we want to get the distance between atom i and j, which distance should we use?

- Because here $r_{ij} > r_{ij}'$, we use for the distance between atoms *i* and *j* the vector r_{ij}'

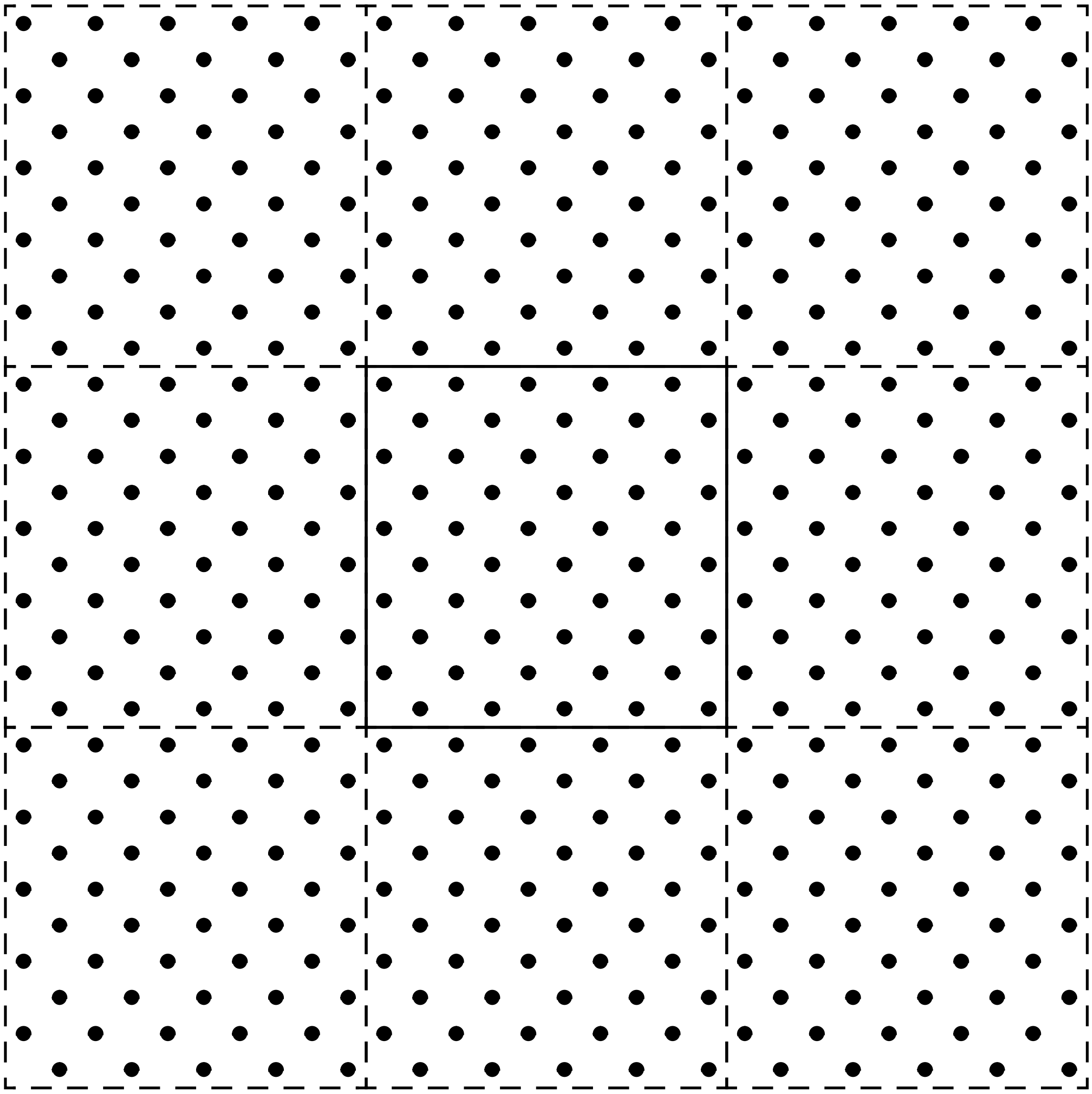
- As a pseudo-algorithm in the *x* dimension:

```
if (periodicx) then
    dx = x(j) - x(i)
    if (dx >   xsize/2.0)  dx = dx - xsize
    if (dx <= -xsize/2.0)  dx = dx + xsize
endif
```

and similarly for y and z

- Note that if the maximum distance by which atoms can interact r_{cutoff} > *xsize/2* the atom *i* should actually interact both with atom *j* and *j'*. To prevent unphysical double interactions we need to have *xsize* > 2 r_{cutoff}

