MD Simulation

Grundsätzlicher Aufbau:

[Smit, Frenkel,
Understanding Molecular Simulation]

```
program md                      simple MD program

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

Initialisierung

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

fs = Skalierung
damit
T(t=0) = temp

[Smit, Frenkel, Understanding Molecular Simulation]

Kräfte

period. Randbed.,
nint() = floor(x+0.5)

Kräfte für
r>rc2
=0 setzen

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

[Smit, Frenkel, Understanding Molecular Simulation]

Verlet-Schritt

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

Messung
Temperatur T(t),
Energie E(t)

[Smit, Frenkel, Understanding Molecular Simulation]

Zeitmittel

1) Äquilibrierung:
   Betrachte Zeitabhängigkeit von einigen Messgrößen: Energie E(t), Temperatur T(t), …
   Es sollte erkennbar sein, wann diese Größen ihre stationären Werte (+ statistische Fluktuationen) annehmen.
   Kann lange dauern bei großen Systemen

2) Danach beginnt die eigentliche Messung:
   Mittele zu messende Größen
   über viele aufeinanderfolgende Zeitschritte bis der relative Fehler klein ist.
   Bei Messungen von Verteilungen/Histogrammen (z.B. g(r)) braucht man dafür länger als bei globalen Größen wie Energie, Temperatur.

Paar-verteilung g(r)

nhis = N_H = Bin-Anzahl
delg = Δr = Bingröße
box = L

ngr = Zahl der Messungen
für Zeitmittelung

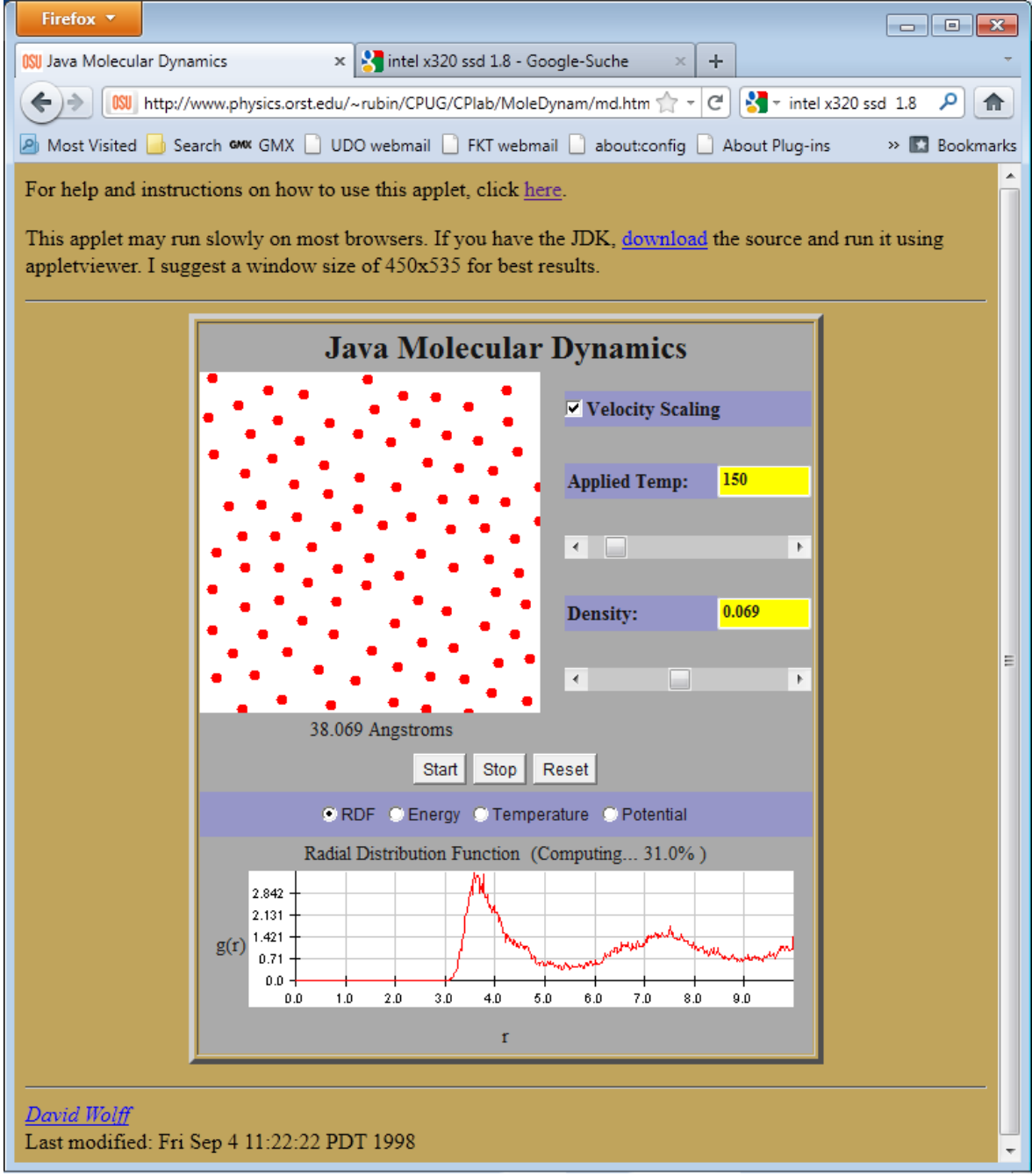
npart = N
rho = ρ = N/V
vb = ΔV

```
subroutine gr(switch)                          radial distribution function
                                               switch = 0 initialization,
                                               = 1 sample, and = 2 results
if (switch.eq.0) then                          initialization
  ngr=0
  delg=box/(2*nhis)                            bin size
  do i=0,nhis                                  nhis total number of bins
     g(i)=0
  enddo
else if (switch.eq.1) then                     sample
  ngr=ngr+1
  do i=1,npart-1
    do j=i+1,npart                             loop over all pairs
      xr=x(i)-x(j)
      xr=xr-box*nint(xr/box)                   periodic boundary conditions
      r=sqrt(xr**2)
      if (r.lt.box/2) then                     only within half the box length
         ig=int(r/delg)
         g(ig)=g(ig)+2                         contribution for particle i and j
      endif
    enddo
  enddo
else if (switch.eq.2) then                     determine g(r)
  do i=1,nhis
    r=delg*(i+0.5)                             distance r
    vb=((i+1)**3-i**3)*delg**3                 volume between bin i+1 and i
    nid=(4/3)*pi*vb*rho                        number of ideal gas part. in vb
    g(i)=g(i)/(ngr*npart*nid)                  normalize g(r)
  enddo
endif
return
end
```

[Smit, Frenkel,
Understanding Molecular Sim

Lennard Jones Flüssigkeit MD Simulation Java Applet

Java applet



[David Wolff,
Rubin Landau
http://www.physics.orst.edu/~rubin/CPUG/CPlab/MoleDynam/md.html