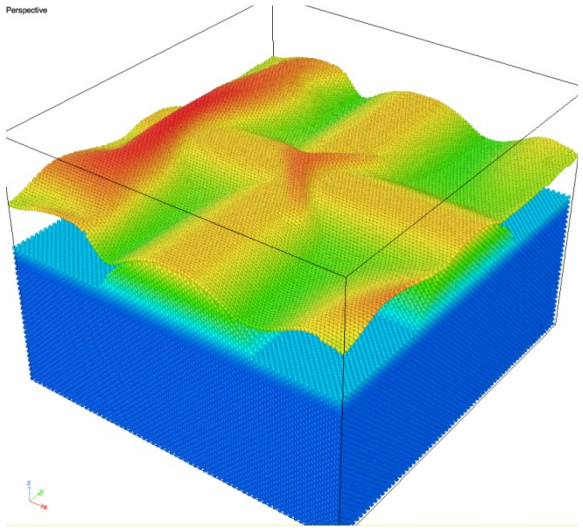
Application of Molecular Dynamics

Computational Material Scence
Lecture 6



Today

- Application of molecular dynamics simulation
- Overview of the package LAMMPS
- Finishing your molecular dynamics code

Goal

The goal of nanoscale simulations is to

1. unravel the atomistic mechanisms that can help understand materials behavior at the nanoscale and
2. assist in designing improved materials for existing and new applications.

It makes little sense to use this simulations to reproduce experimental results

Suggested reading

- Frenkel D and Smit B, 2002, *Understanding Molecular simulations*, Elsevier Academic Press
- Allen MP, Tildesley DJ, 1987, *Computer simulations of liquids*, Oxford University Press
- Haile JM, *Molecular dynamics simulations: elementary methods*, Wiley-Interscience

Applications

(1) Analysis of nanoscale components (actuators, cantilevers)

- Tests of nano-tension and nano-compression
- nanoindentation
- contact, friction
- thin film deposition
- nanowelding
- neutron bombardment
- adsorption of gases or ions

(2) Analysis at the nano-scale of phenomena that affect the macro-scale

- plasticity
- fracture
- phase transformation
- grain boundary sliding
- formation of defects and interaction between defects

Nanoindentation

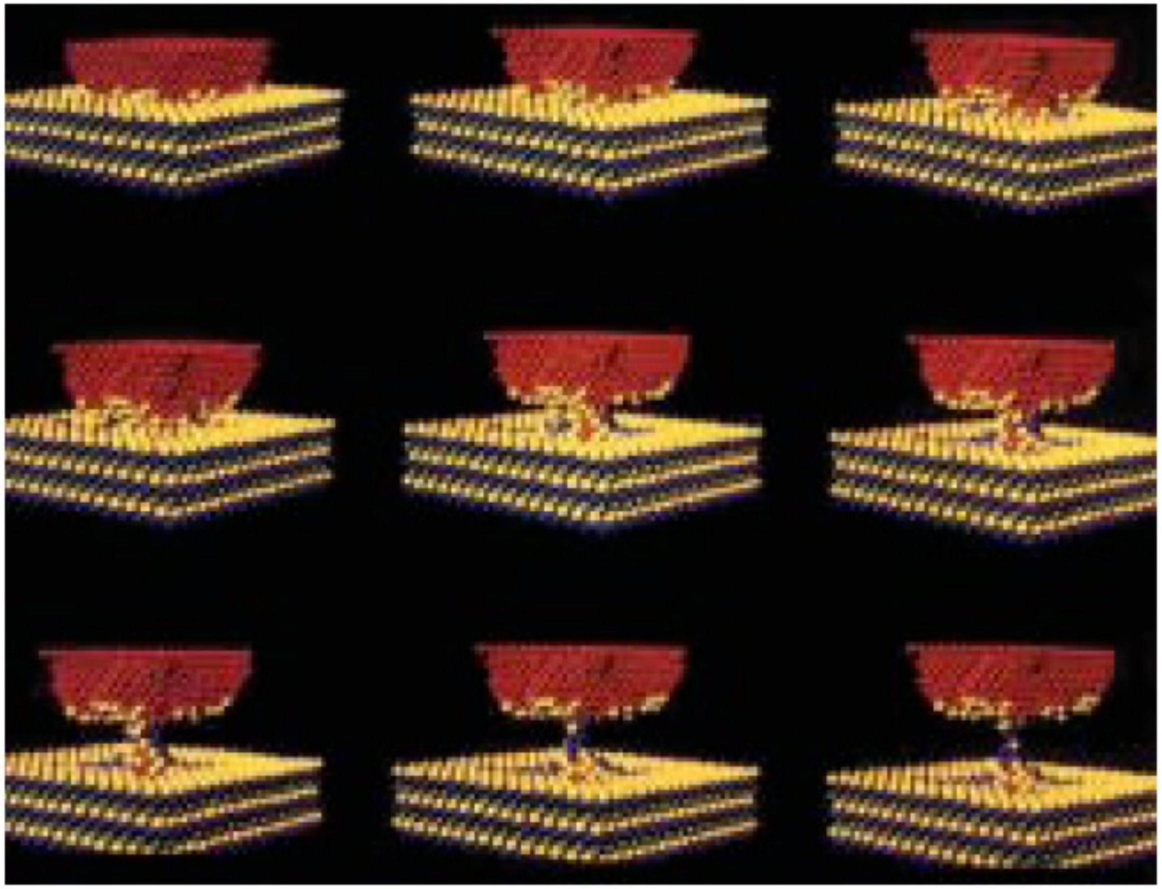


Fig. 2 MD simulation of indentation of solid Au with a Ni indenter. Atomic positions during loading and unloading simulations are shown from top left to bottom right. During unloading a connective neck is formed by Au (yellow) atoms. (Reprinted with permission from[32]*. © 1995 Nature Publishing Group.)*

Load depth curves

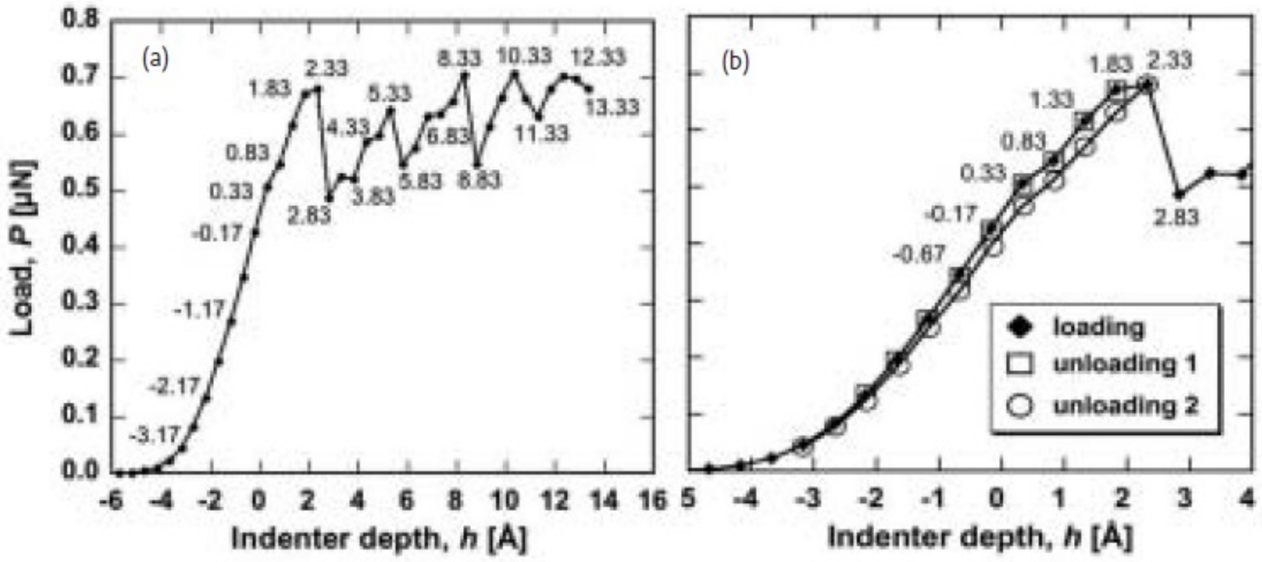
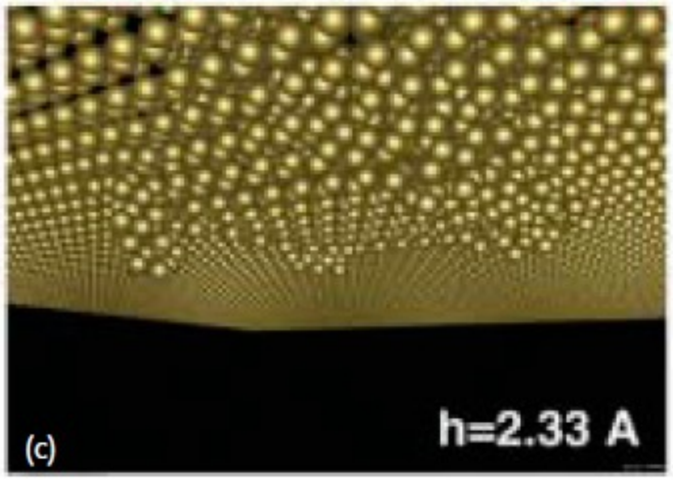
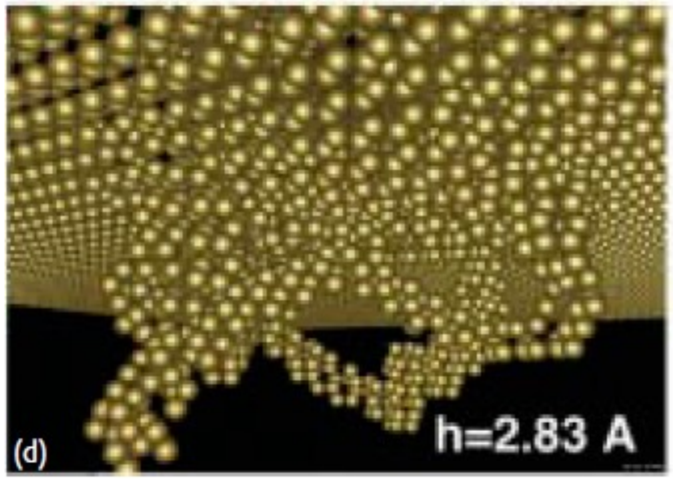


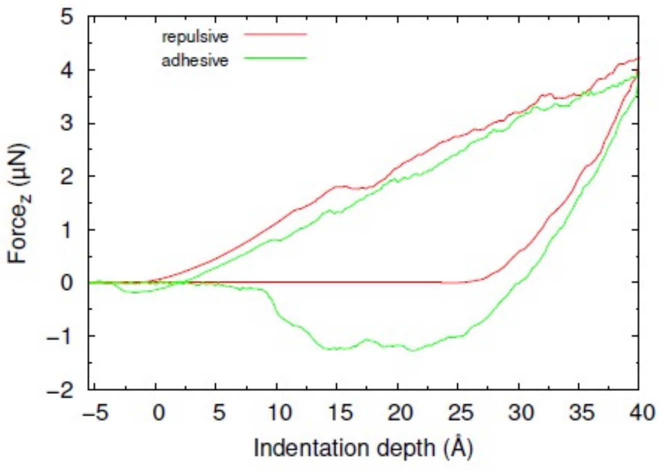
Fig. 1 (a) Force P versus depth h of the indenter obtained in an MD simulation of crystalline SiC. The load drops in the P-h response correspond to discrete plastic events in the indented material. (b) Plot of P against h during the unloading phase where the indenter is pulled out from the sample in 0.5 Å increments. Up to h = 1.83 Å, the deformation is entirely elastic, i.e. unloading from that depth produces a curve (squares) that retraces the loading curve (diamonds). The onset of plastic deformation happens at h = 2.33 Å, reflected in the hysteresis of the second unloading curve (circles). (Reprinted with permission from[13]. © 2004 American Institute of Physics.)

Dislocation nucleation upon indentation

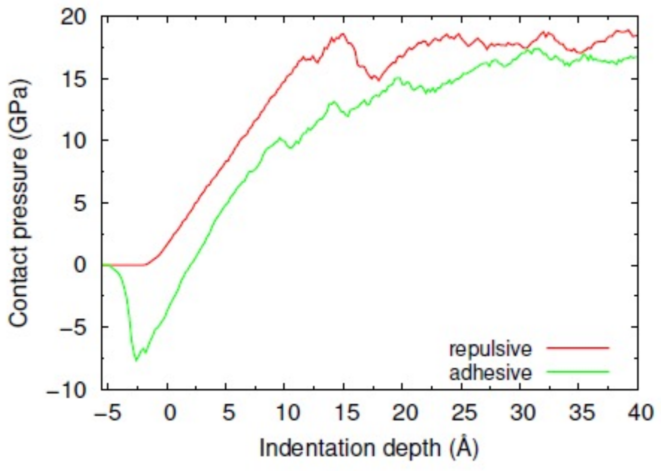




Repulsive vs adhesive potential

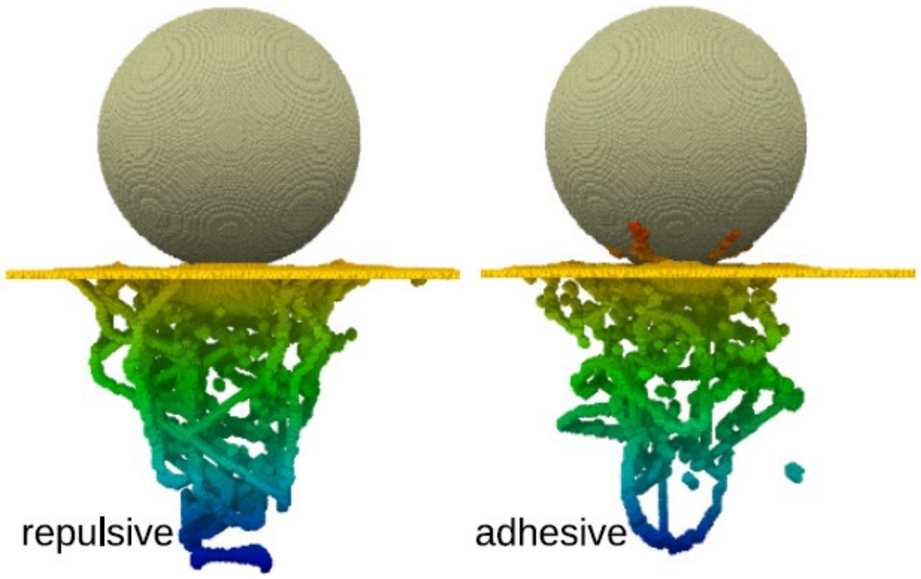


(a)



(b)

Morse potential
between C and Fe



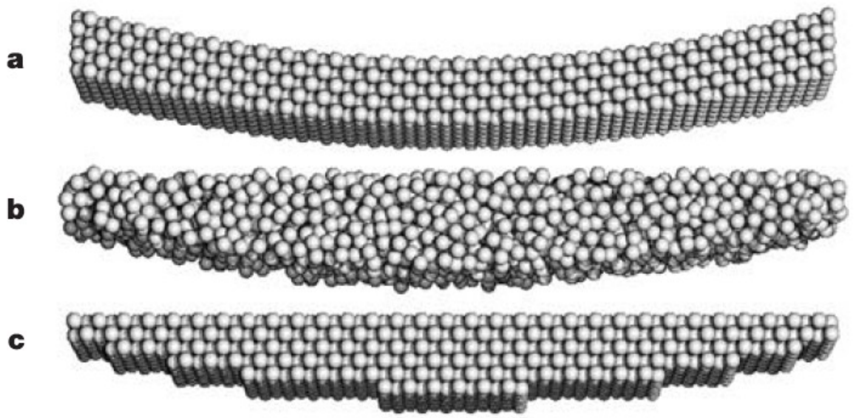
Contact mechanics

The breakdown of continuum models for mechanical contacts

Binquan Luan[1] & Mark O. Robbins[1]

Vol 435|16 June 2005|**doi:10.1038/nature03700**

MD is used to test the limits of contact mechanics under ideal conditions
Assumptions:
1) Continuum displacements and strain fields
2) Perfectly flat surfaces