- Thus we get a system where the simulation cell has an infinite number of image cells in all directions, and a model of an infinite system.



- But one still has to be careful with the size of the basic simulation cell: the periodicity brings an artificial interaction over the simulation cell borders which can distort some simulation cell results. For instance, a strain field arising from a point source, which is infinite, will obviously be distorted at the periodic borders.

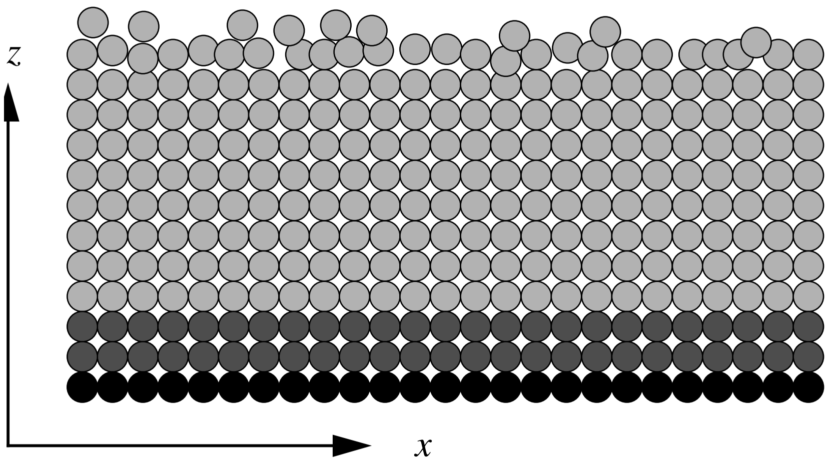
- For instance a single vacancy (one missing atom) in Si: in quantum mechanical calculations at least some 200 atoms are required to get the energy reliably [Puska 1998 Phys. Rev. B]

- And for instance a 5 nm Co cluster in Cu: about 1 000 000 atoms needed to get the strain energy reliably.

- To test this: simulate with different N and monitor the convergence.

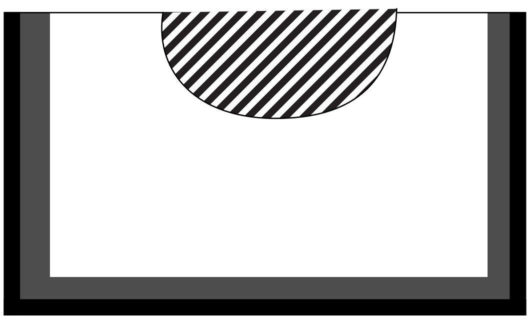
- Simulating surfaces:

  - periodic boundaries only in x- and y-directions
  - free surface:



  - the bottom either:
    a) free: simulation of a free-standing thin foil with two surfaces
    or b) fixed to model a bulk below:
      - Very bottommost atoms fixed:
      - A few atom layers above fixed layers damped with e.g. a temperature control algorithm

- Simulation of energetic processes:



  - In a simulation where a lot of energy is brought into the MD cell in a local region, the energy has to be scaled out from the system to model a much cooler 'heat bath' in a realistic system.

  - The energetic processes may also introduce a lot of momentum into the cell, which could cause the entire cell to move.

  - Solution: fix all boundary atoms except at the surface, and do T scaling in a few atom layers above these, as above.

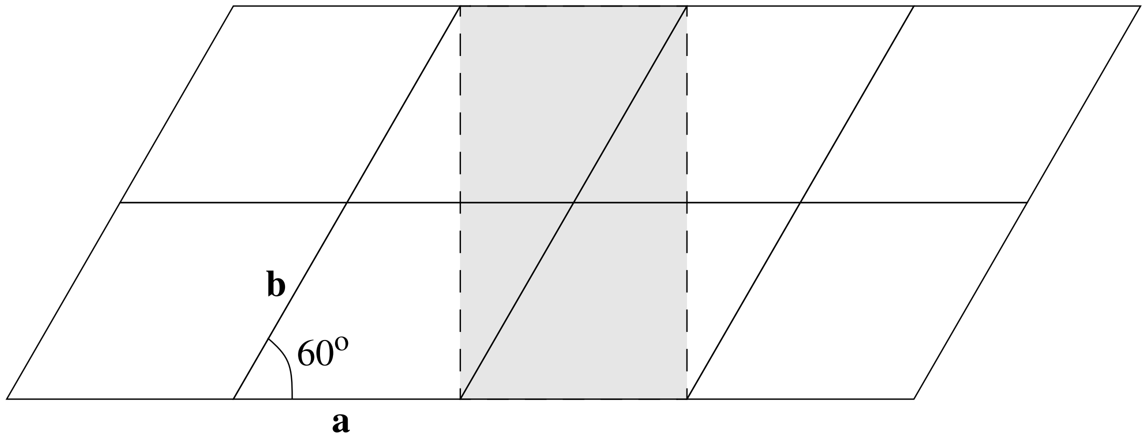
Initial conditions: creating atoms

- For cubic lattices (FCC, BCC, SC, DIA) it is easy to create the lattice. For instance FCC:

```
basis(1,1)=0.0;     basis(1,2)=0.0;     basis(1,3)=0.0;
basis(2,1)=0.5;     basis(2,2)=0.5;     basis(2,3)=0.0;
basis(3,1)=0.5;     basis(3,2)=0.0;     basis(3,3)=0.5;
basis(4,1)=0.0;     basis(4,2)=0.5;     basis(4,3)=0.5;
offset(1)=0.25;     offset(2)=0.25;     offset(3)=0.25;
nbasis=4;

n=1;
do i=0,nx-1
   do j=0,ny-1
      do k=0,nz-1
         do l=1,nbasis
            x(n)=-xsize/2+(i+offset(1)+basis(l,1))*a
            y(n)=-ysize/2+(j+offset(2)+basis(l,2))*a
            z(n)=-zsize/2+(k+offset(3)+basis(l,3))*a
            n=n+1
         enddo
       enddo
    enddo
enddo
```

- The HCP lattice is also very common, but not orthogonal in the conventional representation: the angle between the **a** and **b** axes is 60^{o} (or 120^{o} in an alternative notation).



- Because in the HCP structure **a** = **b**, and because cos $60^{o} = 1/2$, the HCP lattice can be transformed into an equivalent orthogonal representation in the way illustrated in the figure. Now the new unit cell (grayscale) corresponds to two of the conventional HCP unit cells.

Initial atom velocities

- How do we set the cell temperature to a desired value?

  - We have to generate initial atom velocities which correspond to the Maxwell-Boltzmann distribution (which is surprisingly well valid even in crystals):

$$\rho(v_{i\alpha}) = \left(\frac{m_i}{2\pi k_B T}\right)^{1/2} \exp\left(-\frac{1}{2} m_i v_{i\alpha}^2 / k_B T\right); \quad \alpha = x, y, z.$$