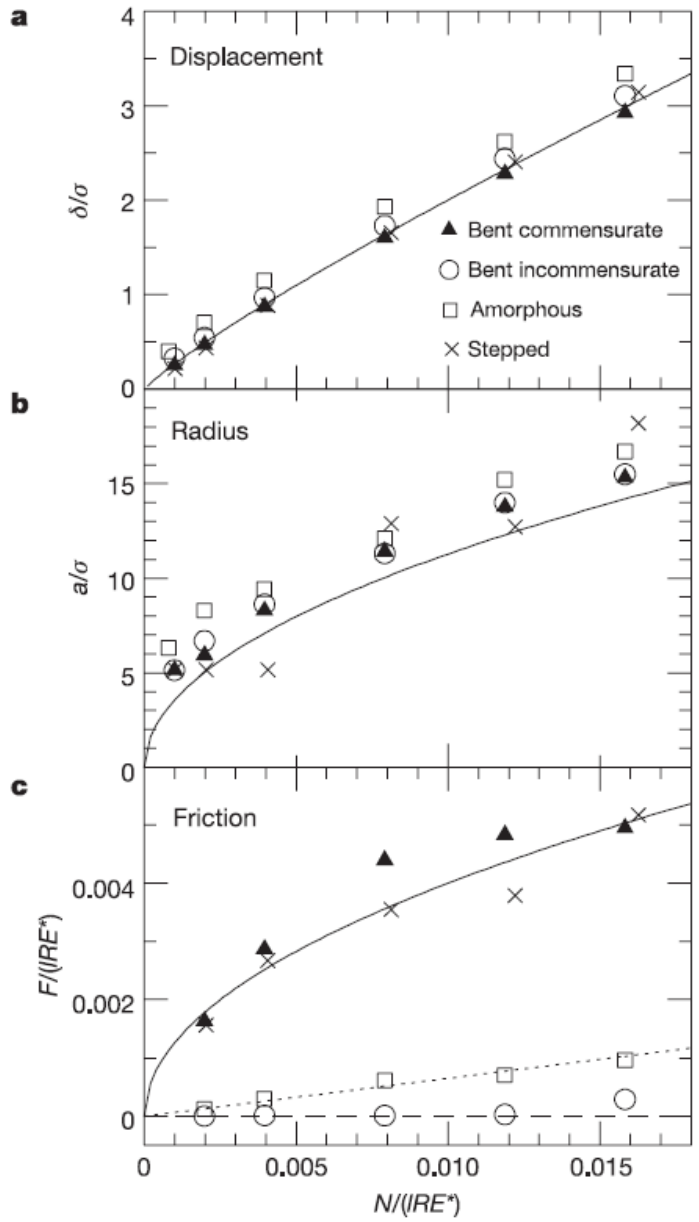


Cylindrical surfaces with different atomic-scale roughness.

A simulation that can be conducted at small and large scale:
Rigid cylinder or sphere indenting elastic flat body

Atoms of the cylinder interact with the substrate through LJ

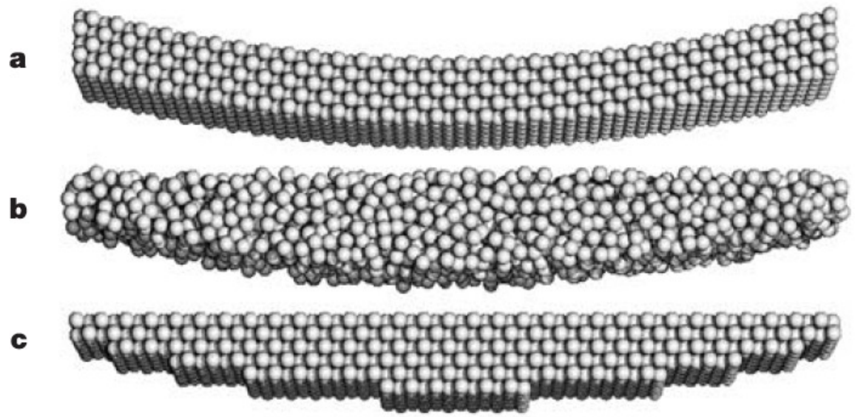
Displacement, contact area, friction



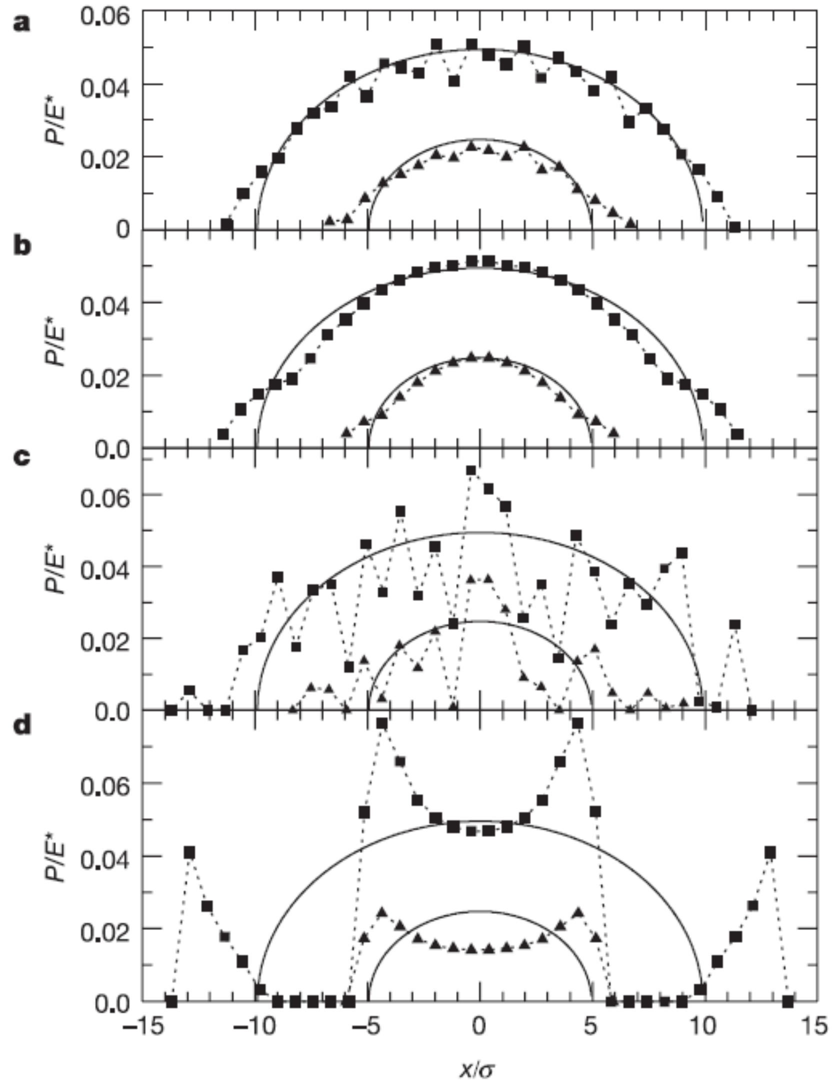
The assumption on displacement is good

Contact area is underestimated

Friction strongly depends on atomic roughness



Pressure profiles

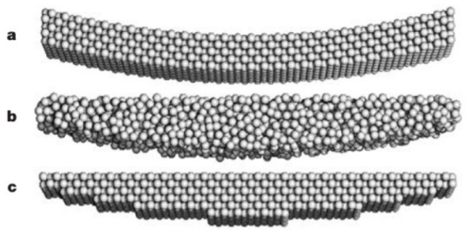


incommensurate

commensurate

amorphous

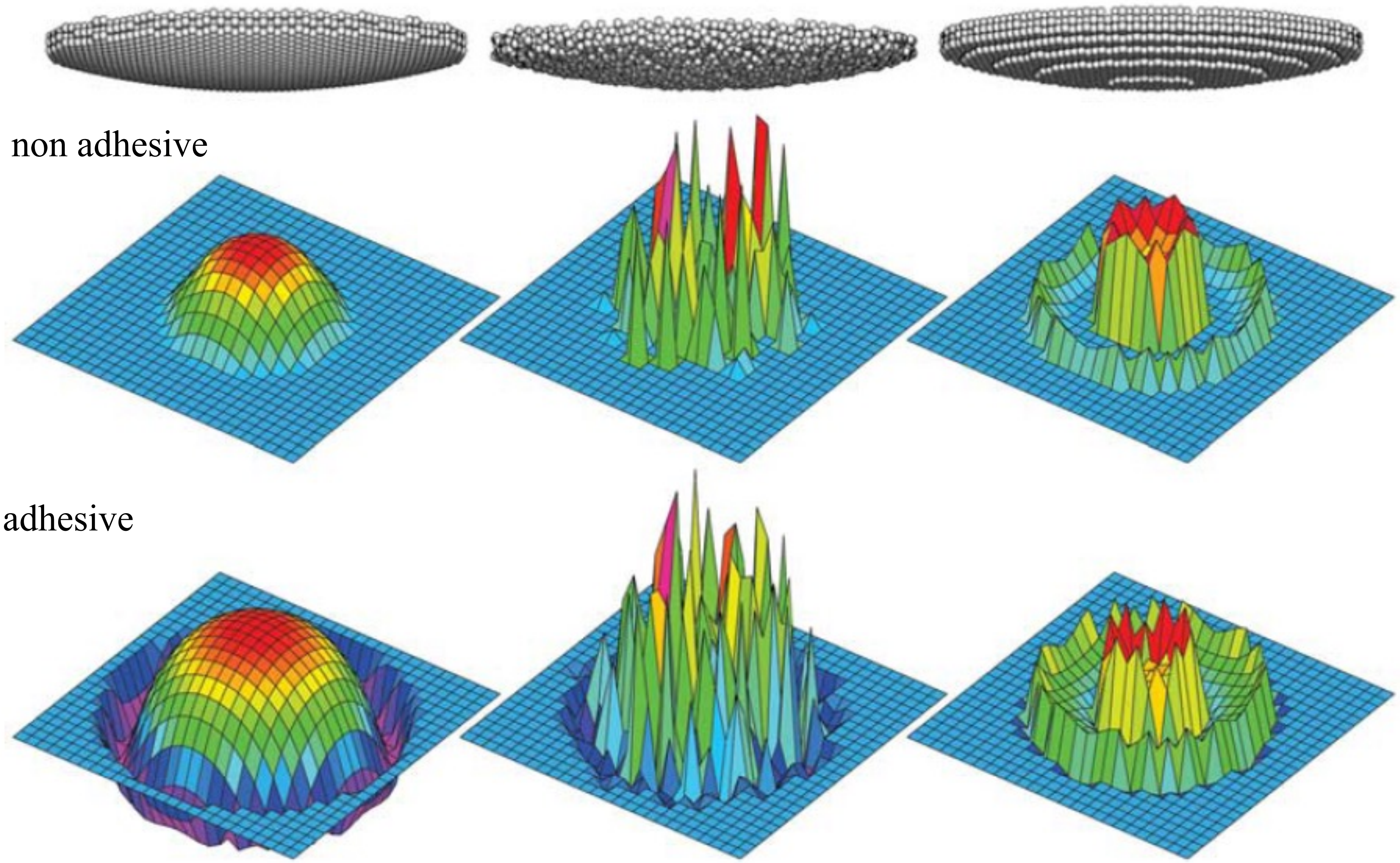
stepped



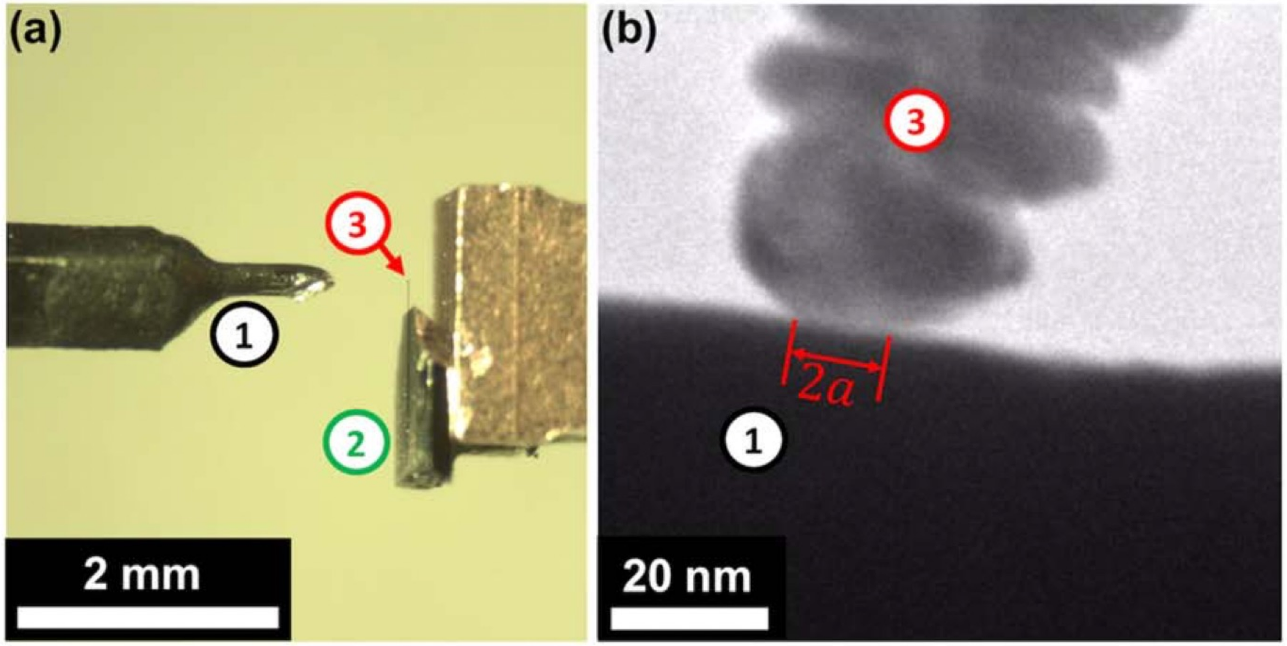
Stress distribution, effect of adhesion

non adhesive

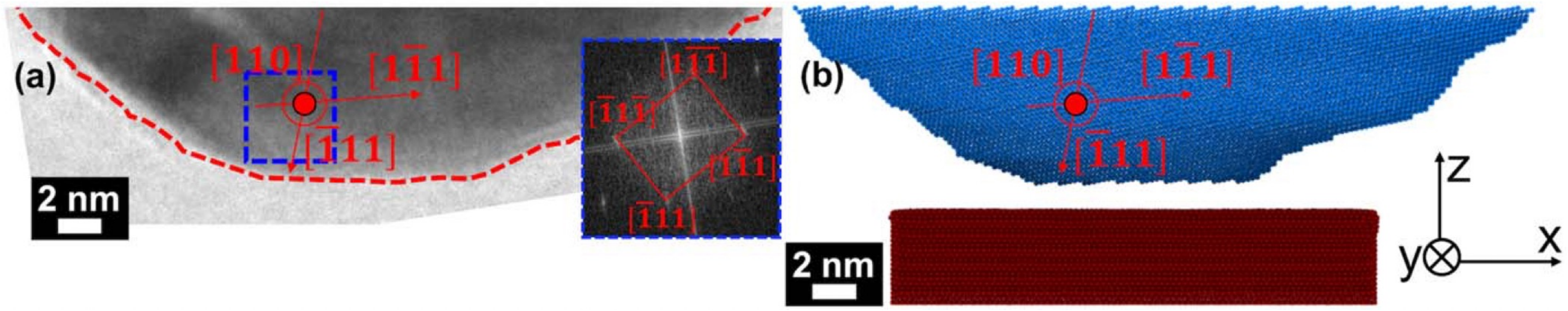
adhesive



Contact between platinum nanotips

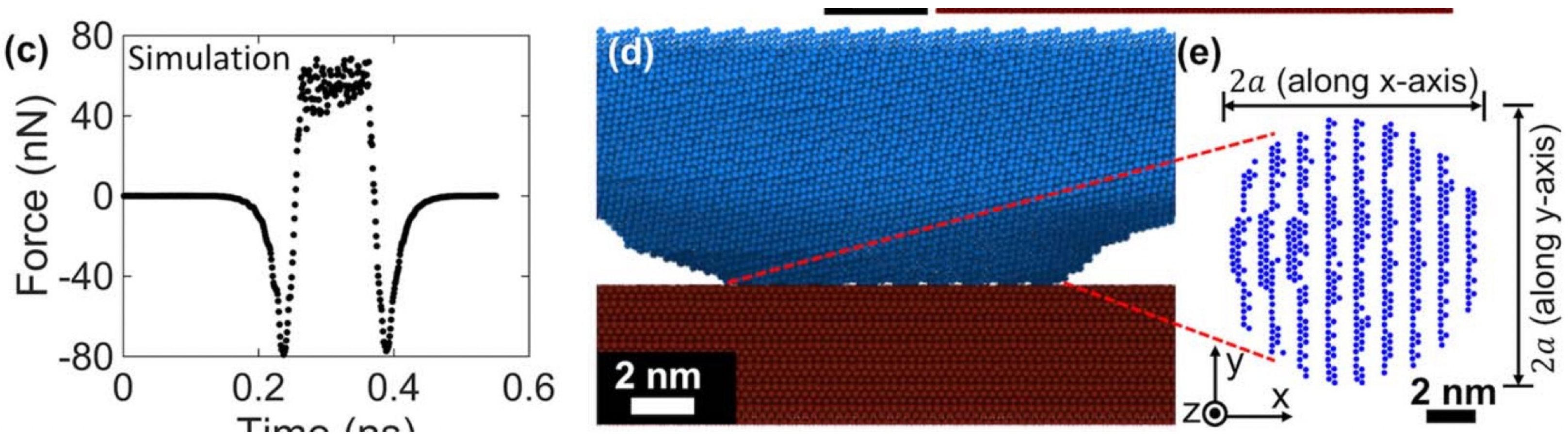


Pt contacts
are used in microscopy



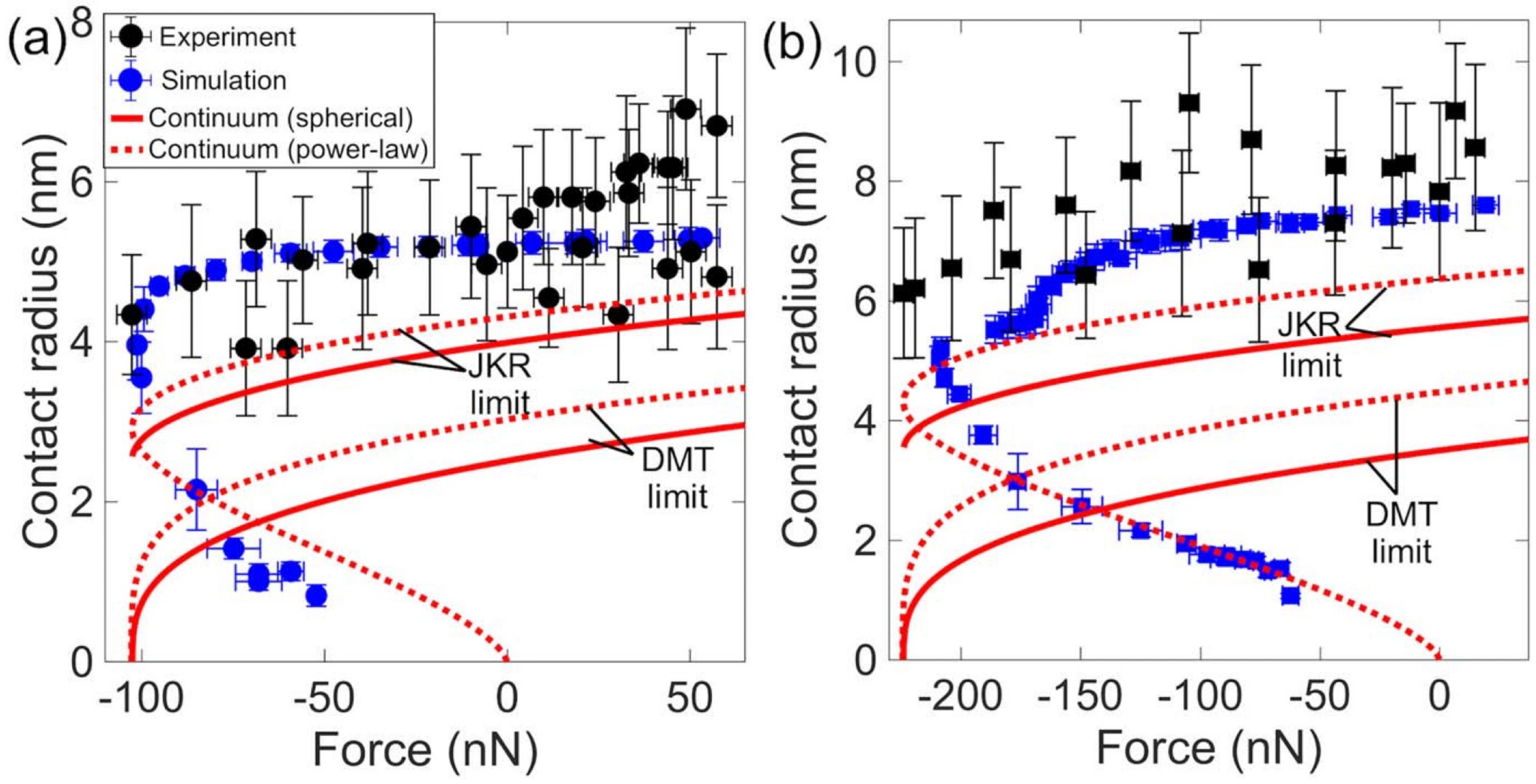
Vishnubhotia *et al.* Nanotechnology 30 (2019)

Platinum nanotips



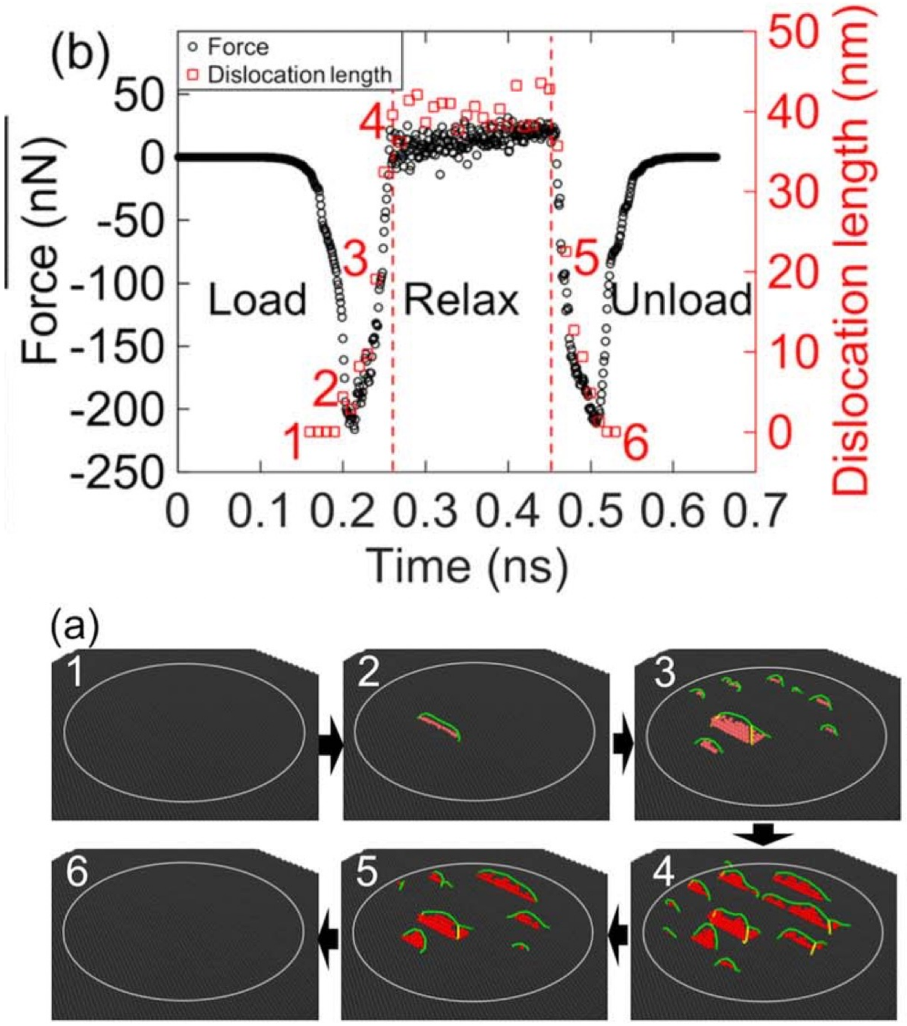
Simulations were carried out using the MD simulation package LAMMPS [29]. The embedded-atom method potential was applied to model the interaction of atoms within the probe and the substrate [30]. A Lennard-Jones potential was used to model interactions between the probe and substrate in order to reproduce the interfacial interaction strength in experiments, which could be affected by factors that are not explicitly captured in the model (e.g., surface adsorbates).

Comparison with classical theories



Rather significant disagreement with the theories that are usually used to calculate contact area.

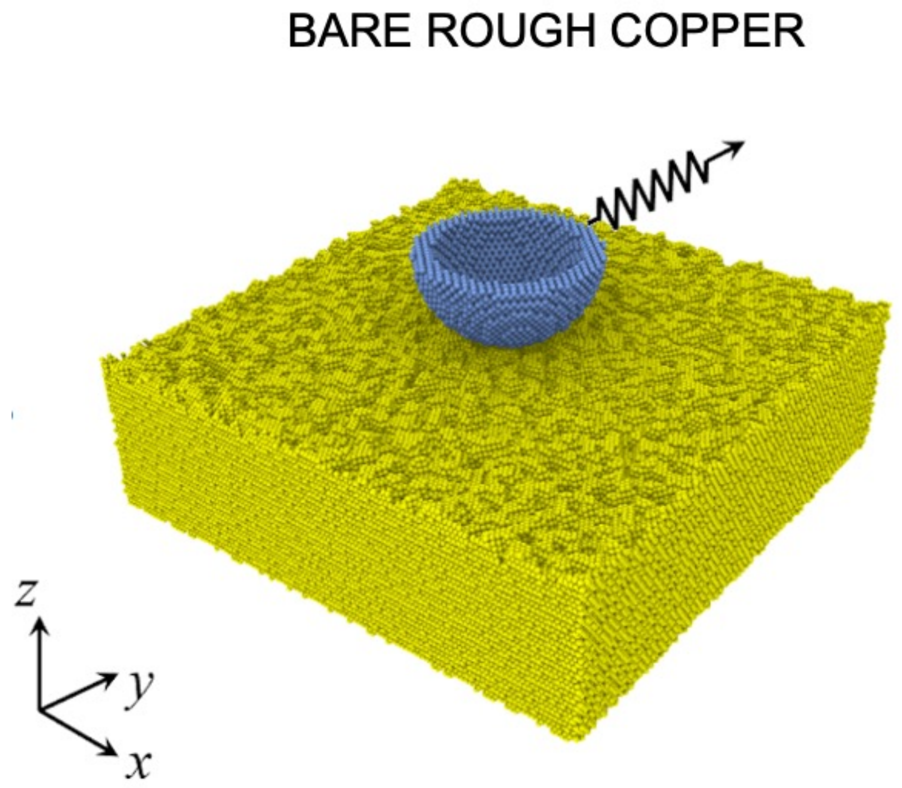
Relaxation due to to dislocation motion



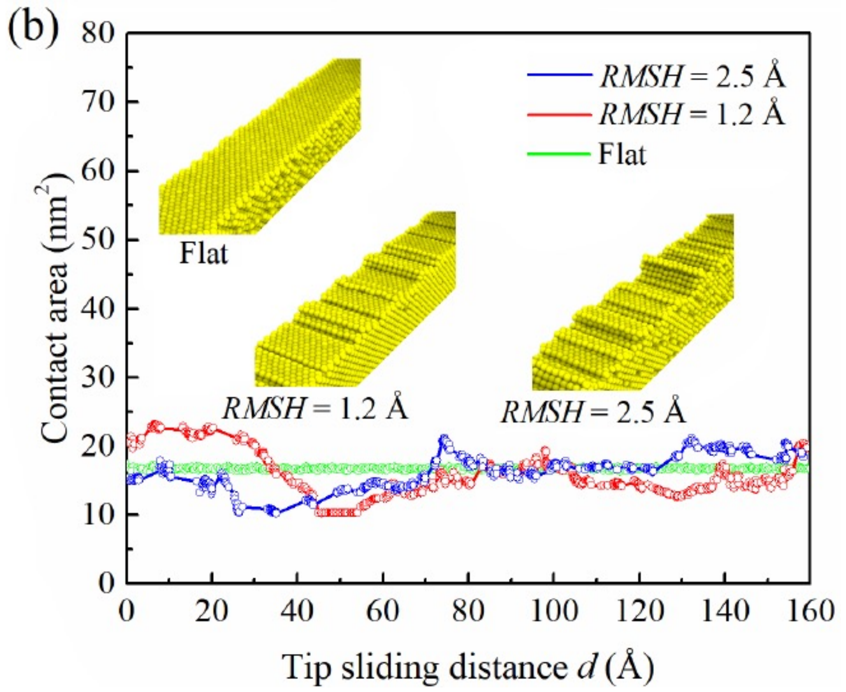
Relaxation of stresses due to significant dislocation nucleation from surface steps, which are partly flattened, giving rise to a larger contact area than predicted by adhesive theories.

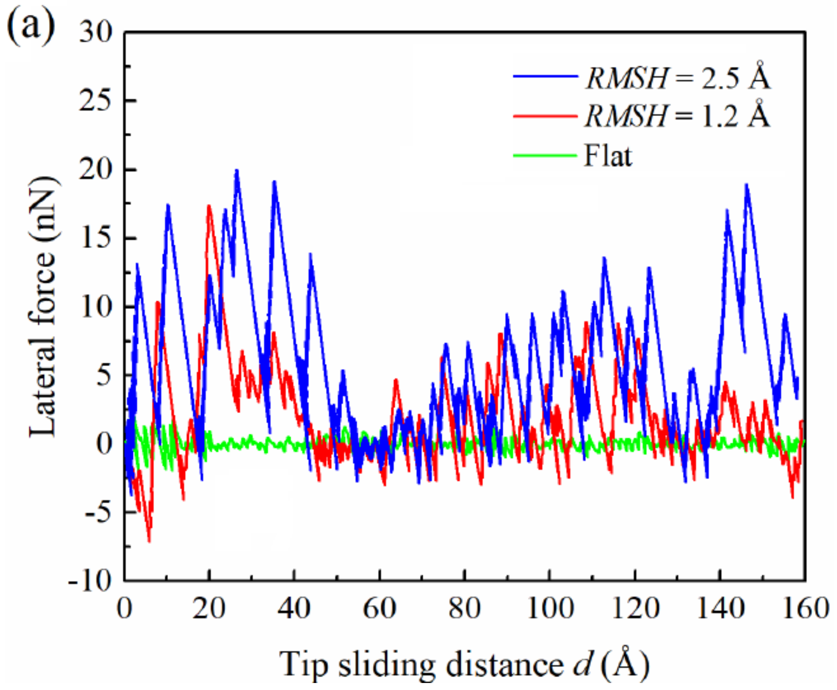
Full dislocation disappearance during unloading