  - This is just a Gaussian function with suitable scaling

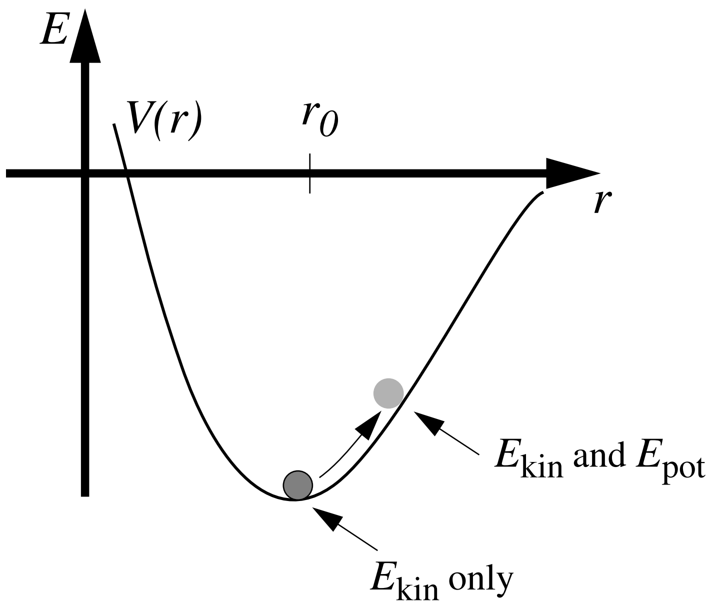
  - We usually also want to set the total momentum of the cell to zero to prevent the entire cell from starting to move:

$$\mathbf{P} = \sum_{i=1}^{N} m_i \mathbf{v}_i = 0.$$

- So in practice all this can be achieved with:

```
vrms=sqrt(kB*2*T/(m*u))/vunit
do i=1,n
      vx(i)=vrms*gaussrandom(iseed)
      vy(i)=vrms*gaussrandom(iseed)
      vz(i)=vrms*gaussrandom(iseed)
      vxtot=vxtot+vx(i)    ! If all atoms have the same mass,
      vytot=vytot+vy(i)    ! it is enough to scale the total v
      vztot=vztot+vz(i)    ! to zero
enddo
vxtot=vxtot/n
vytot=vytot/n
vztot=vztot/n
do i=1,n
      vx(i)=vx(i)-vxtot
      vy(i)=vy(i)-vytot
      vz(i)=vz(i)-vztot
enddo
```

- Note the factor of 2: if the simulation is started from perfect lattice sites, or bound equilibrium positions in a molecule, half of the initial kinetic energy will be changed to potential energy after a while due to the equipartition theorem of energy:



- It is also possible to get realistic initial random displacements

  - This can be derived from the Debye model: the thermal displacement in the direction of the axis i is a Gaussian distribution of the form

$$w(\delta_i, T) = [2\pi\sigma]^{-1/2} e^{-\delta_i^2/2\sigma^2}$$

where

$$\sigma = \frac{20.89}{\sqrt{3}\Theta_D}\sqrt{\frac{T}{A}}\text{Å} \text{ , where } 20.89 = \sqrt{\frac{9\hbar^2}{k_B u}}$$

Θ_D = Debye temperature of the material

A = Atomic mass

  - The initial position can now be obtained with Gaussian-distributed random numbers as above.

  - Note, however, that this does not account for quantum mechanical zero-point vibrations which give additional displacements near 0 K.

Generating random numbers

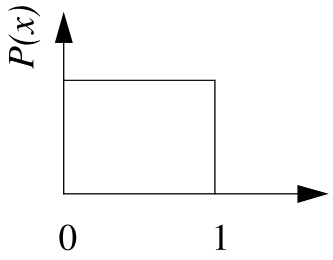
(This topic is dealt with in much more detail on the MC simulation course)

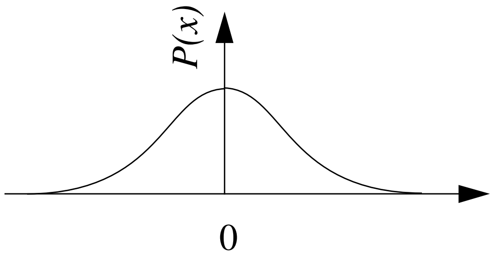
- Almost all kinds of simuations in physics use random numbers somewhere. As we saw above, MD simulations need them at least for initial velocity generation.

- Computer-generated random numbers are of course not truly random, but if they have been generated with a good algorithm, they start to repeat each other only after a very large (e.g. 10^{20}) number of iterations. If the number of random numbers used in the entire simulation is much less than the repeat number, the algorithm *probably* is good enough for the application.

- Random numbers can be generated for different distributions. This means that if we generate a large number of numbers and make statistics out of them, they will eventually approach some distribution.