Slider on a surface

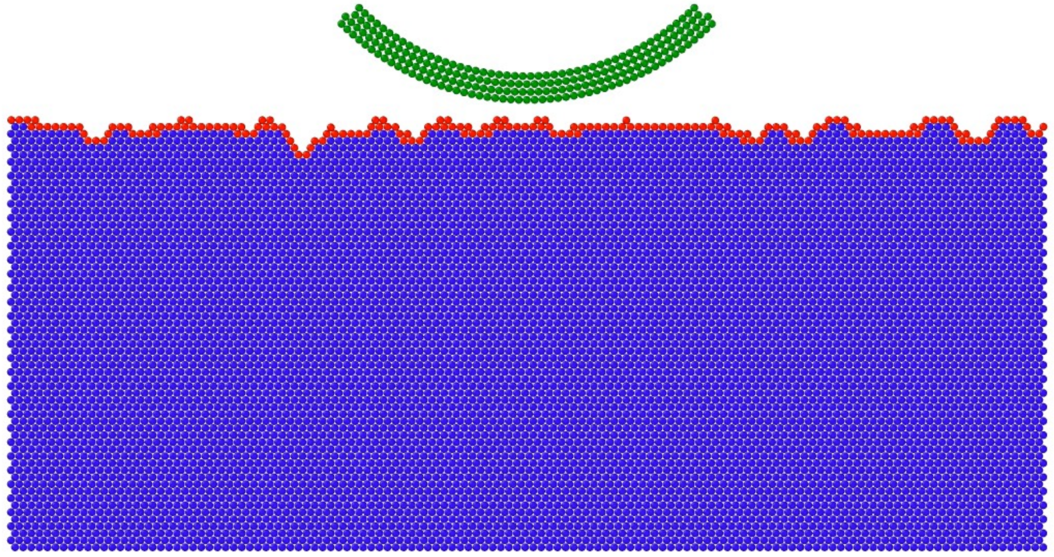


Effect of roughness on friction force

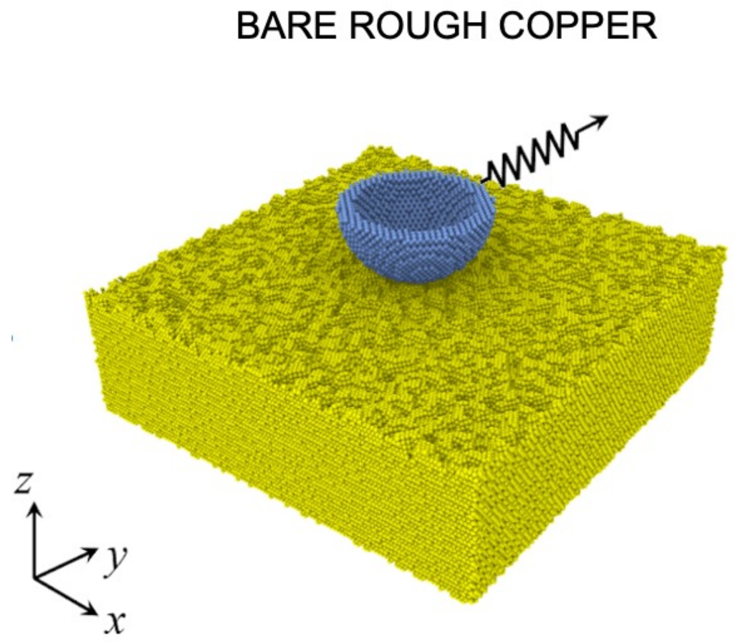


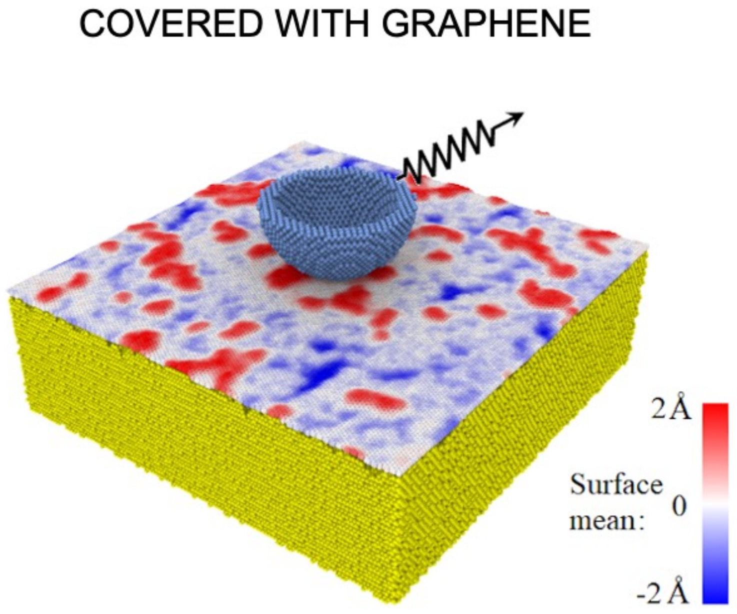


Light scratching

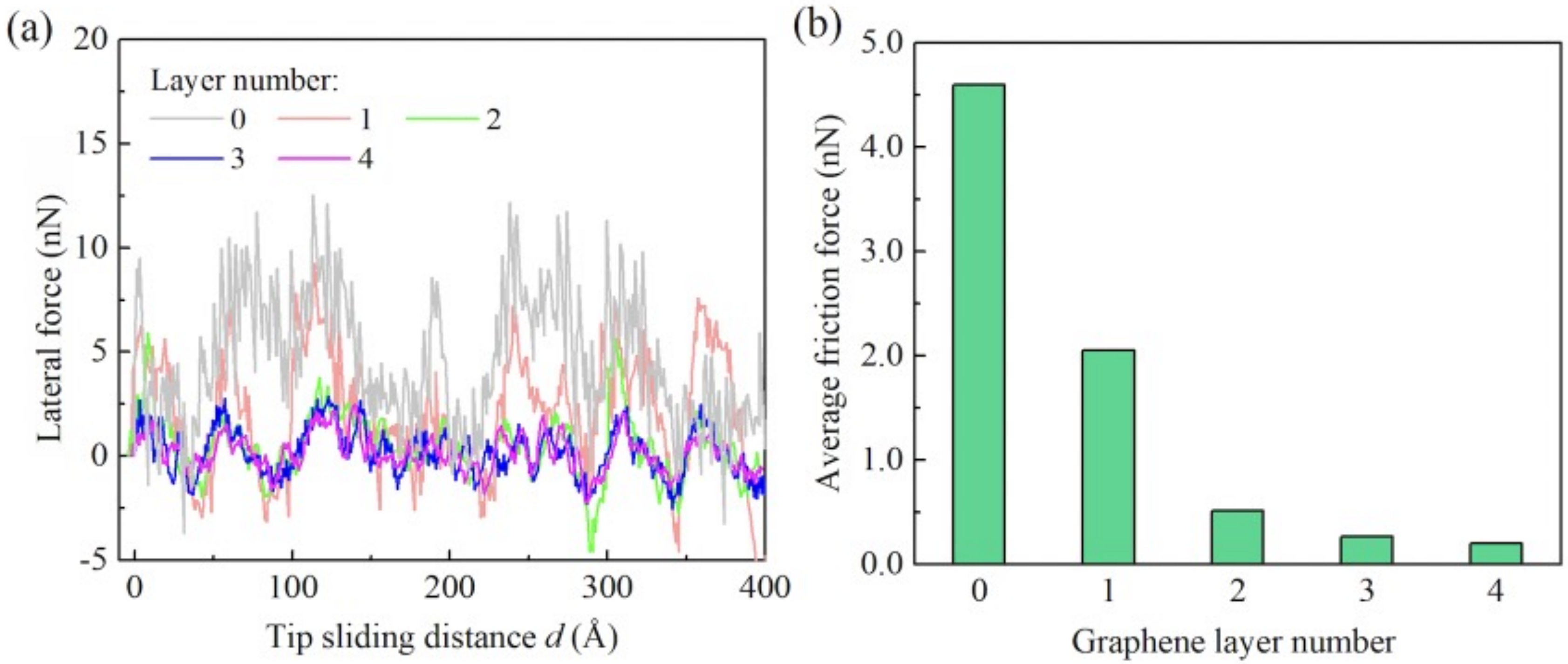


Lubricating properties of graphene

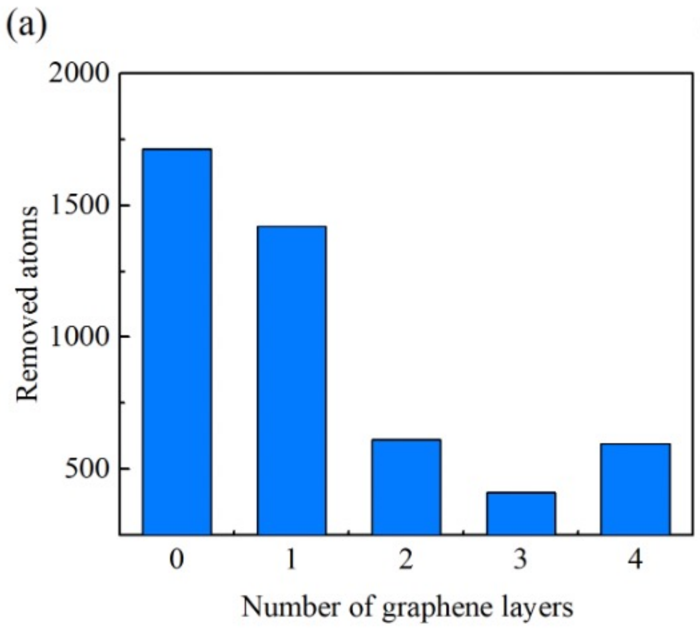


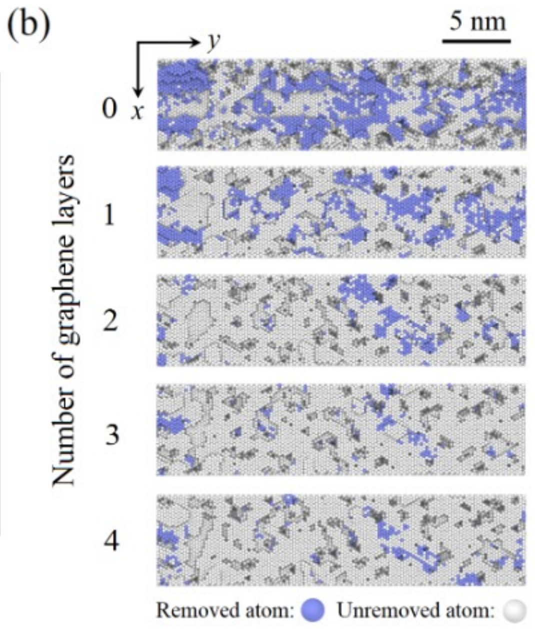


Lubricating properties pf graphene

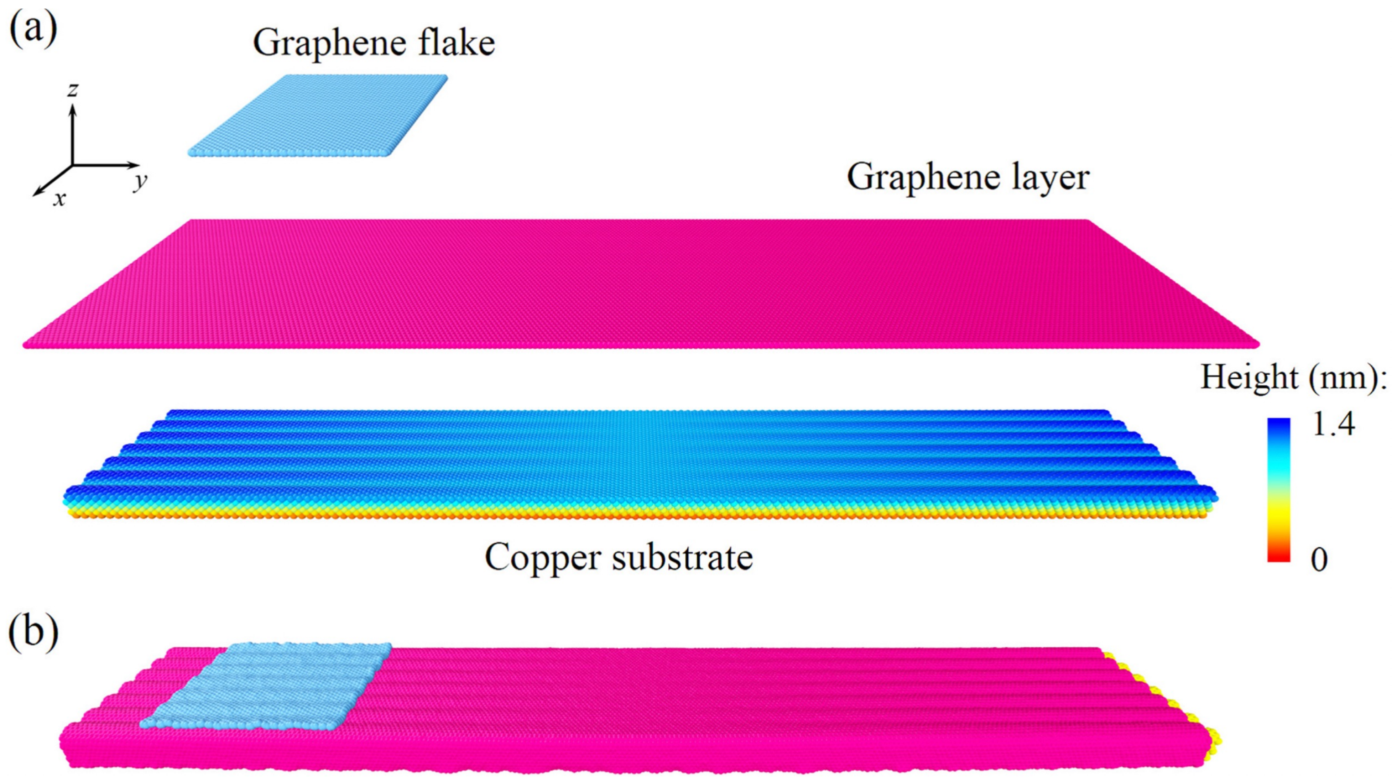


Wear protection





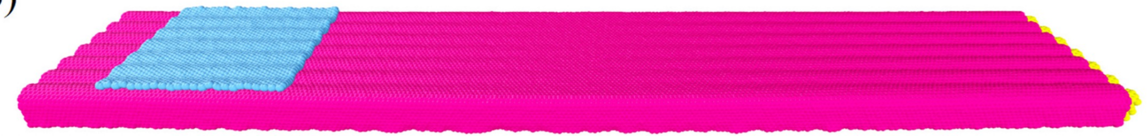
Graphene flake on undulated substrate



A graphene flake is positioned on an undulated substrate covered by a graphene layer that fully conforms to the substrate.

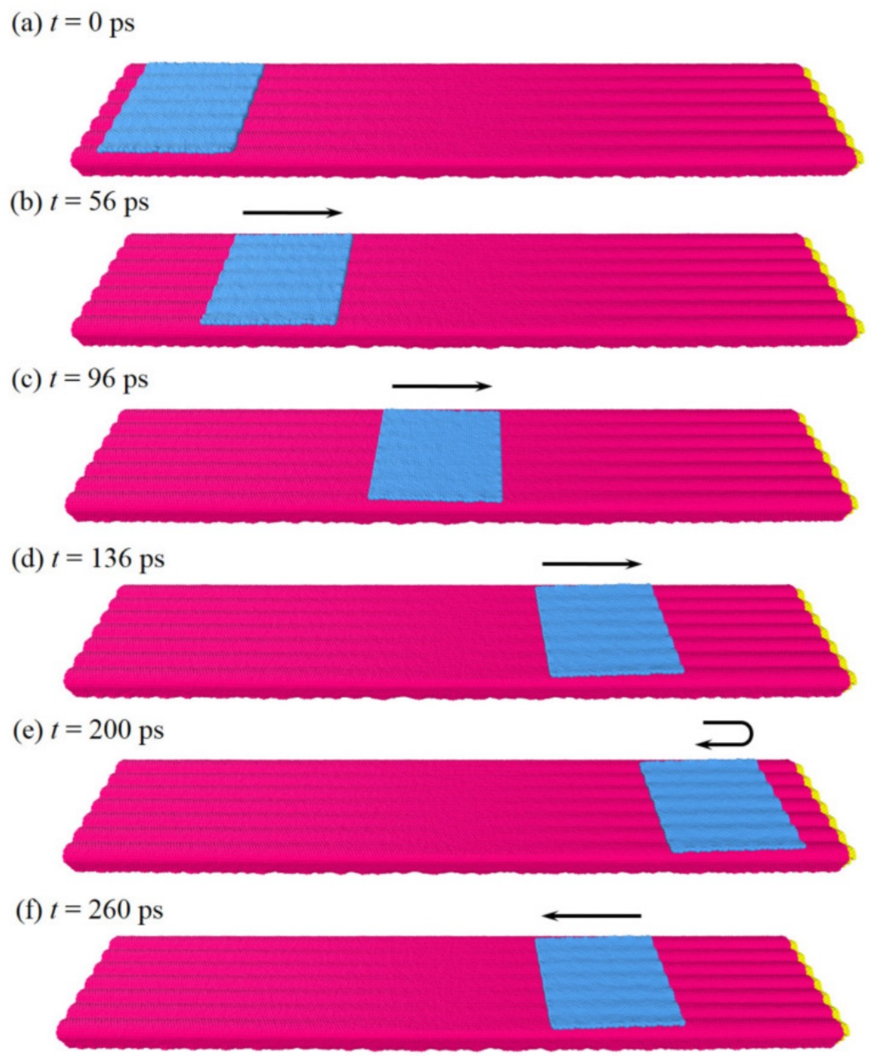
Graphene flake on undulated substrate

(b)



The covalent carbon–carbon bonds in the graphene layer and the flake are described by the Adaptive Interatomic Reactive Empirical Bond Order (AIREBO) potential [24]. The Van der Waals interactions between the graphene layer and the copper substrate, and those between the flake and the graphene layer are described by the classical 6–12 pairwise Lennard-Jones (LJ) potential. The LJ parameters for the copper substrate–graphene interaction are adopted from previous work [25] with a well-depth of $\varepsilon = 27.58$ meV and an equilibrium distance of $\sigma = 0.3083$ nm. Following [16], the potential well-depth for the interaction between graphene layer and flake is $\varepsilon = 2.968$ meV, and the equilibrium distance is $\sigma = 0.3407$ nm.

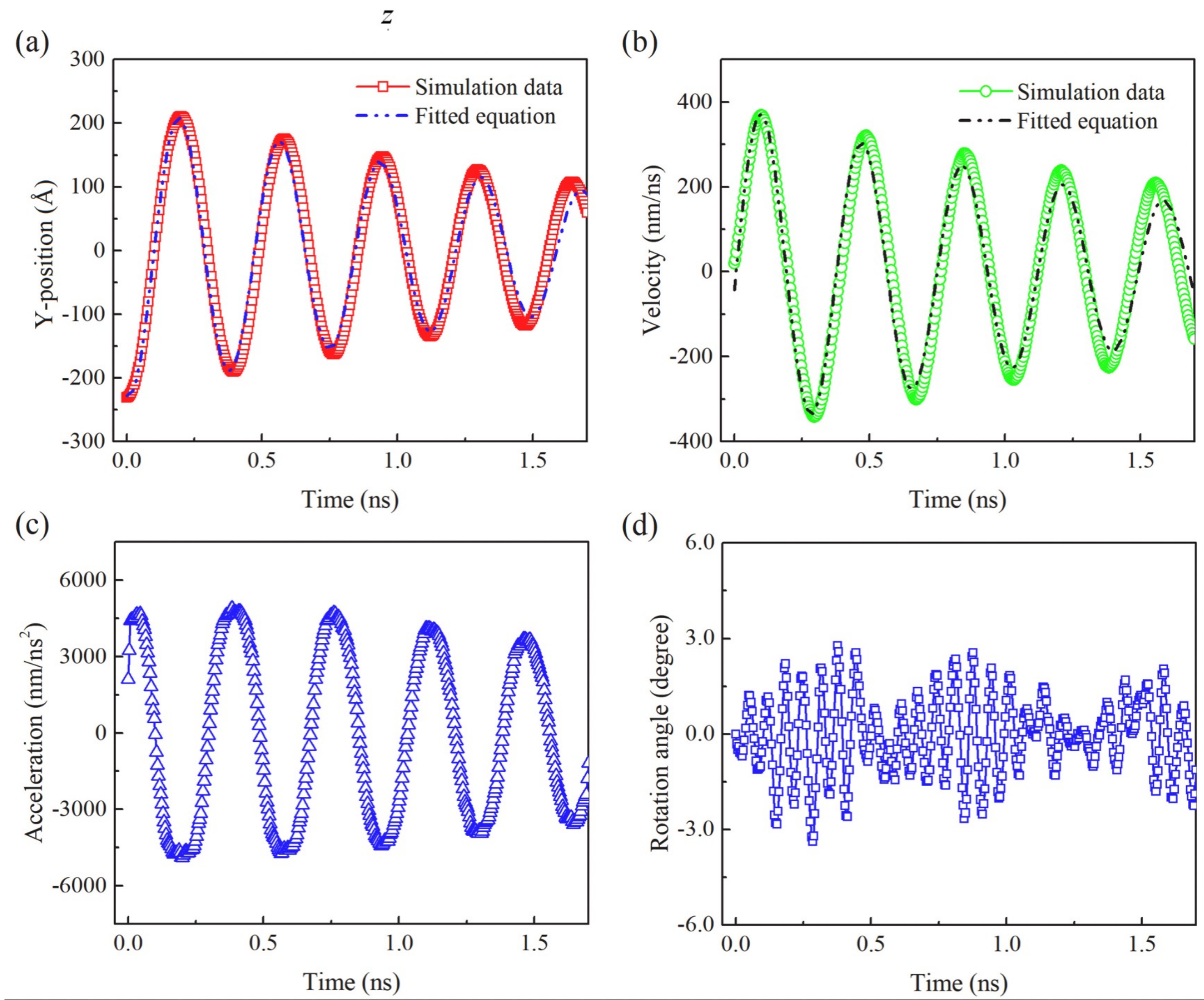
Graphene flake as oscillator



The undulation of the substrate has an amplitude gradient, which acts as driving force for the motion of the flake, which tries to minimize its corrugation energy.

Friction acts as a damper on the oscillations, which stop after a few cycles.

Similarity with harmonic oscillator



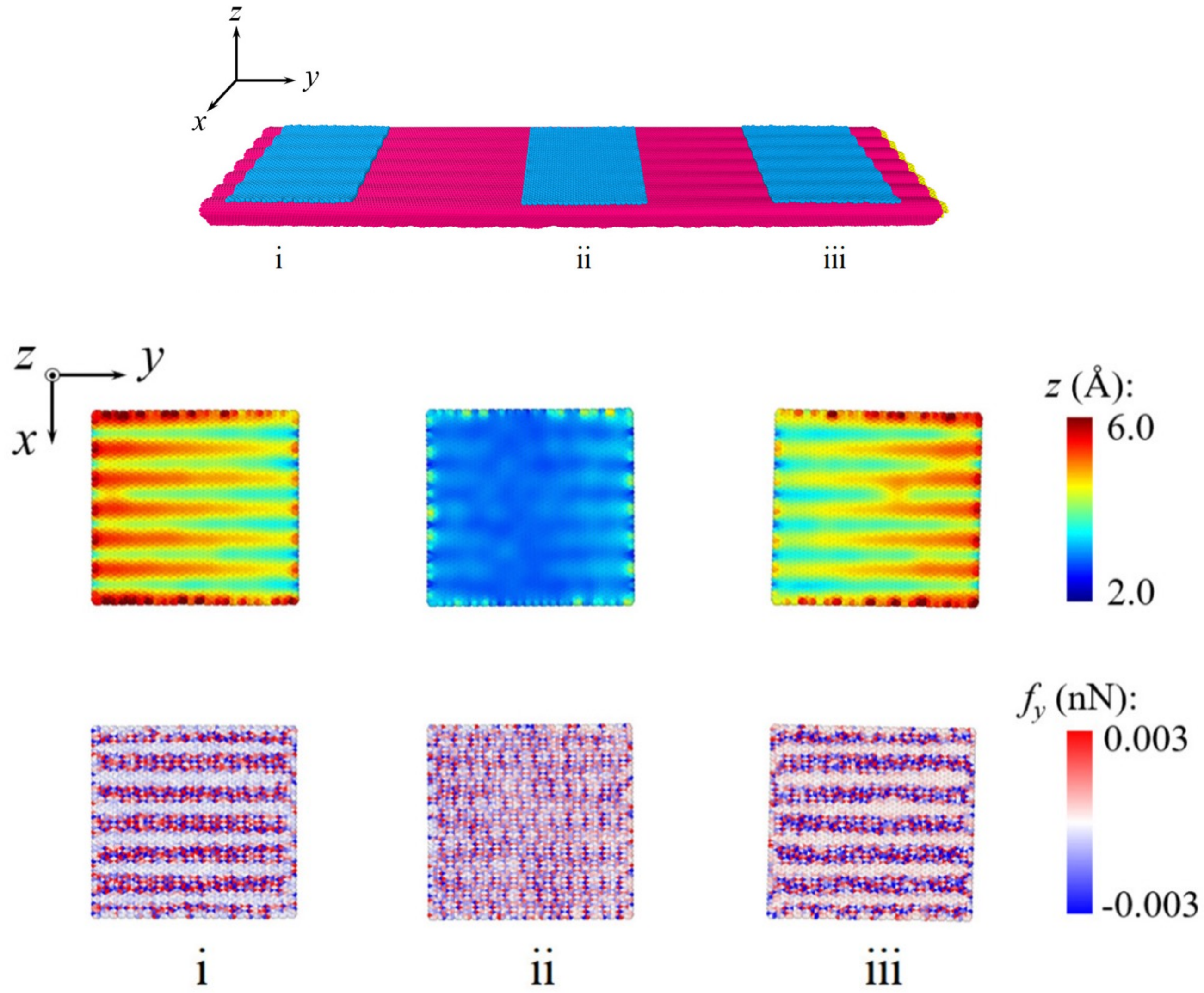
):
5.0

2.0

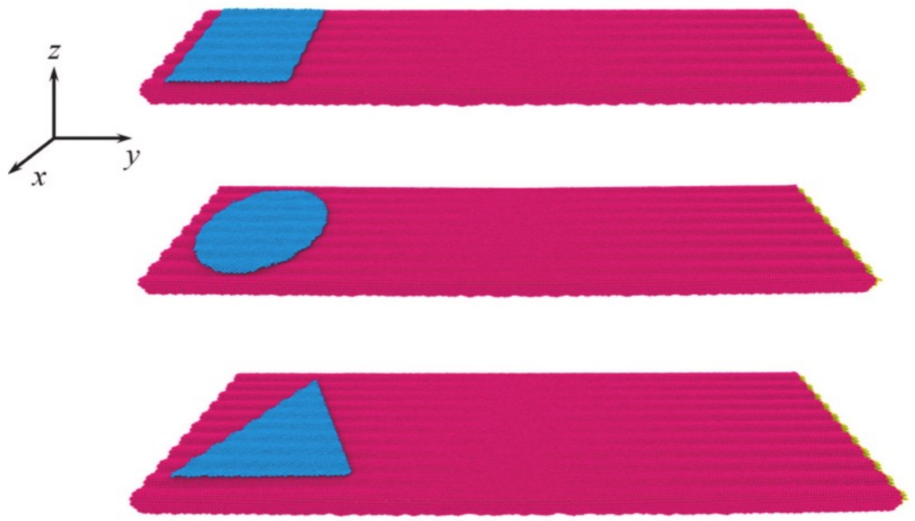
):
).003

0.003

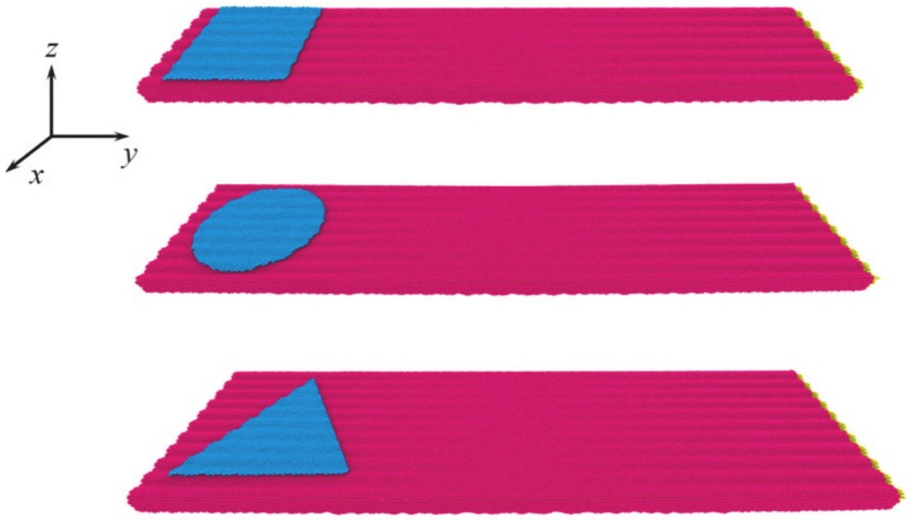
Oscillation of a graphene flake on undulated Cu

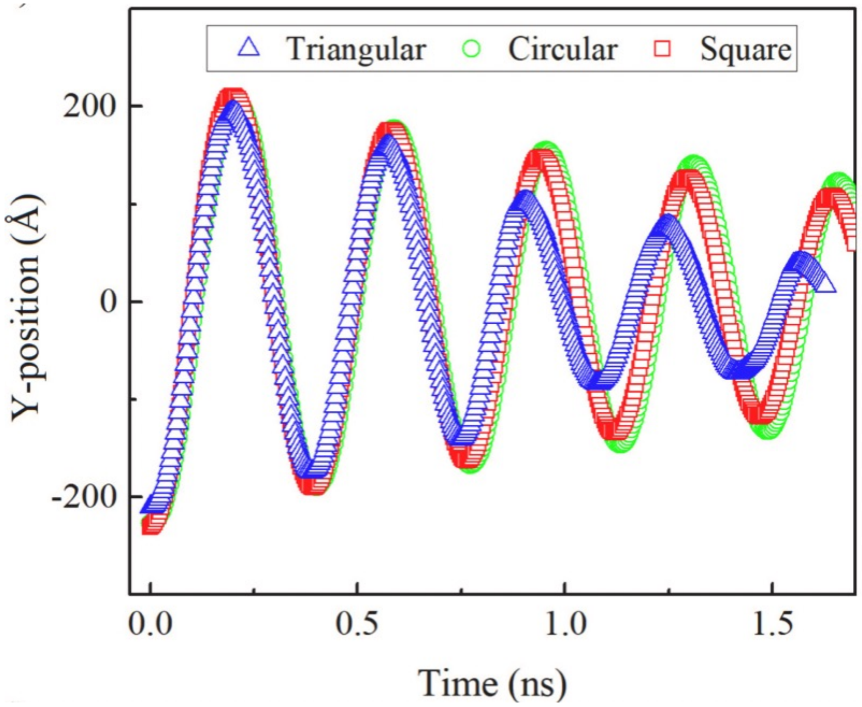


Which shape is best?



Which shape is best?





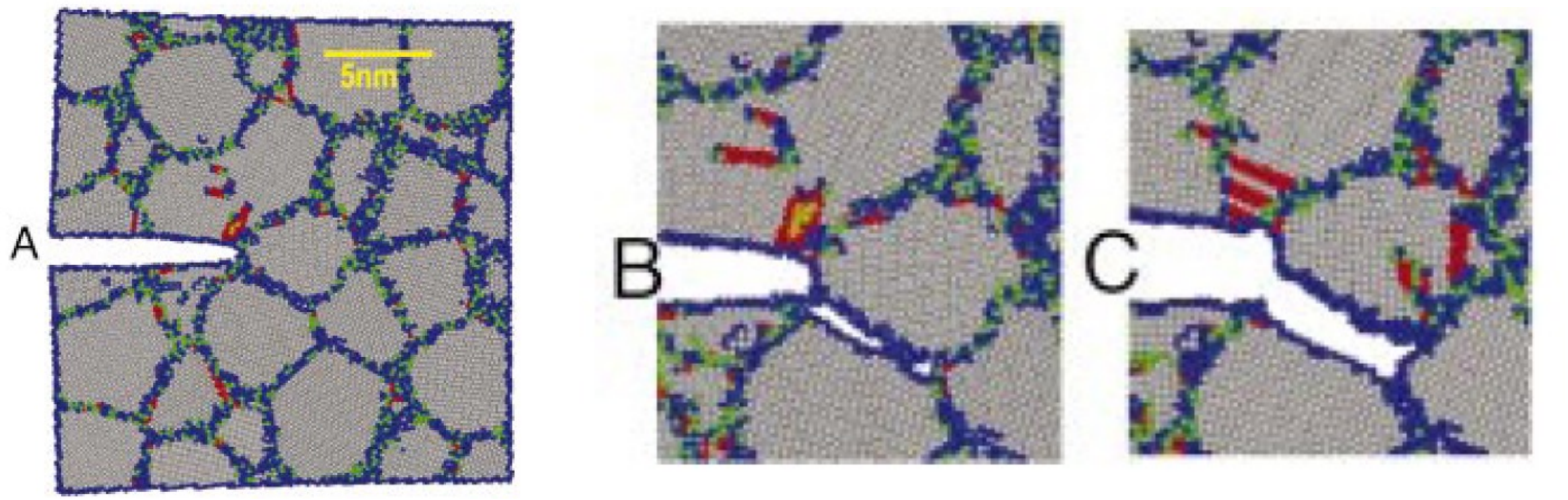
The circular is best because of less losses due to rotations

Polycrystals

How would you create a polycrystalline structure for your MD code?