  The most common is of course an even distribution in an interval, another very common is Gaussian-distributed numbers:





Evenly distributed random numbers:

- Many programming languages offer their own random number generator (e.g. in ANSI-C rand()). A good rule-of-thumb regarding these is:

  Never use them for anything serious !

  - The reason is simply that the language standard only specifies that the generator has to be there, not that it works sensibly. Since there are no guarantees it does (there are famous examples of the opposite) it should not be used.

- Most random number generators are based on modulo-arithmetics (the calculation of remainders in a division) and iteration. In the simplest possible form:

$$I_{j+1} = aI_j \pmod m$$

  - Park and Miller 'minimal standard'-generaatot: a=16807, m = 2^{31}-1

  - In the beginning the number I_0 i.e. the **seed number** is chosen randomly.

  - One practical implementation (Fortran90):

```
real function uniformrand(seed)
  implicit none
  integer :: seed,IA,IM,IQ,IR,MASK
  real :: ran0,AM
  integer :: k

  parameter (IA=16807,IM=2147483647,AM=1.0/IM, &
    IQ=127773,IR=2836,MASK=123459876)

  seed=ieor(seed,MASK)
  k=seed/IQ
  seed=IA*(seed-k*IQ)-IR*k
  if (seed < 0) seed=seed+IM
  uniformrand=AM*seed
  seed=ieor(seed,MASK)

  return

end function uniformrand
```

- The repeat interval for this routine ~ $2.1 \cdot 10^9$

- This routine is easily good enough if for instance it is only needed for the choice of random numbers in the beginning of an MD simulation. But for instance in a long Monte Carlo integration where random numbers are used all the time, the repeat interval may be reached after which continued running will not improve on the results (and for instance the error estimate of the result will be calculated outright wrong).

- More on this topic for instance from the book: Press, Teukolsky, Vetterling, Flannery: **Numerical Recipes** in C/Fortran, 2nd. ed., chapter 7.

  - The book can be found in its entirety in http://www.std.com/nr/

- But see also:

"Why not use Numerical Recipes?", http://math.jpl.nasa.gov/nr/

and the reply to this: http://www.nr.com/bug-rebutt.html

Gaussian-distributed random numbers

- To generate Gaussian random velocities we need to be able to generate Gaussian-distributed random numbers.
- How to do this is dealt with in great detail in my MC course notes, part 4, and also in Numerical Recipes ch. 7.2. Here I only present the most efficient accurate algorithm for this:

1° Obtain two evenly distributed random numbers v_1 and v_2 between -1 and 1, then calculate $w = v_1^2 + v_2^2$

2° If w >= 1 return to step 1°

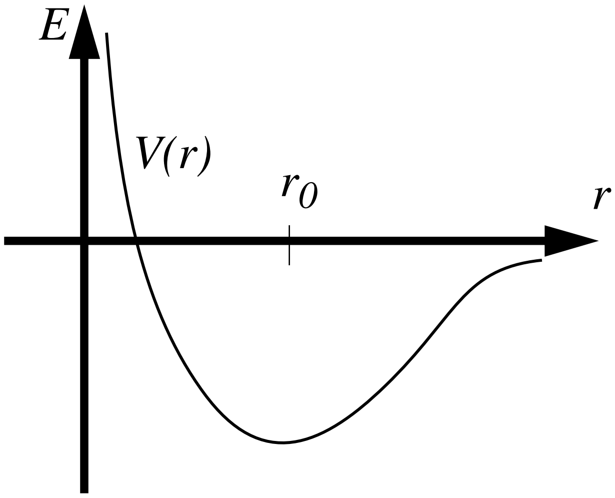
3° Calculate $r = \sqrt{-2\log w}$

4° Calculate $x = rv_1/\sqrt{w}$ and $y = rv_2/\sqrt{w}$

5° Return x and on next step y

Choosing the MD time step

- Depends on the algorithm used, but not too strongly.



- The change in the atom position in the potential used should not be too strong.

- A practical, rough rule-of-thumb: the atoms should not move more than 1/20 of the nearest-neighbour distance.