Fracture

Nanocrystalline Ni, average grain size: 5nm
EAM potential from DFT, reproducing equilibrium lattice properties and defect properties

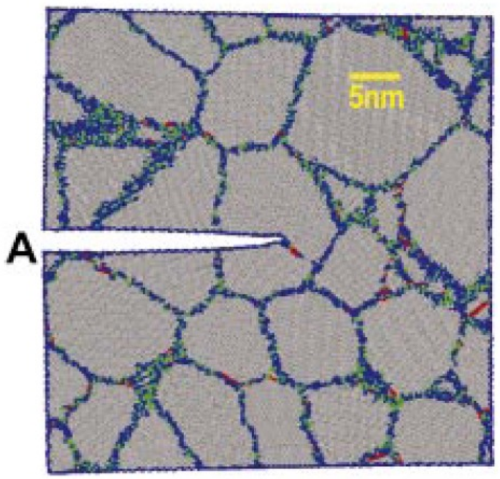


Griffith load	2 x Griffith load	2.5 x Griffith load
blunting	more blunting and nucleation of voids	intergranular propagation

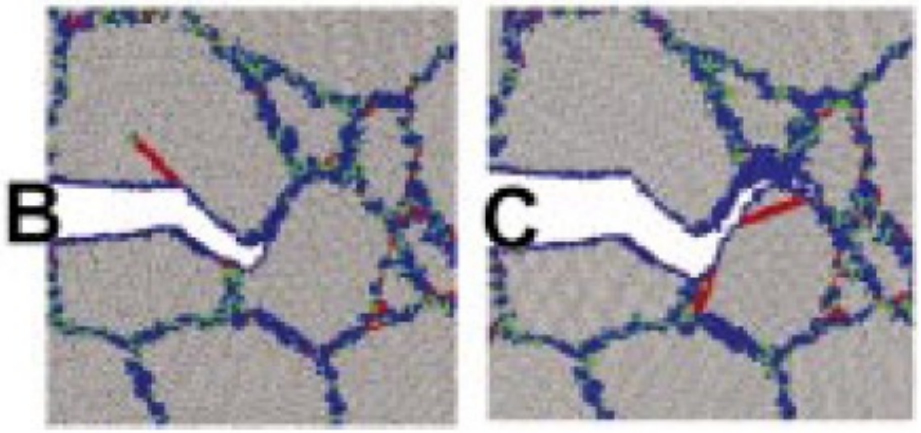
PHYSICAL REVIEW B **66**, 060101(R) (2002)

Fracture

Nanocrystalline Ni, average grain size: 10nm



Griffith load

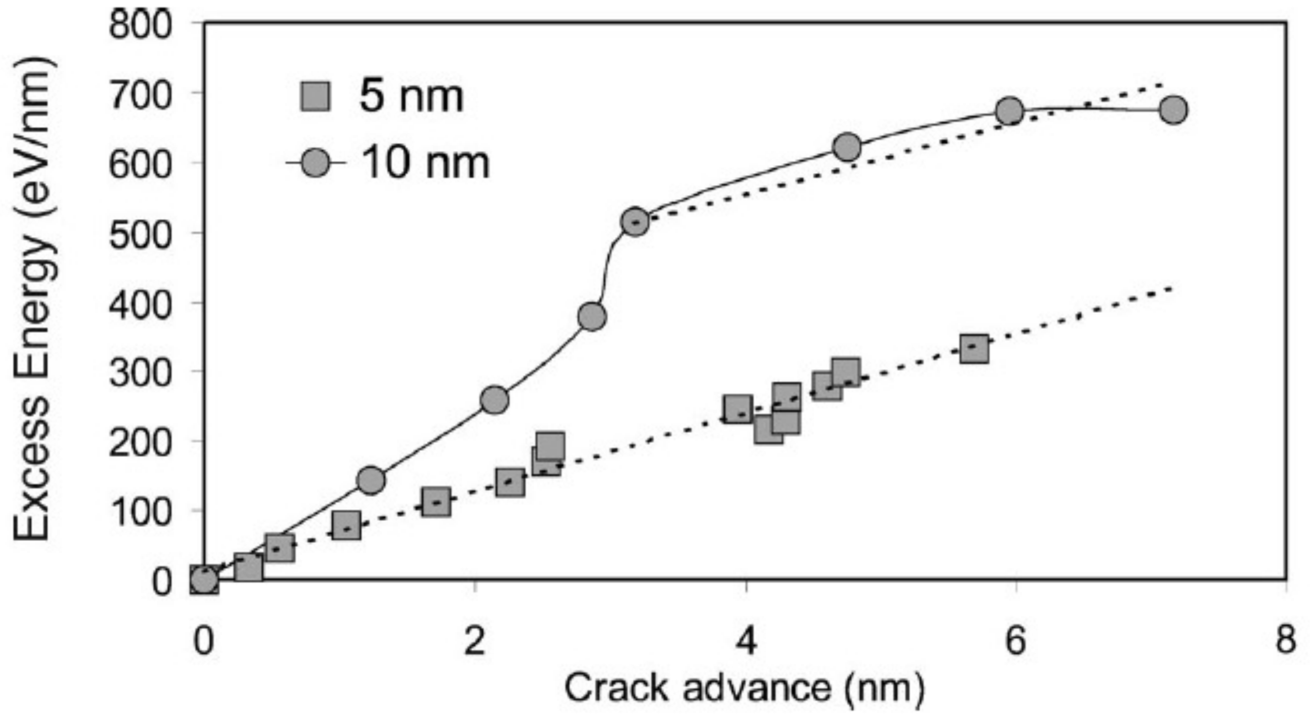


2 x Griffith load

2.5 x Griffith load

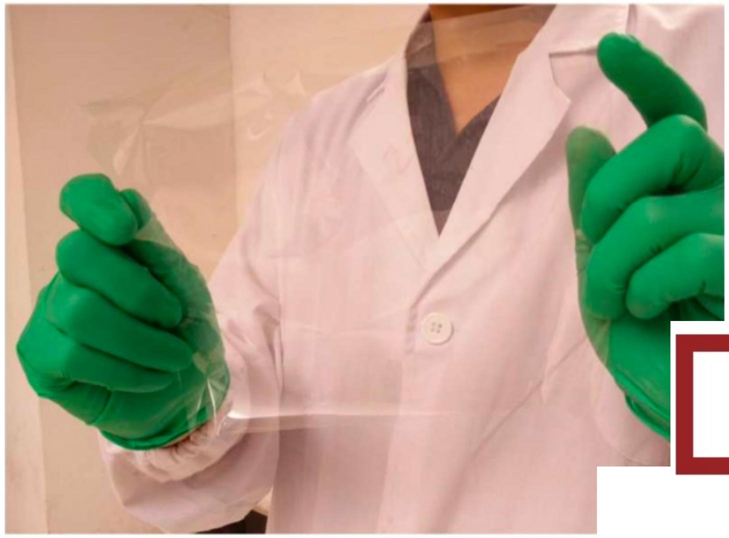
PHYSICAL REVIEW B **66**, 060101(R) (2002)

Fracture



800
700
600
500
400
300
200
100
0
Excess Energy (eV/nm)
5 nm
10 nm
0
2
4
6
8
Crack advance (nm)

Transparent flexible electrodes



Trasparenza

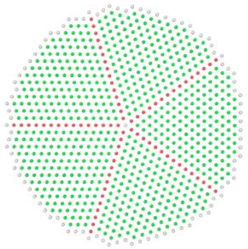
Flessibilità

Conducibilità

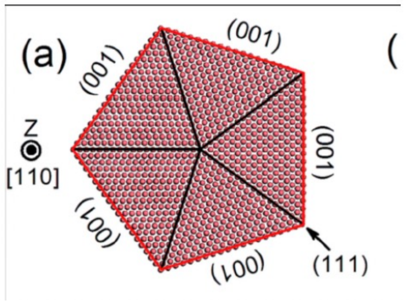
Penta-nano twinned nanotubes

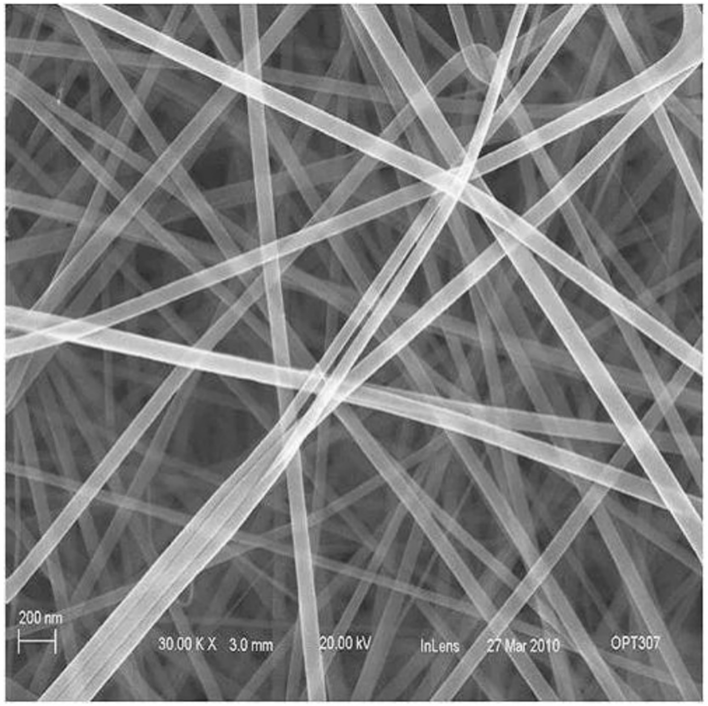
Proprietà meccaniche

Giunzioni

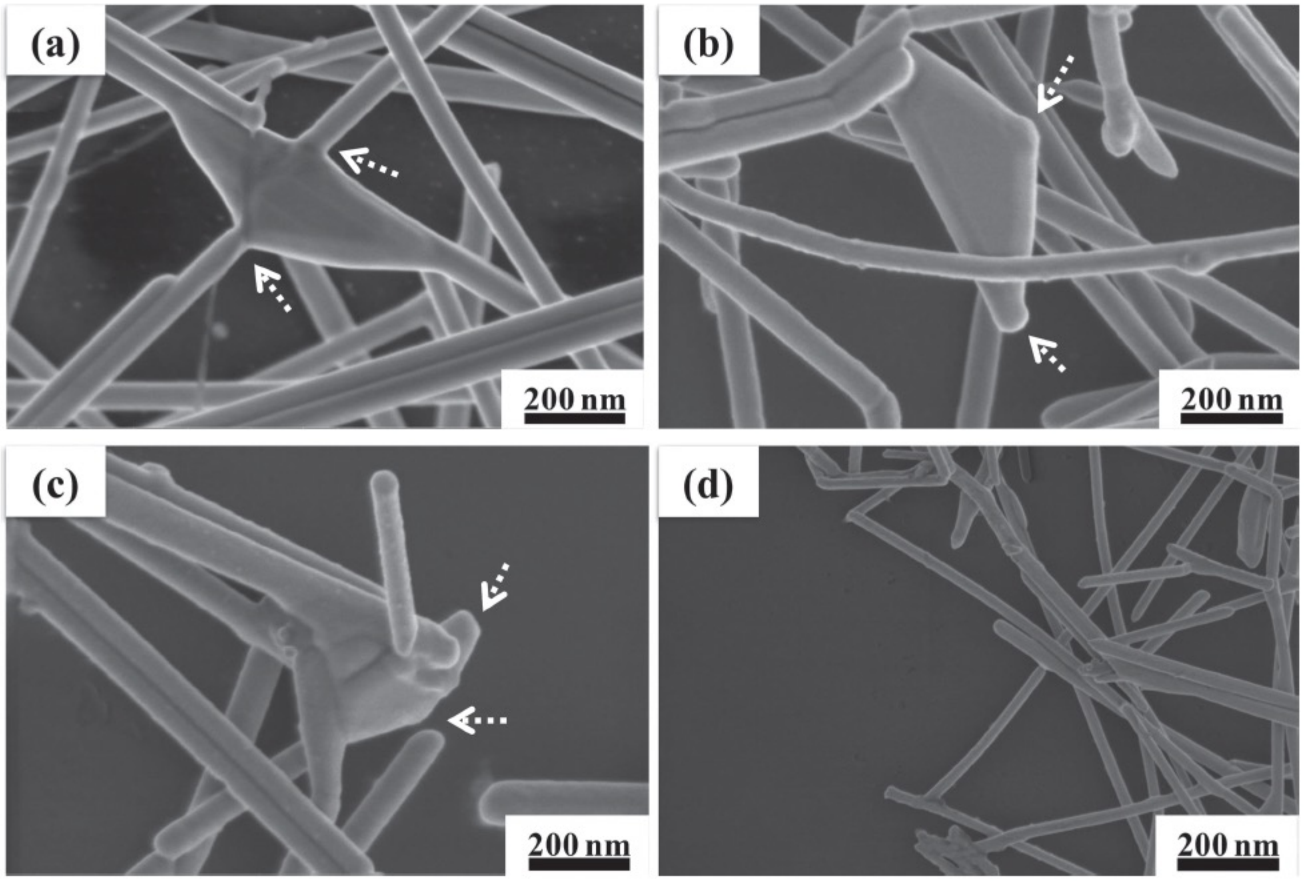


- Formazione network conduttivo.
- Integrità meccanica determina prestazioni elettriche.