- Thermal velocity of atoms (Maxwell-Boltzmann distribution):

$$E_{rms} = \frac{3}{2}k_B T = \frac{1}{2}mv^2 \Rightarrow v_{rms} = \sqrt{\frac{3kT}{m}}$$

  - But the distribution continues much beyond this.

  - Rough estimate of the time step needed: 300 K Cu (m=63.55 u):

  - 5 v_{rms} = 0.017 Å/fs.

  - Nearest-neighbour distance 2.55 Å => Δt = (2.55/20)/0.017 = 7 fs.

- In practice for stability Δt <= 4 fs.

- In pure MD there is no way to increase the time step above ~ 10 fs in atom systems in almost any real material at ordinary temperatures (77 K and up). This is the main limitation of MD. Of we would want to simulate a process which, say, takes 1 s, we would need at least 10^{14} time steps!

- This gives an easy way to estimate the order-of-magnitude of the upper limit for the time scale MD can handle in a given time:

  - Most realistic classical MD interatomic potentials require at least of the order of 100 flops/atom/time step.

  - Say our time step is 1 fs, and we want to simulate a 10000 atom system.

  - Hence we need 10^6 flops/time step. To get to 1 ns = 10^9 fs we would need 10^{15} flops. On a typical (AD 2003) 1 Gflop/s processor, the simulation would thus require $10^{15}/10^9$ seconds = 10^6 s i.e. about 11 days. To get to 1 μs would require some 30 years on this processor.

- Hence we see that ordinary MD is restricted to <~ 100 ns processes in practical use.

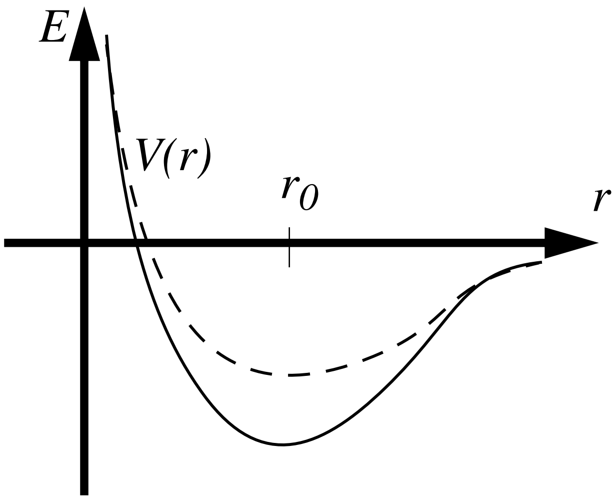
Is it possible to speed up MD ?

- Yes, sometimes. This can be achieved at least in some cases where we are interested in transitions induced by thermally activated processes, i.e. processes which follow a behaviour of the type

$$\nu = \nu_0 \, e^{E_A / k_B T}$$

  where υ is the rate of the process occurring.

- Art Voter has presented so called **HyperDynamics** [A. F. Voter, J. Chem. Phys. **106** (1997) 4665 ja Phys. Rev. Lett. 78 (1997) 3908]. It can in some cases speed up MD by a factor of the order of 100-1000, in others not at all.



  - In this method, Δt does not really increase, but the potential well is made shallower so that the probability of processes with a large activation energy increases. The error which is thus formed is compensated by transition state theory (which is beyond the scope of this course).