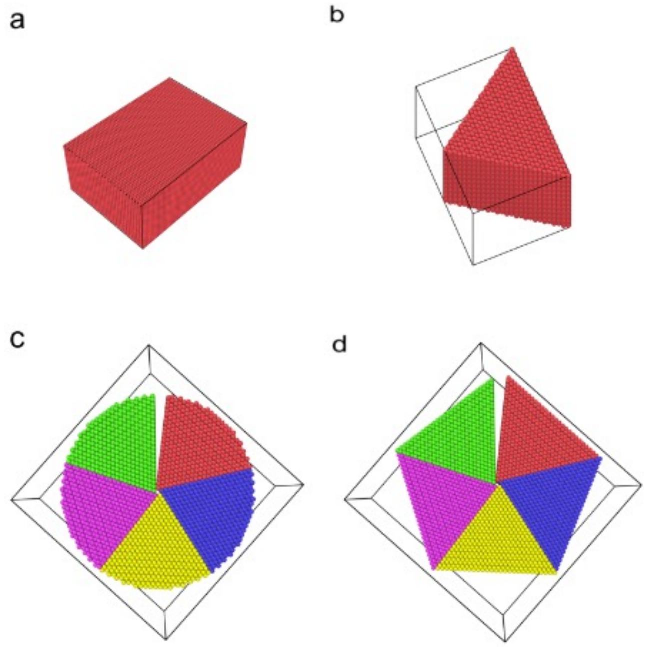


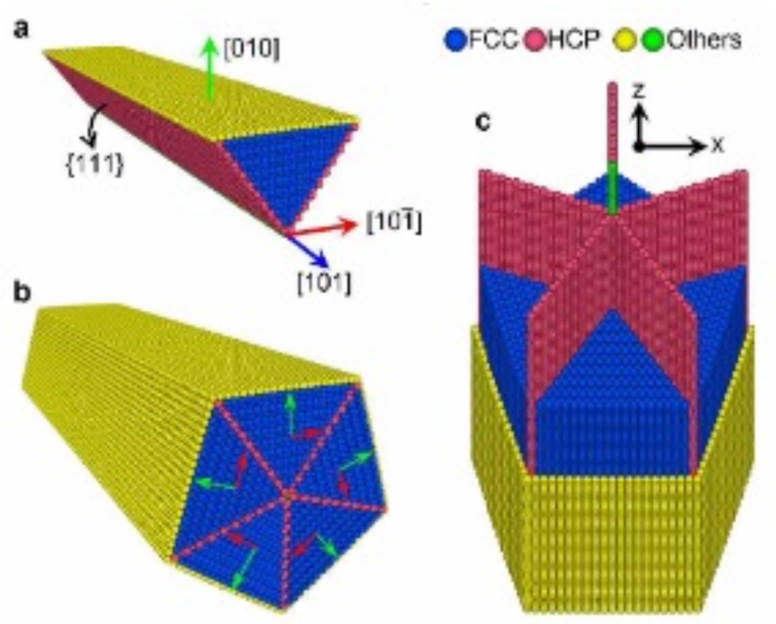


Non homogeneously welded

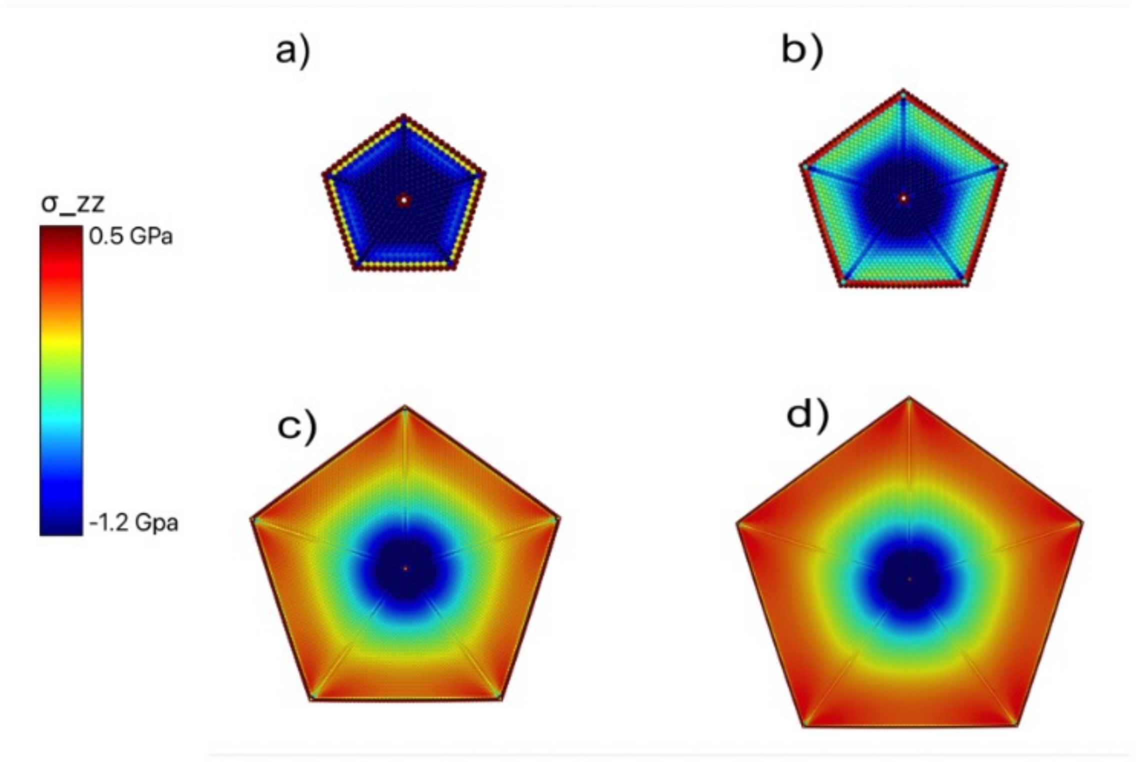


Creating the nanowire

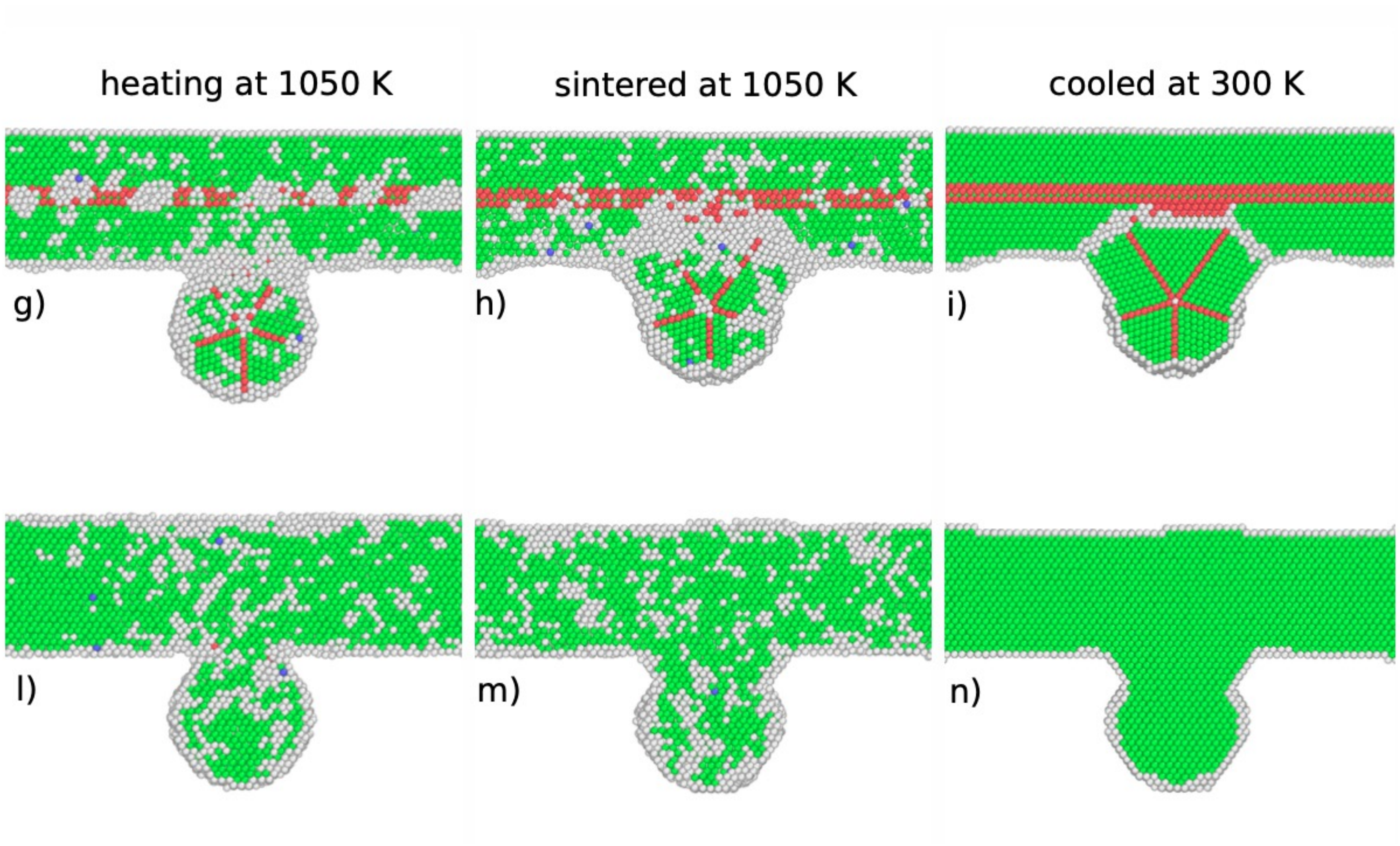




Non homogeneous stress state



Nanowelds



Limitations of MD simulations

- Interatomic potentials are not that accurate, one should check that they are appropriate for the material under investigation, otherwise DFT

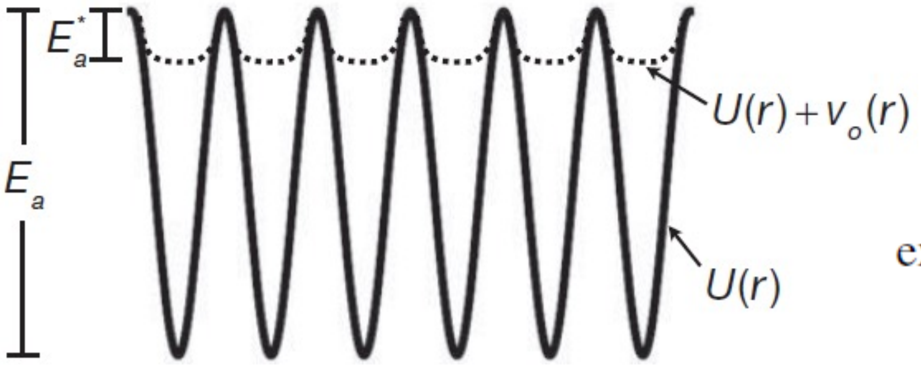
- Size: with a parallel computer multimillion atoms are possible, with supercomputers even multibillion atoms have been achieved.1 micron cube of copper has 10^{11} atoms, just doable

- the limitation on the atoms is given by the burden of computing the forces and energy. Since density N/L^3 must be kept constant, doubling the linear dimension of the cell means increasing the number of atoms by 8N, the simulation will take 8 times longer to run.

- Time: With a time step of the order of 10^{-15}-10^{-14} nanoseconds, a long simulation yields about a few nanoseconds. Many processes that are longer cannot be studied, accelerated dynamics gives hope on this

Accelerated dynamics

Hyperdynamics: a bias potential is added to decrease the barrier height without changing shape



Rate of jumping:

$$\exp(-E_a/k_B T)$$

$$\Downarrow$$

$$\exp(-(E_a^* - E_a)/k_B T)$$

Simulations in the microsecond range with EAM potentials for metals

LAMMPS

Large-scale **A**tomic/**M**olecular **M**assively **P**arallel **S**imulator

What is LAMMPS?

LAMMPS is a classical molecular dynamics code designed to efficiently compute Newton's equations of motion for an ensemble of particles in a liquid, solid or gaseous state.

It can model atomic, polymeric, metallic, biological, granular systems using a variety of interatomic potentials and boundary conditions.

It is designed for parallel computers but runs also on single-processor machine as serial code.

LAMMPS is a freely-available open-source code → can be modified by users

It is written in C++.

Download and compilation

1) Download the lammps package from docs.lammps.org

2) cd lammps/src   *# change to main LAMMPS source folder*

3) Compile the package to get the executable using the 'make' command:

```
make serial     # build a serial LAMMPS executable using GNU g++
make mpi        # build a parallel LAMMPS executable with MPI
```

This command creates the executable either serial lmp_serial or to be run in parallel lmp_mpi

Download the visualization package OVITO from https://www.ovito.org

Run the simulation

1) Move the executable to your bin directory:

mv lmp_serial to your bin directory→ mv lmp_serial ~/bin/

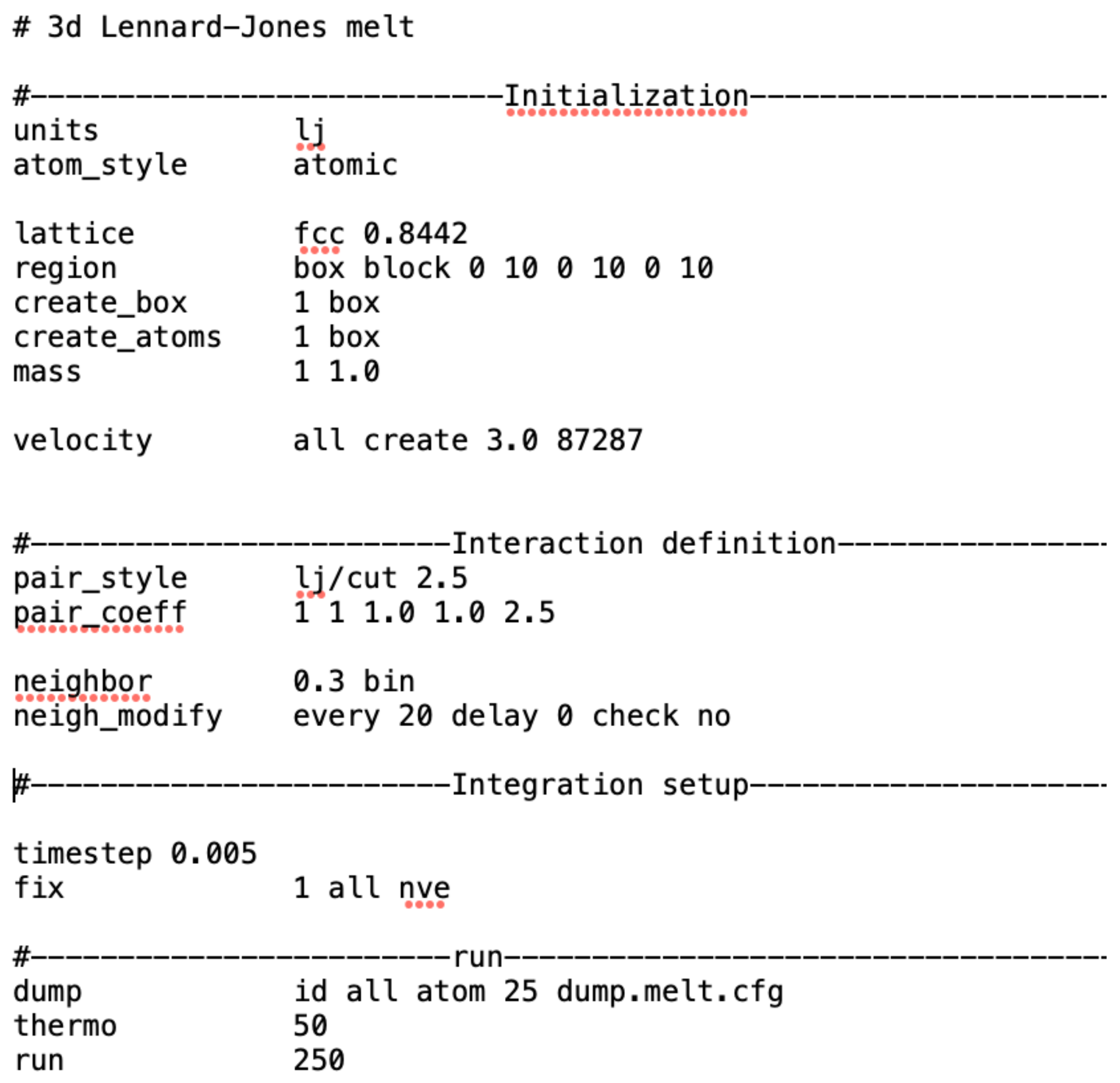
2) Run the serial code, from the directory where you have the input file in.file calling the executable from the bin directory:

```
~/bin/lmp_serial -in in.file
```

3) Alternatively, run the parallel code, from the directory where you have the input file in.file calling the executable from the bin directory:

```
mpirun -np 8 ~/bin/lmp_mpi -in in.file
```

Melting a crystal



```
# 3d Lennard-Jones melt

#---------------------------Initialization--------------------
units           lj
atom_style      atomic

lattice         fcc 0.8442
region          box block 0 10 0 10 0 10
create_box      1 box
create_atoms    1 box
mass            1 1.0

velocity        all create 3.0 87287


#------------------------Interaction definition---------------
pair_style      lj/cut 2.5
pair_coeff      1 1 1.0 1.0 2.5

neighbor        0.3 bin
neigh_modify    every 20 delay 0 check no

#------------------------Integration setup--------------------

timestep 0.005
fix             1 all nve

#------------------------run----------------------------------
dump            id all atom 25 dump.melt.cfg
thermo          50
run             250
```

Initialization

UNITS: LJ, real, metal, etc.