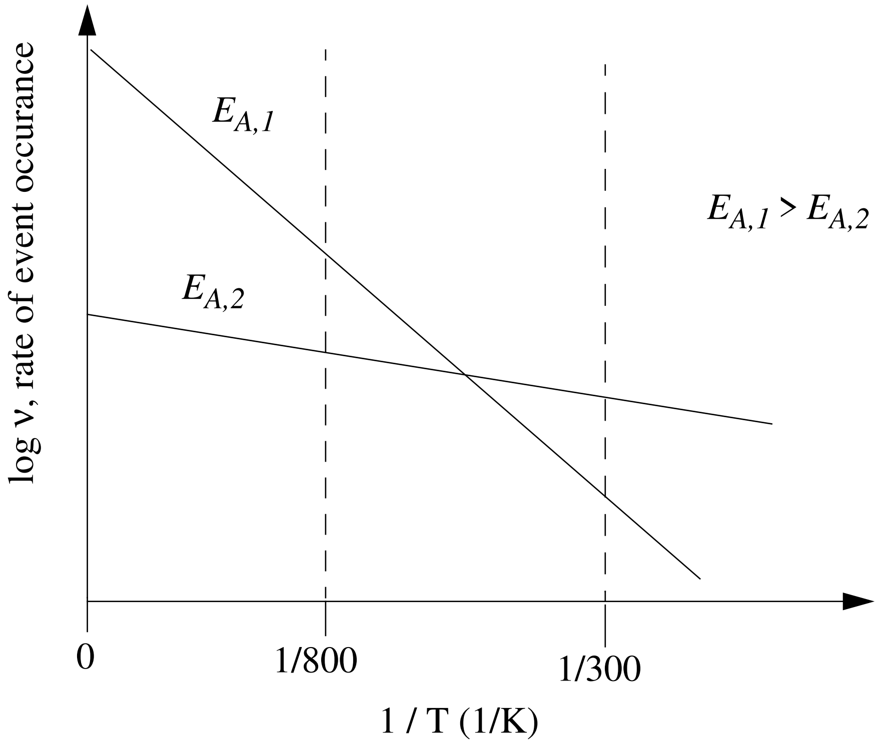
  - The method is well suited for cases where we have to overcome a high potential energy barrier in an ordered system, e.g. vacancy and adatom difusion. But if the energy barrier is low (e.g. interstitial migration in metals) or if we have numerous local energy minima close to each other, like in most amorphous and liquid systems, the method is useless [Art Voter, private communication].

Temperature accelerated dynamics (TAD)

- There is of course always is the Arrhenius extrapolation method: if we know that in our system there is only one single activated process occurring, and nothing else, we can simulate at higher T and then extrapolate the Arrhenius-like exponential $\exp(-E_A/k_BT)$ to lower T to know the rate or time scale at lower T.

- A smart extension to Arrhenius extrapolation is Art Voter's TAD method [e.g. Sorensen, Phys. Rev. B 62 (2000) 3658; a review of Voters methods is given in Annu. Rev. Mater. Res. 32 (2002) 321]

- To understand the idea in this, let us consider a system with exactly 2 activation energies (this is just a tutorial example, the method works in principle for any number of activation energies). We want to simulate what the system does at 300 K, but the processes are so slow nothing happens there. So we will use a higher T, say 800 K. Let us then assume that the Arrhenius plot of the system looks as follows:



• Now when we simulate at 800 K, event type 1 will occur much more frequently than type 2. But we want to know the behaviour at 300 K, so this is wrong. The idea in TAD is to recognize *every* transition that occurs, determine its activation energy, and then leave out the events that would not occur at the lower T. In our example, this means that (almost) only events of type 2 would be accepted.

- In principle this is an excellent idea, but in practice one needs thousands of force evaluations to recognize a transition barrier. Hence the difference between the rates of occurrance needs to be very large for a significant gain to be achieved. But the gain can be huge, up to infinite.

- Like hyperdynamics, if there are lots of shallow minima TAD tends to get stuck and never really gets anywhere.

  - In my personal TAD tests, in some 9/10 test cases no significant gain was achieved, but in 1 case I was able to do a TAD simulation to a time scale of 800 seconds, i.e. about 13 minutes.

- Art Voter himself prefers TAD over hyperdynamics now, and is developing it rapidly towards wider applicability, so it will be interesting to follow the progress.

- There are several other recent ideas to "speed up MD" (for instance M. Kaukonen, PRB **61** (2000) 961, discrete grid MD), but it remains to be seen how useful all of these really are

Time step selection

- In ordinary equilibrium MD Δt usually constant throughout simulation

- But if the maximum velocity of atoms changes a lot during the simulation, it is best to use a variable time step, which increases as the maximum velocity decreases.

  - Simulations of energetic processes:

$$\Delta t_{n+1} = min\left(\frac{k_t}{v_{max}}, \frac{E_t}{F_{max}v_{max}}, c_{\Delta t}\Delta t_n, t_{max}\right)$$

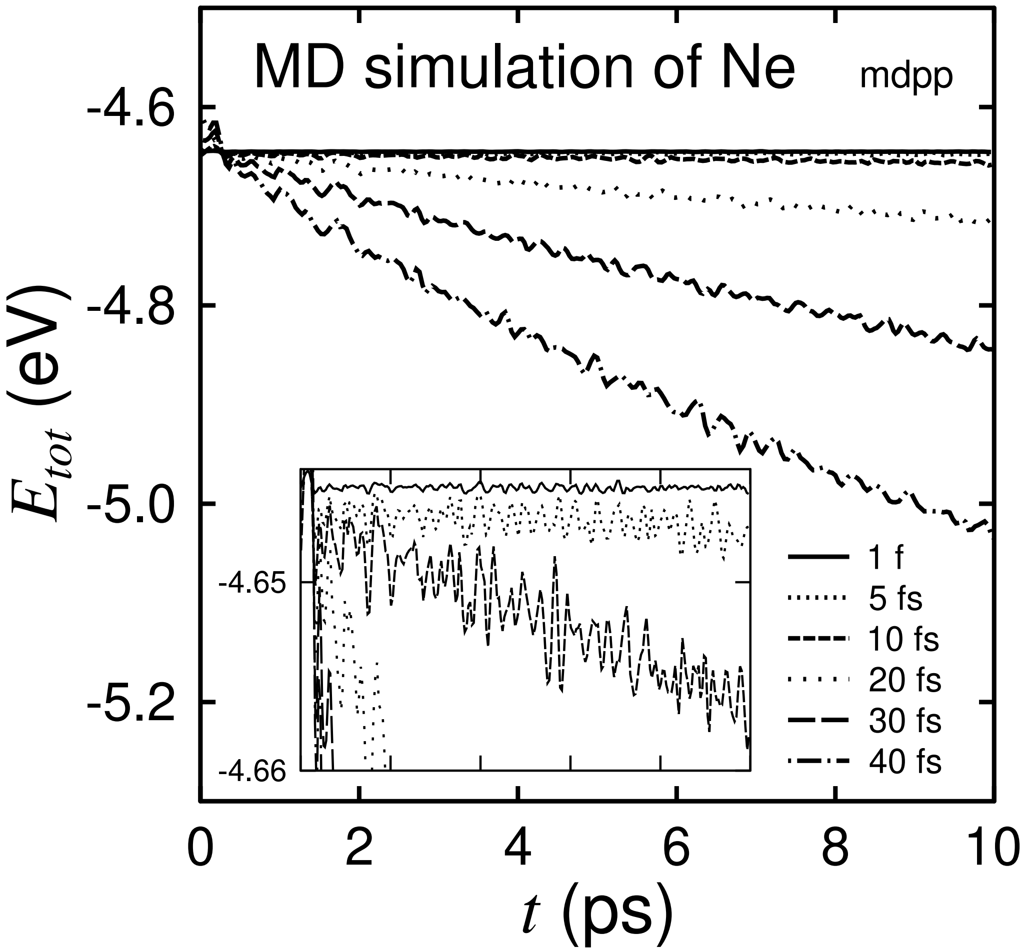
k_t : maximum movement distance/time step (e.g. 0.1Å)
E_t : maximum allowed energy change/time step (e.g. 300 eV)
$c_{\Delta t}$:precents too large sudden changes (e.g. $c_{\Delta t} = 1.1$)
v_{max} : maximum atom speed in system
F_{max} : maximum force on any atom in system
t_{max} : time step once heat bath T has been reached

  - the example values above have been found to work well for binary collisions up to 1 GeV in many materials.

- What happens if the MD time step is too long ?

  - The energy is not conserved.

  - For instance solid neon (FCC lattice, $a = 4.43\text{Å}$) at 30 K:



- Hence the real criterion for selecting the time step becomes energy conservation: for every:

  - new kind of system
  - new kind of process simulated
  - new material
  - new interaction potential

  one needs to check that energy is conserved ‘well enough’ by some test simulations, before starting the real production runs.