ATOM STYLE: atomic, molecular, charge

DIMENSIONS OF THE BOX

ATOMIC STRUCTURE

BOUNDARY CONDITIONS: p p p (periodic in x, in y, in z)

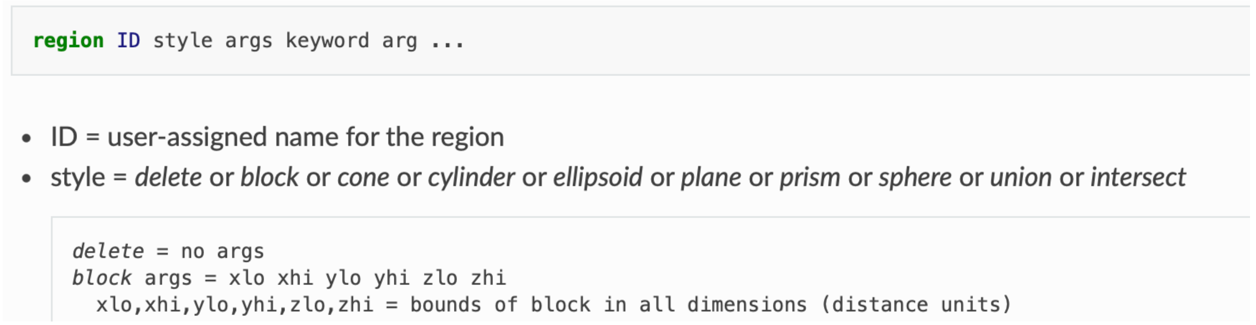
INITIALIZE PARTICLE VELOCITY

Initialization

```
#----------------------------Initialization---------------------
units           lj
atom_style      atomic

lattice         fcc 0.8442
region          box block 0 10 0 10 0 10
create_box      1 box
create_atoms    1 box
mass            1 1.0

velocity        all create 3.0 87287
```

reduced density

commands



```
region ID style args keyword arg ...
```

- ID = user-assigned name for the region
- style = *delete* or *block* or *cone* or *cylinder* or *ellipsoid* or *plane* or *prism* or *sphere* or *union* or *intersect*

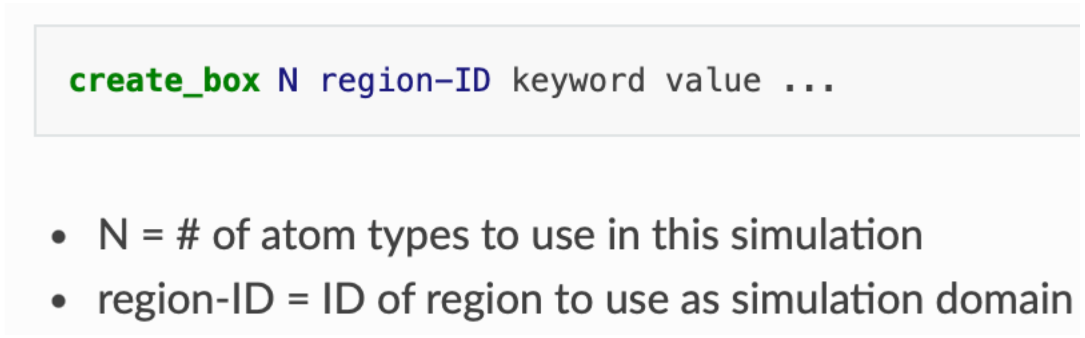
```
delete = no args
block args = xlo xhi ylo yhi zlo zhi
  xlo,xhi,ylo,yhi,zlo,zhi = bounds of block in all dimensions (distance units)
```

Initialization

```
#-----------------------------Initialization---------------------
units           lj
atom_style      atomic

lattice         fcc 0.8442
region          box block 0 10 0 10 0 10
create_box      1 box
create_atoms    1 box
mass            1 1.0

velocity        all create 3.0 87287
```

reduced density

commands



```
create_box N region-ID keyword value ...
```

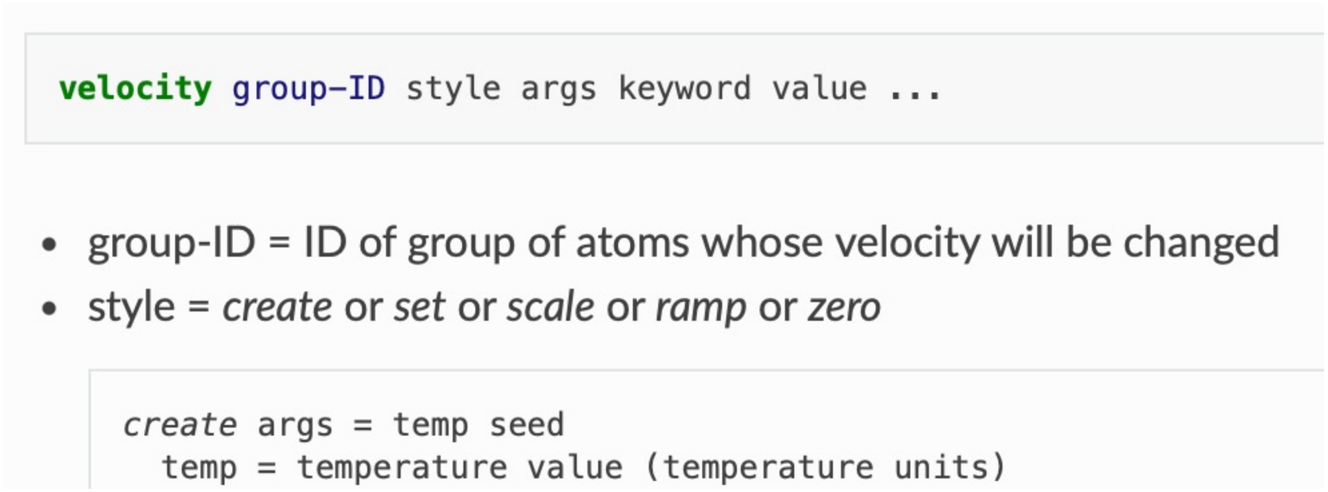
- N = # of atom types to use in this simulation
- region-ID = ID of region to use as simulation domain

Initialization

```
#-----------------------------Initialization--------------------
units           lj
atom_style      atomic

lattice         fcc 0.8442
region          box block 0 10 0 10 0 10
create_box      1 box
create_atoms    1 box
mass            1 1.0

velocity        all create 3.0 87287
```

reduced density

commands



```
velocity group-ID style args keyword value ...
```

- group-ID = ID of group of atoms whose velocity will be changed
- style = *create* or *set* or *scale* or *ramp* or *zero*

```
create args = temp seed
  temp = temperature value (temperature units)
```

Interaction definition

‘pair style’ – pair potential

‘pair_coeff’ – parameters for each pair

‘neighbor’ – types of neighbor lists (bin or nsq)

‘neighbor_modify’ – additional parameters

- ϵ (energy units)
- σ (distance units)
- LJ cutoff (distance units)

$$E = 4\epsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right] \qquad r < r_c$$

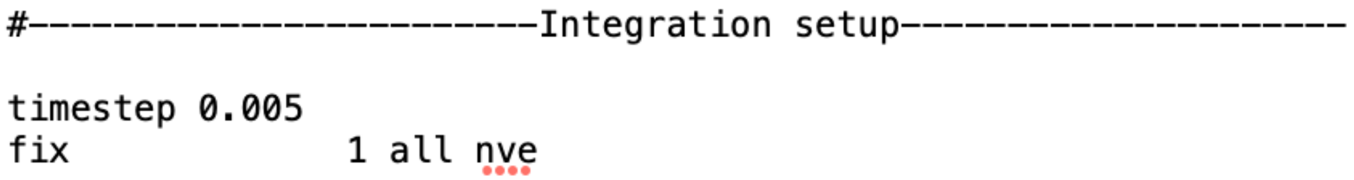
```
#-----------------------Interaction definition--------------------
pair_style      lj/cut 2.5
pair_coeff      1 1 1.0 1.0 2.5

neighbor        0.3 bin
neigh_modify    every 20 delay 0 check no
```

allocation of neighbors
to processors

Integrator setup

TIMESTEP OF INTEGRATION

SET ENSAMBLE TYPE (canonical etc)

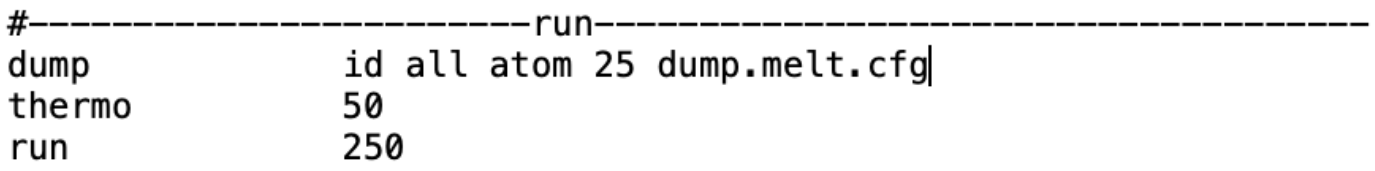


```
#------------------------Integration setup---------------------

timestep 0.005
fix             1 all nve
```

Run parameters

Frequency of thermos data printed to screen

Frequency of configuration data sent to output file



```
#-----------------------run-------------------------------------
dump                id all atom 25 dump.melt.cfg
thermo              50
run                 250
```

Compute and print thermodynamic info (e.g. temperature, energy, pressure) on timesteps that are a multiple of N and at the beginning and end of a simulation.
N=50

Run dynamics for 250 time steps

Last time

Calculate the energy (lattice sums) and the forces on each atom, for a periodic FCC crystal with interatomic interactions described by Lennard-Jones. Use the minimum image convention to compute the forces.

Reminder: Lennard-Jones potential and forces in reduced units

$$\phi^*(r^*_{ij}) = 4\left[\left(\frac{1}{r^*_{ij}}\right)^{12} - \left(\frac{1}{r^*_{ij}}\right)^{6}\right]$$

$$\mathbf{f}^*_{ij}(r^*_{ij}) = -\frac{24}{r^2_{ij}}\left[\left(\frac{2}{r^*_{ij}}\right)^{12} - \left(\frac{1}{r^*_{ij}}\right)^{6}\right]\mathbf{r}^*_{ij}$$

Value	**In reduced units**
Potential energy	$U^* = U/\epsilon$
Temperature	$T^* = k_B T/\epsilon$
Density	$\rho^* = \rho\sigma^3$
Pressure	$P^* = P\sigma^3/\epsilon$
Time	$t^* = t/t_o$, where $t_o = \sigma\sqrt{m/\epsilon}$

Last time

Force acting on an atom at a given time for a Lennard-Jones potential in reduced units:

$$\vec{F}_i = \sum_{j \neq i} \vec{f}_{ij} = \sum_{j \neq i} \left(-\frac{1}{r_{ij}} \frac{d\varphi}{dr_i} \right) \vec{r}_{ij} = \sum_{j \neq i} \frac{24}{r_{ij}^2} \left(2 \left(\frac{1}{r_{ij}} \right)^{12} - \left(\frac{1}{r_{ij}} \right)^{6} \right) \vec{r}_{ij}$$

$$P = \frac{N}{V} k_B T + \frac{1}{3V} \left\langle \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \frac{24}{r_{ij}^2} \left(2 \left(\frac{1}{r_{ij}} \right)^{12} - \left(\frac{1}{r_{ij}} \right)^{6} \right) \right\rangle$$

```
%  we calculate force (fx,fy,fz), energy (u), and
%  part of the pressure (w)
%
function[u,w,fx,fy,fz]= fLJsum(a,n,rc,x,y,z)
%  set force components, potential energy, and pressure to 0
%  fx,fy,fz are each vectors, with an entry for every atom
```

fLJsum.m part1

```
%  we calculate force (fx,fy,fz), energy (u), and
%     part of the pressure (w)
function[u,w,fx,fy,fz]= fLJsum(a,n,rc,x,y,z)
%  set force components, potential energy, and pressure to 0
fx=zeros(n,1);
fy=zeros(n,1);
fz=zeros(n,1);
u = 0;
w = 0;
for i = 1:n-1  % note limits
    ftx = 0;
    fty = 0;
    ftz = 0;
    for j=i+1:n  %  note limits
% mimimum image convention
        dx = x(j) - x(i);
        dy = y(j) - y(i);
        dz = z(j) - z(i);
        dx = dx - round(dx);
        dy = dy - round(dy);
        dz = dz - round(dz);
        dist = a*sqrt(dx^2 + dy^2 + dz^2);
        if dist <= rc
```

fLJsum.m part2

```
        if dist <= rc
            dphi = (2/dist^(12)-1/dist^6);
            ffx = dphi*a*dx/dist^2;
            ffy = dphi*a*dy/dist^2;
            ffz = dphi*a*dz/dist^2;
            ftx = ftx + ffx;
            fty = fty + ffy;
            ftz = ftz + ffz;
            phi = (1/dist^(12)-1/dist^6);
            u = u +  phi;
            w = w + dphi;
%  add -f to sum of force on j
            fx(j) = fx(j) - ffx;
            fy(j) = fy(j) - ffy;
            fz(j) = fz(j) - ffz;
        end
    end
  %  sum up force on i (fi)
    fx(i) = fx(i) + ftx;
    fy(i) = fy(i) + fty;
    fz(i) = fz(i) + ftz;
end
%  need to multiply LJ by 4 and force and pressure by 24
%  also need to correct sign in f
u = 4*u;
w = 24*w;
for i=1:n
    fx(i) = -24*fx(i);
    fy(i) = -24*fy(i);
    fz(i) = -24*fz(i);
end
```

MDLJ

```
function[un,kn,en,tn,pn]= MDLJ(nc,density,tin,nsteps,dt)

% initialize positions and velocities
[n,x,y,z,vx,vy,vz] = initLJMD(nc,tin);

% calculate some useful quantities
vol = n/density;
a = vol^(1/3);
rc = a/2;

% calculate initial energy and forces
[u,w,fx,fy,fz] = fLJsum(a,n,rc,x,y,z);
```

Now use the Verlet algorithm to advance the position of the atoms to the next time step.

MD: integration scheme

Use the Verlet algorithm to advance the position of the atoms to the next time step. Calculate positions, velocities, kinetic energy.

$$\vec{r}_i(t+\delta t) = 2\vec{r}_i(t) - \vec{r}_i(t-\delta t) + \vec{a}_i(t)\delta t^2 \qquad \vec{a}_i(t) = \vec{F}_i(t)/m_i$$

$$\vec{v}_i(t) = \frac{\vec{r}_i(t+\delta t) - \vec{r}_i(t-\delta t)}{2\delta t}$$

The first evaluation at $-\delta t$ is done using the following equation:

$$\vec{r}_i(-\delta t) = \vec{r}_i(0) - \vec{v}_i(0)\delta t + \frac{1}{2}\vec{a}_i(0)\delta t^2$$

MDLJ

```
% now start the time stepping with the verlet algorithm
% initialize variables
xold = zeros(n,1);
yold = zeros(n,1);
zold = zeros(n,1);
xnew = zeros(n,1);
ynew = zeros(n,1);
znew = zeros(n,1);


 % first find the positions at t-dt
```

MDLJ

```
% now start the time stepping with the verlet algorithm
% initialize variables
xold = zeros(n,1);
yold = zeros(n,1);
zold = zeros(n,1);
xnew = zeros(n,1);
ynew = zeros(n,1);
znew = zeros(n,1);


 % first find the positions at t-dt

for i=1:n
    xold(i) = x(i) - vx(i)*dt/a + .5*fx(i)*dt^2/a;
    yold(i) = y(i) - vy(i)*dt/a + .5*fy(i)*dt^2/a;
    zold(i) = z(i) - vz(i)*dt/a + .5*fz(i)*dt^2/a;
end
```

MDLJ

```
% start the time steps
for j=1:nsteps
    k = 0;
 %  find positions for time t + dt
 %  find velocities for time t
 %  find kinetic energy for time t
    for i=1:n
        xnew(i) = 2*x(i) - xold(i) + fx(i)*dt^2/a;
        ynew(i) = 2*y(i) - yold(i) + fy(i)*dt^2/a;
        znew(i) = 2*z(i) - zold(i) + fz(i)*dt^2/a;
        vx(i) = a*(xnew(i) - xold(i))/(2*dt);
        vy(i) = a*(ynew(i) - yold(i))/(2*dt);
        vz(i) = a*(znew(i) - zold(i))/(2*dt);
        k = k + vx(i)^2 + vy(i)^2 + vz(i)^2;
    end
    k = .5*k;
    temp = 2*k/(3*n);
```

MDLJ

Now calculate energies per atom, temperature, pressure.
Then update positions for the new time step.
Calculate the force again and close the time stepping loop.
Done!

MDLJ

```
   %  create time series of values
       e = k + u;
       un(j) = u/n;
       kn(j) = k/n;
       en(j) = e/n;
       tn(j) = temp;
       pn(j) = density*temp + w/(3*vol);

% reset positions for next time step
    for i=1:n
        xold(i) = x(i);
        yold(i) = y(i);
        zold(i) = z(i);
        x(i) = xnew(i);
        y(i) = ynew(i);
        z(i) = znew(i);
    end
% calculate force and energy at new positions for next cycle
    [u,w,fx,fy,fz] = fLJsum(a,n,rc,x,y,z);
end
```